MILESTONE 3 – FINAL REPORT

PROJECT: TABLE FOR FOUR

TEAM: WILDCATTS ANALYTICS

CATHERINE TOLLEY
TAMARA WILLIAMS
TOM ALIG
SHEELA RAO

May 29, 2018

TABLE OF CONTENTS

PROBLEM STATEMENT

With the world's ninth-largest population (127 million) [1] Japan has approximately 107 million mobile phone users [2]. Adding in roughly 24 million foreign tourists with their smartphones, these mobile phone users represent a marketing opportunity for the travel, restaurant, and entertainment business. Recruit Holdings Co., Ltd. is a global holding company, headquartered in Japan. For this engagement, we will be targeting the Media and Solutions vertical of Recruit Holdings' (RH) business, focusing on the company's offerings in the restaurant sector.

RH's AirREGI point-of-sale (POS) product currently offers cash register and table management services via iOS tablet and mobile devices for restaurants in Japan. A related RH business, Hot Pepper Gourmet (HPG), a restaurant marketing website and mobile app, provides Japanese diners with a searchable interface to locate restaurants and obtain promotional coupons. The company has targeted these two products for continued growth in 2018. RH has hired WildCATTS Analytics to further its vision of connecting customers with businesses, specifically in the Japan restaurant market. The proposed project will provide new business insights and drive shareholder value.

By tapping into RH breadth of data collection in the Japan restaurant market, WildCATTS Analytics will add predictive modeling to the capabilities of the AirREGI system. Achieving success in the project goals will enable RH to adopt the same capabilities across its multiple product channels worldwide.

In a review of the importance of data to RH, WildCATTS Analytics analyzed the company's 2017 Annual Report. Part of the review was to generate a word cloud of the 154-page document. Notice the word "data" in Figure 1. Its prominence informs us that data is a heavily used word at RH and is viewed as a primary element of the company's success. The word has importance, it is focused on, and it is a major part of the business.

Figure 1

PROJECT GOALS

1. Develop a robust predictive model to predict daily restaurant volume up to 30 days in advance.
2. Develop a dashboard for the AirREGI POS solution to deliver predictions in context, enabling restaurant managers to take appropriate actions, including pushing customized promotions to the HPG marketing product.
3. Develop a prototype of a mobile experience for restaurant clients to interact with and potentially respond to volume predictions.
4. Maximize insights from RH's valuable restaurant sector data with meaningful data visualizations.
5. Recommend strategy to monetize prediction-enhanced AirREGI POS premium-subscription solution.

OVERVIEW OF THE DATA

The primary source of data for this project is the Kaggle Recruit Restaurant Visitor Forecasting data set. The dataset is comprised of seven files at differing levels of granularity, forming a relatively complex data ecosystem. The files and their relationships are shown below in Table 1.

Table	Records	Variables
Total Visit Data	252,108	3 plus modeling target visit count
Date Data	517	3
AirREGI Restaurant Info	829	5
HPG Restaurant Info	4,690	5
AirREGI Reservation Data	92,378	4
HPG Reservation Data	2,000,320	4
Store Relation Data	150	2

Table 1

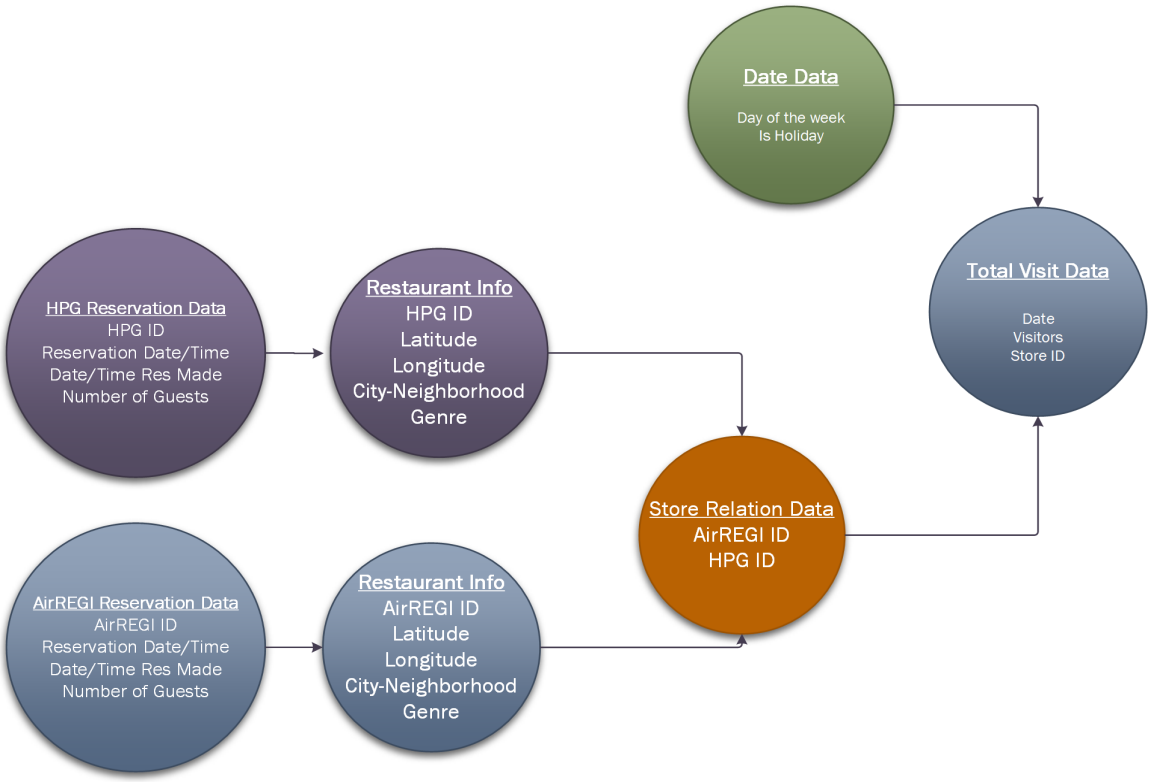

Figure 2 - Visual Representation of Kaggle Data Files

- Approximately 16 months of historical data for the AirREGI restaurants will be used to predict the next 39 days of visitor volume
- Daily visitor counts demonstrate periodic patterns but are less complete for the first 6 months of historical data provided. The predominant period is weekly with the highest counts on Saturdays.
- Many missing daily visitor counts can be attributed to regular weekly open and closed days for each restaurant and to holiday closures.
- Holidays impact restaurant visitor counts, when open.
- Restaurant geographic coordinates are grouped approximately to the labeled geographic area, which has high cardinality.
- Reservations are typically made 24 hours prior to dining and are most often made for the evening meal.
- Reservation data from the AirREGI system are inconsistent during the first year of the data. The usefulness of AirREGI reservations for predictive modeling is questionable.
- Reservation data from the Hot Pepper Gourmet (HPG) web and mobile reservation system are more consistent but contain reservations for only 150 of the 829 AirREGI restaurants. The usefulness of HPG reservation data for predictive modeling is questionable.
- There are outliers in both the daily visitor counts and the reservation data.

Initial investigation into the training data was primarily carried out using R. Some visual exploration was also done in Microsoft PowerQuery/PowerBI and Tableau. The model training data has 252,108 observations of 829 restaurants utilizing the AirREGI POS product, spanning 517 calendar days from January 1, 2016 to April 22, 2017.

Full details of the data exploration work are provided in Appendix B.

DATA TRANSFORMATION AND SHAPING

We evaluated a number of different ways to shape the data. We augmented it with external data from publicly available alternate sources, and we used feature engineering principles to create new features within the data set. And finally, we transformed the data using imputation.

DATA AUGMENTATION AND FEATURE ENGINEERING

Data handling was done as a series of steps, beginning with the base Kaggle data. First, a dataset comprised of all the reservation data was created by joining HPG and AirREGI reservations into a single set and adding the restaurant information such as latitude and longitude. Next, a dataset showing the visits per store and all related restaurant information was created. These two sets, reservations and visits, were used as the base files for augmenting with external data.

In the belief that weather and proximity to various geographical features might impact restaurant visits, we augmented the data with both weather, and “places” data. Weather data was obtained from a weather data EDA shared on the Kaggle site by Hunter McGushion [3]. Using the weather data csv files he provided, along with his mapping between restaurants and the nearest weather collection station, we mapped 14 weather variables onto the base dataset. In addition to weather, we also added geographical data collected from Google Places [4] via a Python script written by a team member. Using the latitude and longitude of each restaurant, HTTP requests were formed to query:

1. subway/train stations within a 500-meter radius of the restaurant
2. museums and zoo within a 500-meter radius
3. bars, malls, and movie theatres within a 500-meter radius

From the results returned by the Google Places query, the distinct number of place names were counted, and appended to dataset to form the new, doubly enhanced dataset with both weather and nearby places.

Predictions will be made at the restaurant level. Historical volume, reservations and the geographic features described above were available at differing levels of granularity. Consolidation of summary

variables at the restaurant level provides insights into AirREGI POS customers and additional features for predictive modeling. Table 2 summarizes features derived across five intuitive categories.

Category	Derived Features
Availability	Number/percent of dates with visitor counts, Number/percent holidays with visitor counts, Mean days per week open
Location	Subway stops, tourist attractions, nightlife, prefecture, zone
Reservations	Number/percent of dates with reservations, mean/median daily reservations, Mean/median daily reserved visitors
	Mean/median reservation lead hours, number of AirREGI and HPG reservations
Volume	Mean/median daily visitors, busiest day of week, mean visits on busiest day, slowest day of week, mean visits on slowest day
Other	Genre, restaurant is also on HPG

Table 2

The final data asset after all new feature additions is depicted in Figure 3.

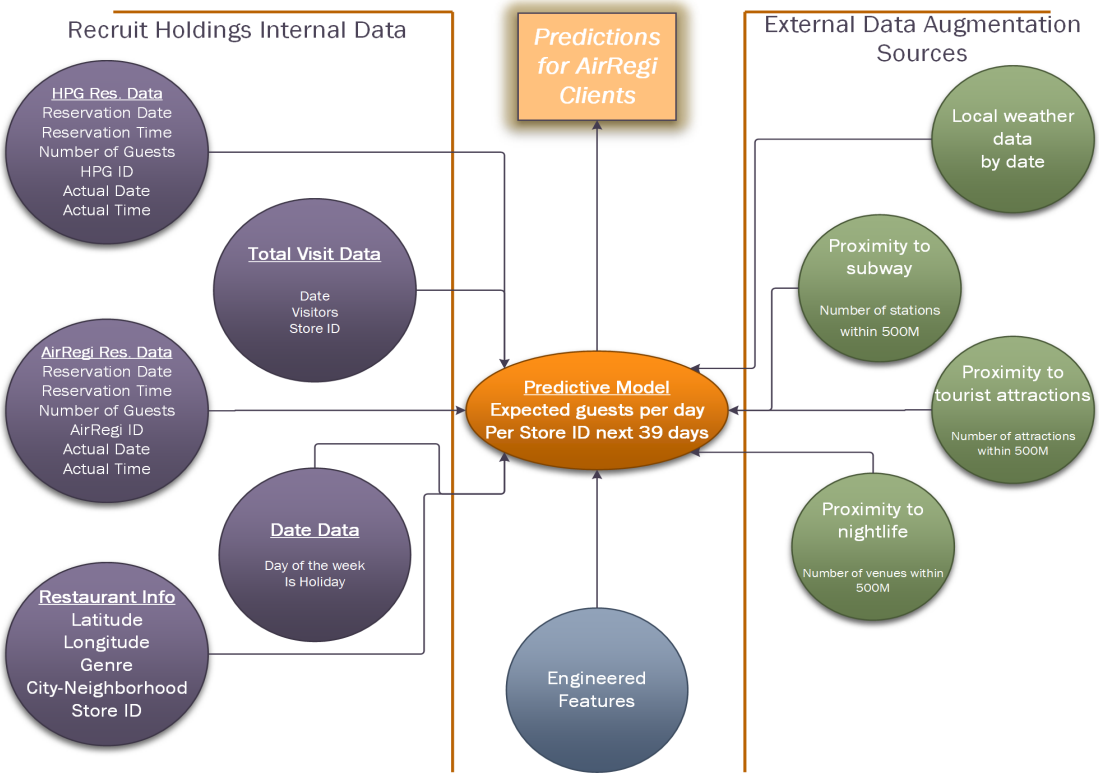

Figure 3 - Enhanced project data

MISSING VALUE TREATMENT

Two modelling paths were evaluated as part of development. One path used ARIMA modelling, the other used boosted trees. In the case of boosted trees, initial models were built without any imputation; missing values were simply converted to 0. Results from this approach proved satisfactory, and all boosted tree models were built without imputation except for a single experiment. To ensure that imputation did not yield additional improvements to boosted tree models an experiment was run using the imputed dataset; this actually yielded a drop in model accuracy, so no future testing was done and this path was abandoned.

For time series models like exponential smoothing and auto ARIMA, a cascading conditional approach was used for imputation.

1. In order to reflect the first day of a restaurant's visitor counts, all NAs before the first visits row were ignored.
2. Missing data every 7 days were interpreted as a regular weekly closure: Impute 0.
3. Missing data at irregular weekdays where the date was a holiday were interpreted as restaurant closure: Impute 0.
4. Remaining unexplained missing values: Impute median visitor count for the restaurant.

The example below illustrates the imputation process for a few restaurants in the month of March 2017.

A	B	C	D	E	F	G
Date	Day	Holiday Flg	Day	air_97c	air_97e	air_9828
3/1/2017	Wednesday	0	1	NA	21	21
3/2/2017	Thursday	0	2	10	NA	13
3/3/2017	Friday	0	3	18	16	18
3/4/2017	Saturday	0	4	27	35	23
3/5/2017	Sunday	0	5	15	10	18
3/6/2017	Monday	0	6	5	1	7
3/7/2017	Tuesday	0	7	6	21	NA
3/8/2017	Wednesday	0	8	NA	13	16
3/9/2017	Thursday	0	9	9	NA	8
3/10/2017	Friday	0	10	16	17	20
3/11/2017	Saturday	0	11	25	22	26
3/12/2017	Sunday	0	12	18	37	NA
3/13/2017	Monday	0	13	12	13	24
3/14/2017	Tuesday	0	14	11	11	NA
3/15/2017	Wednesday	0	15	NA	20	15
3/16/2017	Thursday	0	16	10	NA	15
3/17/2017	Friday	0	17	9	22	17
3/18/2017	Saturday	0	18	28	39	18
3/19/2017	Sunday	0	19	19	54	12
3/20/2017	Monday	1	20	5	26	NA
3/21/2017	Tuesday	0	21	5	25	NA
3/22/2017	Wednesday	0	22	NA	30	17
3/23/2017	Thursday	0	23	20	NA	19
3/24/2017	Friday	0	24	6	34	5
3/25/2017	Saturday	0	25	15	33	7
3/26/2017	Sunday	0	26	11	47	22
3/27/2017	Monday	0	27	NA	7	6
3/28/2017	Tuesday	0	28	8	21	NA
3/29/2017	Wednesday	0	29	16	34	22
3/30/2017	Thursday	0	30	9	NA	17
3/31/2017	Friday	0	31	11	19	29

NA's marked in blue occur once a week - imply a weekly resturant closure

NA's on days with holiday_flg=1 - imply holiday closure

Other NAs - interpreted as true missing data

Figure 4- Illustration of imputation inferences

Outliers were also addressed prior to modeling. For simplicity we employed the 'tsclean' function from the *forecast* package for R, which is known to work well on both seasonal and non-seasonal time series. Finally, in order to predict volume at the restaurant level for future dates, the final modeling dataset was converted to a wide format so that models could be run in a loop for each restaurant.

ANALYSIS OF DATA

Prior to modeling, basic analysis was performed on the data. Correlations between variables were evaluated to identify multicollinearity issues. We also used clustering to segment AirREGI restaurants and add rich features to the modeling data.

CORRELATIONS

The consolidated visitor and restaurant data are shown in a correlation plot in Figure 5. Variables are intuitively correlated with respect to weather, geographic location, and nearby places. Note there is little to no univariate correlation to the target, visitors. Historical visits for each individual restaurant are the most powerful predictor; however, findings suggest clusters of variables may demonstrate a relationship to the target.

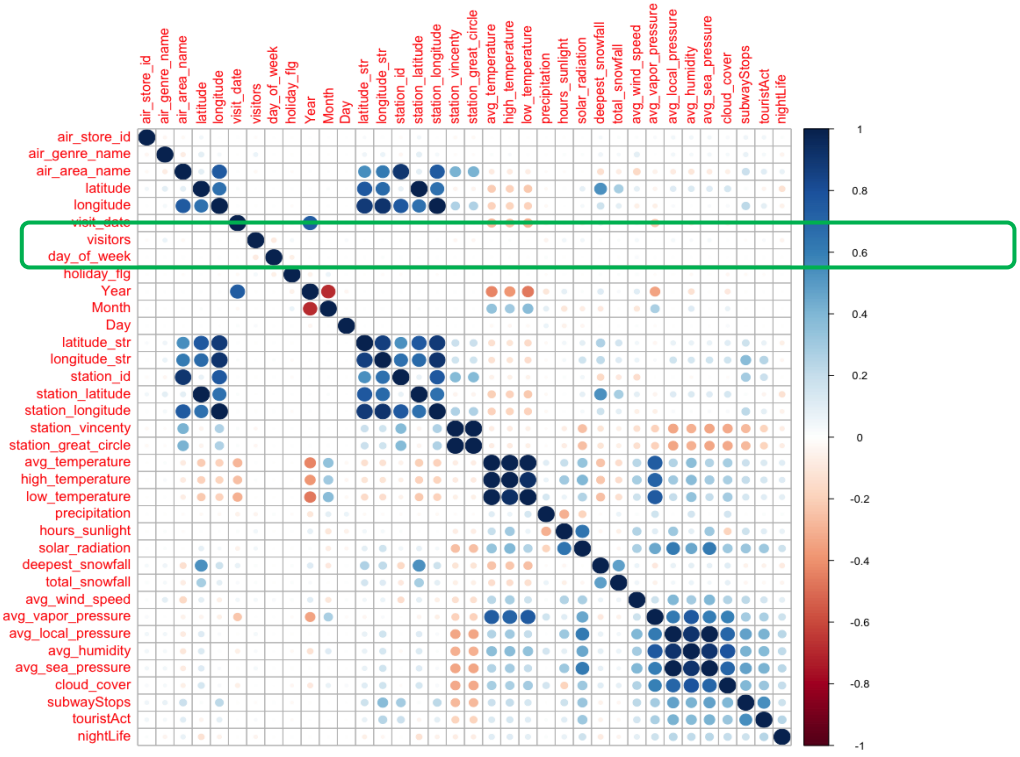

Figure 5 - Correlations among potential restaurant predictors

The correlations in the aggregated reservation data are shown in Figure 6. The number of reserved visitors has little correlation to visitor count. However, the previously noted inconsistencies in the reservation data limit their use in predictive models.

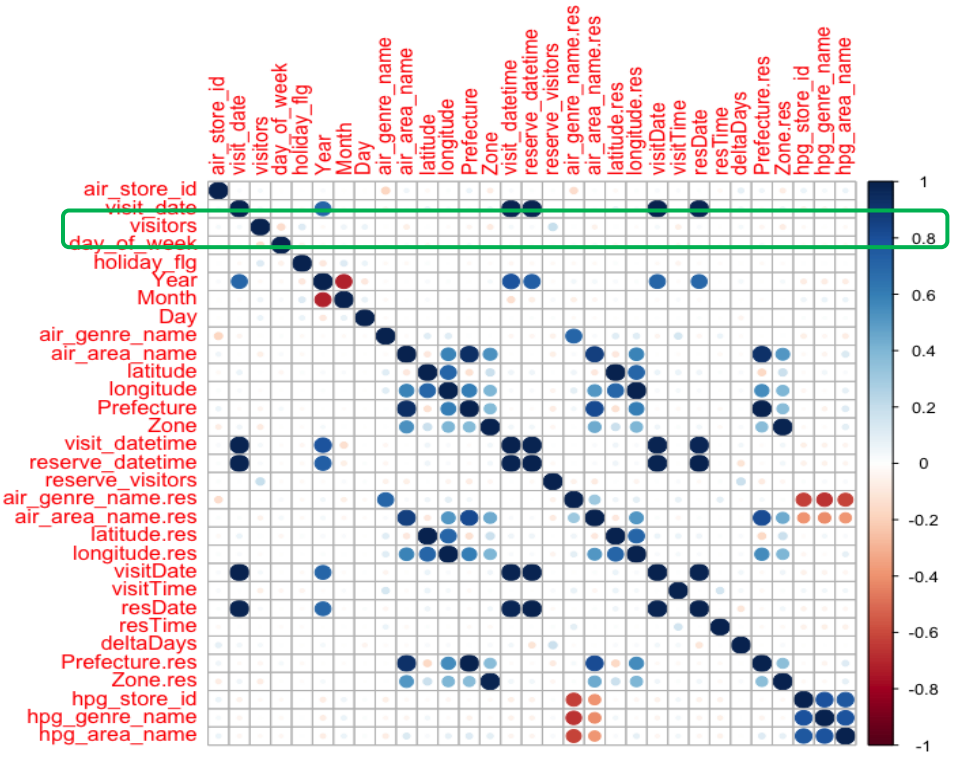

Figure 6 - Correlations among potential reservation predictors

CLUSTERING

The availability of many restaurant characteristics in varying combinations suggested clustering would provide valuable insights. In addition to potential insights, cluster membership was also used in models for predictive value, with solid results.

Cluster profiles provide RH with insights into their AirREGI customer base and provide a basis for targeting the prediction-enhanced AirREGI POS product. The process was as follows:

1. First, group the enhanced restaurant data according to the intuitive categories described in Table 2 in the Augmentation and Feature Engineering section above. For instance, all variables in the “Availability” category are grouped together. Similar groups were created with other variables for “Location”, “Reservation”, and “Volume.”

2. The variables in each group were then standardized with mean of zero and standard deviation of one.
3. Next, all 829 restaurants were clustered with a k-means clustering function. K-means was selected for its relative simplicity. Using the example of the Availability group, a cluster size of six (chosen with the aid of a skree plot) optimized the trade-off between having clusters of meaningful size and ensuring the individual data points were "close" to their assigned cluster.
4. Continuing with the Availability Cluster example, all 829 restaurants were clustered on the availability variables as shown in Figure 7. This same review was done for all clusters on a myriad of variables.
5. Comparison of each cluster with the overall data set population was then performed to identify cluster profiles of interest. For instance, for Cluster 6 in Availability, 66% of its restaurants are based in Tokyo, compared with 54% of all 829 restaurants in the data set.
6. Clusters were built for the remaining variable groupings (Location, Reservations, and Volume) and then profiled as outlined in steps 1 - 5.

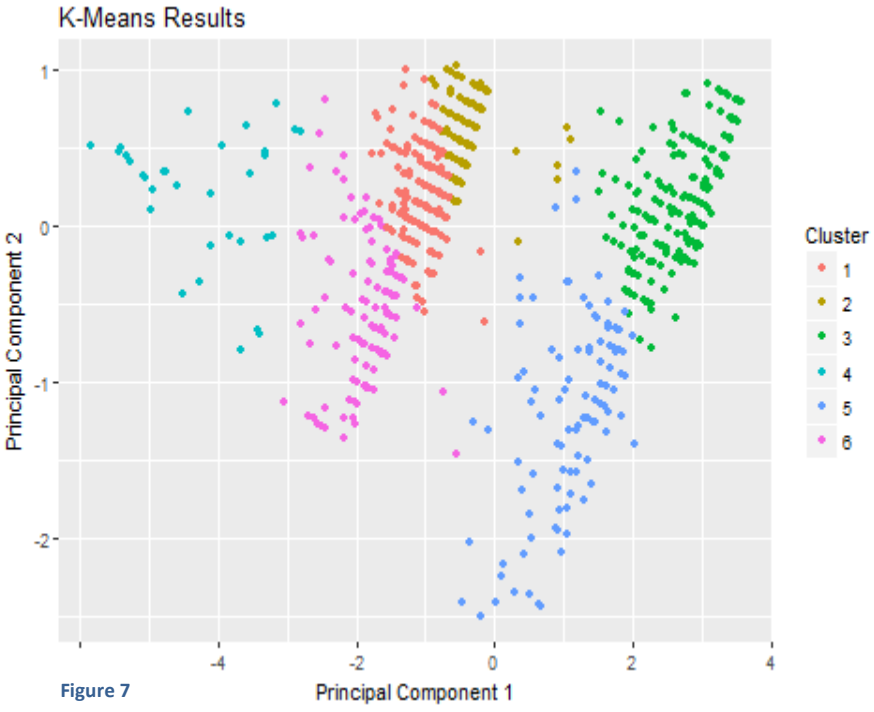

Figure 7

To continue the Availability Cluster 6 example, further review shows that restaurants in Cluster 6 are significantly busier during the weekdays than are restaurants in other clusters. Cluster 6 also has a much higher proportion of restaurants located in Tokyo than the typical restaurant in the data set and tends to be closed for holidays far more often than restaurants in other clusters. The implication is that there is a potential opportunity to boost weekend business. RH could review why the weekend business seems to be not as good as other restaurants and could perhaps try a push notification for drink specials on weekends for restaurants in this cluster.

In total there were four clustering groups comprised of three to eight clusters each. A full exploration of all segments within each group was conducted. For brevity, one interesting example follows. A summary of segment characteristics is provided in Appendix A. Future work may include offering a subscription-based group marketing plan appropriate to specific AirREGI restaurant segments.

CLUSTER INSIGHTS

Restaurants were clustered on location variables as depicted in Table 2. There are some potentially useful findings about Location Cluster 5, which is named “Tourist in Tokyo” and is comprised of 53 restaurants. Table 3 summarizes some key findings for Tourist in Tokyo.

	ALL	Tourist in Tokyo
Friday as busiest day of week	22%	43%
Saturday as busiest day of week	48%	26%
Located in Tokyo Prefecture	54%	96%
Monday as slowest day of week	32%	45%
Tuesday as slowest day of week	21%	8%
Number of tourist attractions nearby	3	12
Number of nightlife activities nearby	7	3
Number of subway stops nearby	1	3

Table 3

Tourist in Tokyo restaurants are located near many more tourist attractions than typical, are easily accessible by subway, and are not located near very much nightlife. These facts indicate a family friendly, tourist-type of location. To further that point, they are not very busy on Saturday, and tend to not be slow on Tuesday, when compared to all the restaurants in the data set. This indicates that they

are busy during the week, again lending well to the fact that these restaurants attract people and families who are traveling from out of town to visit Tokyo for the week.
RH happens to own a hotel and travel information site, Jalan.net. RH could leverage its relationship with both the hotel and the restaurant to ensure that restaurants in the Tourist in Tokyo cluster obtain exclusive recommendations from these Tokyo hotels with whom RH has a relationship.

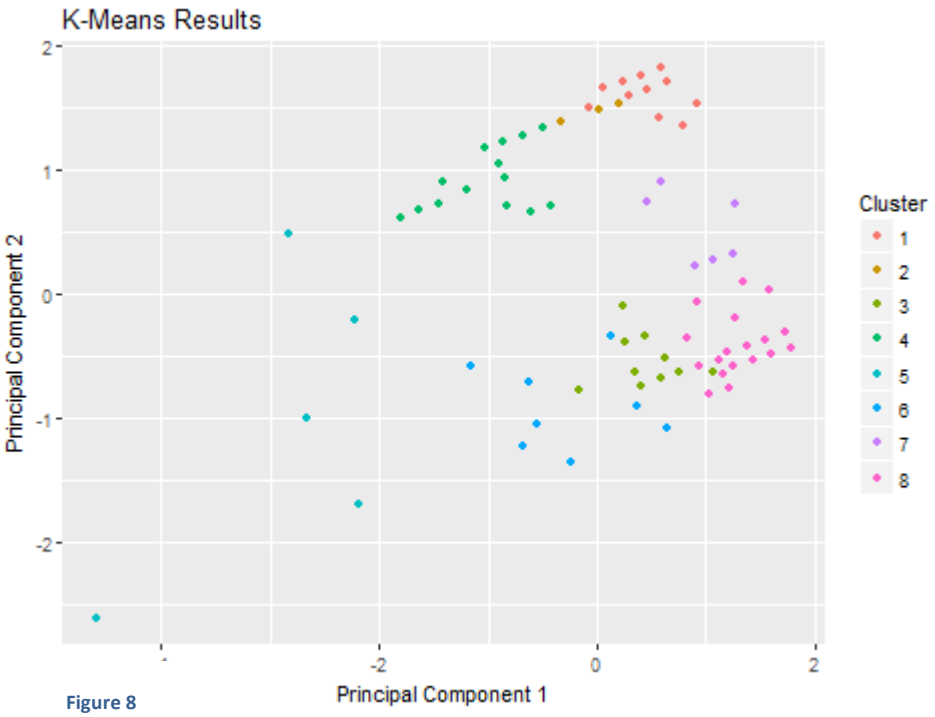

Figure 8

One deliverable of this project is to provide the individual restaurant manager a push promotional message. The manager would use this when forecasted volume is less than desirable. Due to the high number of tourist attractions nearby, the recommendation is for RH to include Tokyo hotels to receive the push marketing notifications for these Location Cluster 5 restaurants.

In addition, advertising at nearby tourist attractions and on subways in Tokyo could help to drive traffic to RH client restaurants. This would all be part of the RH Air Regi Business Support Pack services designed to help restaurants increase business.

MODELING EFFORTS

After initial evaluation of the data and given the time-series nature of the data, we decided to explore both classes of models.

This dataset is from a Kaggle Competition. The root mean squared logarithmic error (RMSLE) was used to evaluate and score submissions. The RMSLE may be thought of as the “distance” between the actual values and the predictions; a lower error value indicates a more accurate model. The Kaggle scoring process serves as an out-of-sample test on the accuracy of the models submitted.

MODEL CLASSES

In an effort to achieve an optimal model, several different techniques were evaluated for use in this project.

TIME SERIES MODELS

Classic forecasting models such as exponential smoothing (ETS) and ARIMA, which stands for Auto Regressive Integrated Moving Average models were built. Time series models require complete historical training data. Therefore, the dataset was imputed as explained above in the 'Missing Value Treatment' section.

a) Exponential smoothing was chosen for its simplicity and
b) auto ARIMA for its ability to handle seasonality, differencing and auto-regression, which is a linear combination of past values of visits. Both ETS and Auto ARIMA models were implemented using Forecast package from Rob J Hyndman [5].
c) Exploring the power of ensemble methods, we used a forecast hybrid forecast ensemble method as well. The hybrid model uses a weighted average of several models such as an auto-selected autoregressive integrated moving average; exponential smoothing state space (both ETS and TBATS); feed forward neural network with a single hidden layer and lagged inputs; and forecasts of loess-based seasonal decomposition.

Of the three methods, the ensemble model from the forecastHybrid R package [6] yielded the best performing set of models in the class.

Note: for all of the time series classes of models:
Imputed air_visit dataset in conjunction with holiday information was reshaped into the wide form with a column representing each restaurant while the rows represented visits for each day. A loop was run for all 829 restaurants and individual models were built.

It was challenging to determine the best performing sets of models. There was no single time series modeling class that worked well for all restaurants. So, reviewing accuracy for randomly selected restaurants served best. AIC (Akaike's Information Criterion) and RMSE (root mean square error) were used to check performance.

ETS

An Exponential Smoothing (ETS) model was implemented without any transformations in the data. The package helps pick type of smoothing automatically based on minimizing AIC . For the best ETS set of models, best AIC was 1191.18. Here's an example restaurant 10's residuals. Both ACF and PACF values are within acceptable limits.

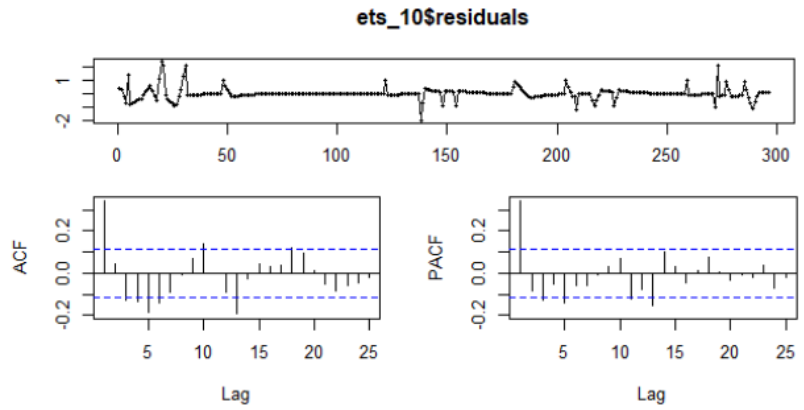

Figure 9 - Residuals Plot ETS model for restaurant 10

AUTO ARIMA

Although one can run ARIMA models and select the autoregressive (p), differencing (d) and moving average (q) parameters, the auto ARIMA model is the best choice as it selects appropriate values for each parameter. The algorithm selected ARIMA(1,0,0) for (p,d,q) as the best auto ARIMA set of models. The AIC value was way better than the ETS set of models, at 292.33. Here's an example restaurant 10's residuals plot, which are within normal limits.

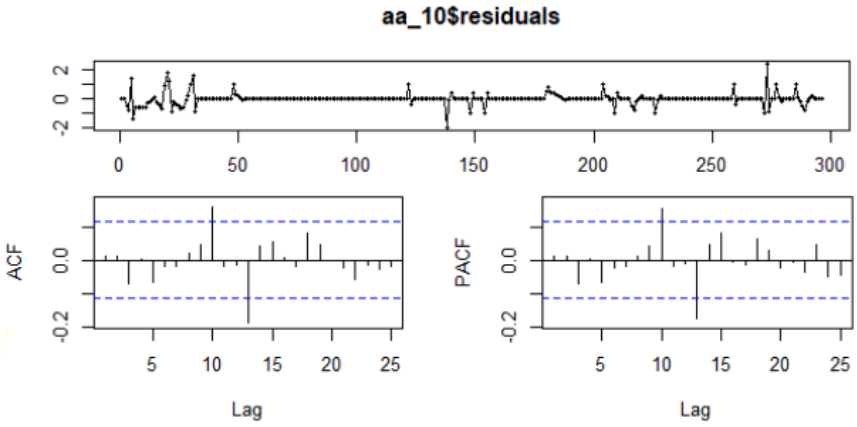

Figure 10 - Residuals Plot auto-ARIMA model for restaurant 10

We explored using the is.holiday flag and clustering as external regressors to this model. The resulting models performed worse than auto ARIMA without regressors.

ENSEMBLE TIME-SERIES MODELS

The forecastHybrid package was used to explore an ensemble of times series forecasting methods. It provides convenient access to ensemble time series forecasting methods, in any combination of up to five of the modelling approaches from Hyndman's forecast package. These are auto-selected autoregressive integrated moving average; exponential smoothing state space (both ETS and TBATS); feed forward neural network with a single hidden layer and lagged inputs; and forecasts of loess-based seasonal decomposition.

This package allowed using the weighted average of ETS, auto-ARIMA, neural net and TBATS. This model further improved the model score, an out-of-sample RMSLE, to 0.573, while the winning XGBoost model had a score of 0.571.

NON-TIME SERIES MODELS

In addition to time series forecasting, several alternate modeling techniques were also evaluated. Tree-type models are known to be effective at modeling complex data; so boosted trees models were tried as one possible method for modeling this data. Classic approaches like Generalized Linear Modeling (GLM), and more contemporary algorithms like Neural Networks were also tried in order to test a variety of non-time series modeling techniques.

The following visual (Figure 11) provides a birds-eye view of both time series and non-time series model classes that improved the performance of our models. As noted, the best performing model is the XGBoost model with features and clustering. Each of these model classes are described in more detail in the subsections below, in the order of least to best performing models.

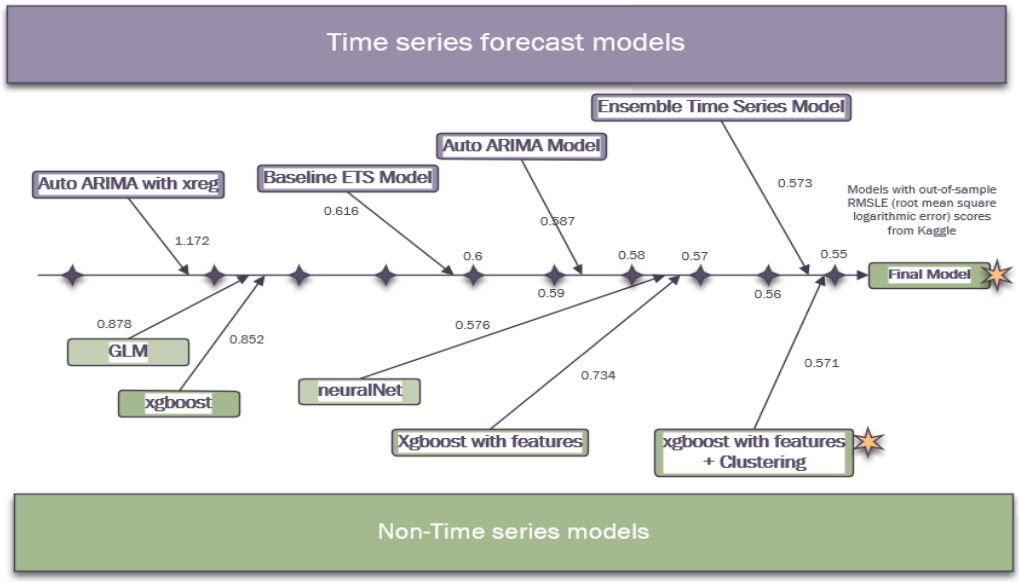

Figure 11 - Model Performance Summary

GLM

Using the R GLM package, models were built using Gaussian, Poisson and Negative Binomial as the family function. In all three models, the formula used for fitting the variables was number of visitors regressed against all the other variables in the feature engineered dataset. In all 3 cases the prediction results were less accurate than the original boosted tree model; additional exploration on this path was halted.

NEURAL NET MODELS

Artificial neural networks allow complex nonlinear relationships between the response variable and its predictors. A simple neural network model with one hidden layer was explored. The resulting set of models for the 829 restaurants yielded the fourth best set with an out-of-sample RMSLE score of 0.576.

EXTREME GRADIENT BOOSTED TREE (XGBOOST) MODELS

Boosted trees are attractive for handling complex patterns and relationships in the data. Time series concepts address the periodicity of visitor volume. Our modelling effort took two approaches to using boosted tree algorithms on this dataset. The first boosted tree approach used the R package 'ForecastXGB' which is under development by Peter Ellis [7]. Ellis describes his package as follows:

> *The forecastxgb package aims to provide time series modelling and forecasting functions that combine the machine learning approach of Chen, He and Benesty's XGBoost with the convenient handling of time series and familiar API of Rob Hyndman's forecast.* [8]

The second approach used XGBoost [9] to model the data without attempting to model the time series explicitly. Incrementally complex models were built using the raw data and adding additional data features: Kaggle data only, Kaggle + weather, and so on. The XGBoost algorithm selects the default direction when data are missing [9]. Since test runs with and without imputation of missing values showed no advantage to doing imputation, missing values received no treatment in the boosted tree models. The effects of the data feature additions is detailed in the Model Comparison and Feature Engineering Impact section below.

Hyperparameters for the XGBoost model were determined using the R gridSearch package [10] over combinations of the hyperparameters. The algorithm tried 6,000 different combinations and used 10-fold cross validation to select the best performing set of parameters. The specifics on the hyperparameters used are detailed in the model recommendations and in the R code (Appendix D)

BEST MODEL

In total, the team built and evaluated 40+ separate models. Iterative model development was done, and a variety of modeling approaches tried; for completeness, the list of models and their Kaggle scores are summarized in Appendix C. Ultimately, the best results came from an XGBoost model built using a dataset which had been augmented with weather data and supplemented with engineered features. Feature engineering and model development happened concurrently; as each new feature came on-line, it was added to the training set and evaluated. The incremental improvements gained by each change in the dataset is summarized in Table 4.

There was a great deal of discussion on the Kaggle forums for this competition on the importance of Golden Week, an important group of holidays between April 29 and May 5. Discussions indicated the impact of how Golden Week was handled would have an effect on the final model. Indeed, we see in our results the single biggest gain came from adding an explicit flag for Golden Week to the dataset.

As described in the "Clustering" section above, the team identified four distinct ways the data could be clustered: availability, location, reservations and volume. Restaurants were clustered for each of these "points of view" on the data, and the cluster membership was added to the training dataset as four new features. Similar to adding the feature for Golden Week, adding the features for cluster membership had a significant positive result on the model accuracy, as seen in Table 4.

Submission Description	Private Score	Public Score
Using only Kaggle-provided data	0.834	0.852
Adding the weather data	0.783	0.768
Adding “is.golden.week” holiday flag	0.635	0.611
Adding clusters	**0.591**	**0.570**

Table 4 - Impact of dataset changes on XGBoost model

Visualization of the total number of predicted visitors per day over the 39 day period is shown in Figure 12.

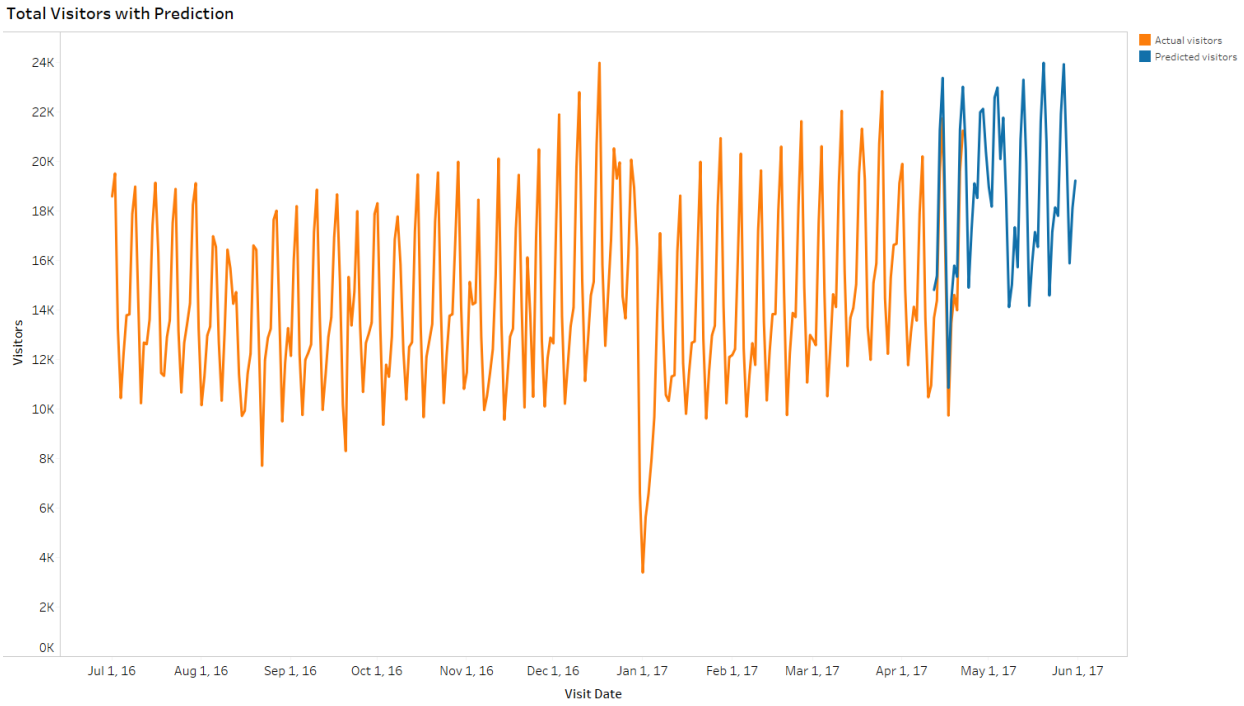

Figure 12 - Actual Visit Data and the 39 Days of Predicted Visits

COMPARISON TO OTHER EFFORTS

Since this these data came from a completed Kaggle competition, there are published efforts outlining the approaches taken during the active competition. Also, team 56 worked with the same data set.

KAGGLE

WildCATTS Analytics developed a solution independent of the existing solutions from the Kaggle competition to ensure a unique approach. Our final model solution is within .09 of the winning Kaggle solution's score. The winning approach confirms our selection of a gradient boosted modelling algorithm. Our use of cluster membership and proximity to subways, attractions, and nightlife is a unique approach. Adding these features to the winning Kaggle approach of including proximity to holidays and individual restaurant volume variability may further improve the performance of the Table for Four predictive model. The winning Kaggle solution was built using a large number of engineered features. The original data set from RH has a total of 105,907 observations of 18 variables; the winning Kaggle solution was built using 1,322,337 observations of 224 variables. [11] The winning solution used a number of "look ahead" and "look behind" features for modeling the proximity of a holiday to a given date. It also uses minimum, maximum, mean, median, standard deviation and skew values for each store for each day. The solution was built using the Light GBM algorithm. It is a gradient boosted framework, which uses tree-based learning. "LightGBM grows tree by leaf-wise (best-first). It will choose the leaf with max delta loss to grow. When growing same leaf, leaf-wise algorithms can reduce more loss than level-wise algorithm." [12]

COLLABORATION WITH OTHER TEAMS

Two teams have developed solutions for the restaurant visitor prediction problem. After both teams had established project goals, business cases naturally diverged, with team 56 focusing on enhancing insights and activities at the restaurant manager level, and WildCATTS Analytics adding the additional layer of insights into AirREGI customers to benefit RH in growing its POS business. Team 56 has developed business insights dashboards in PowerBI, while WildCATTS Analytics developed in Tableau. This provides options for solution deployment using the most compatible technology, as well as building the knowledge base across analytics teams for future projects. There is some conceptual overlap in the deliverables to the restaurant manager audience, providing the CEO an opportunity to evaluate different points of view for this important audience. Future work may include A/B testing of alternative solutions among AirREGI clients.

To limit redundancy and maximize resources, WildCATTS Analytics designated a team member to maintain proactive weekly communication with team 56. As a result, our team was able to maintain focus on a differentiated modeling approach as well as maintain awareness of successes to share and pitfalls to avoid. We assisted team 56 by sharing our data augmentation features at the restaurant level.

DELIVERABLES

In the Table for Four project goals document, we listed five deliverables for this engagement. These deliverables and the specific work supporting each deliverable is covered below.

DELIVERABLE 1- PREDICTIVE MODEL FOR RESTAURANT VISITS

Based on our efforts, we recommend RH use an XGBoost model, using the hyperparameters below in Table 5 , running on a dataset comprised of the air visits data, the flag feature for “is golden week” and the cluster data described in “Clustering” section of this document. Code for this model is given in Appendix D.

XGBoost Hyperparameters	
eta = 0.3	colsample_bytree = 0.6
max_depth = 15	gamma = .2
nrounds=35	min_child = 1
nfold = 10	eval_metric = 'rmse'
subsample = 0.7	objective = "reg:linear"

Table 5 - Hyperparameter values

DELIVERABLE 2- AIRREGI FRANCHISE MANAGER DASHBOARD - DESKTOP VERSION

SCENARIO: Akiko Ito manages multiple restaurants in different genres within prefectures Hyogo, Osaka and Niligata. She visits the AirREGI dashboard on her desktop machine. The landing page shows filters for the three prefectures. Clicking on any restaurant displays a view of reservation data and volume predictions for next few days. Considering the reservation information in conjunction with the predictions, Akiko can take a couple of actions to help boost visitors to these restaurants:

- The ‘select promotion coupon’ button allows her to choose from an existing list of pre-prepared coupons. For example, she can select the ‘Happy Hour coupon’ which will publish coupons to the HPG reservation app.
- The ‘Menu Changes and Optimization’ link allows Akiko to update and optimize the day’s menu and optimize resources.

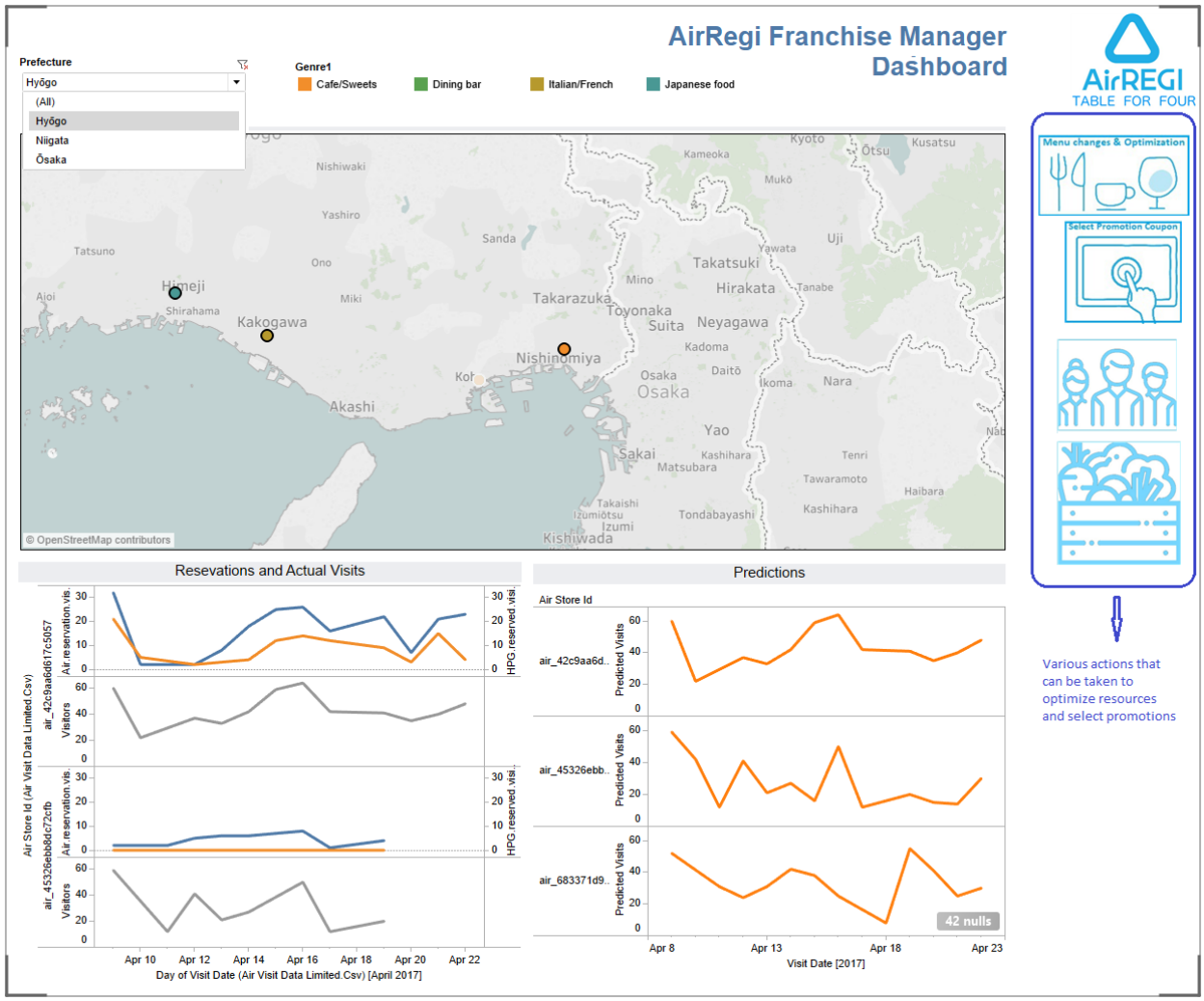

Figure 13 - AirREGI Franchise Manager Scenario

DELIVERABLE 3 – AIRREGI RESTAURANT MANAGER MOBILE EXPERIENCE

SCENARIO: Yuichi Sato is a restaurant manager in the Osaka prefecture. He manages two stores which specialize in serving Teppanyaki. As part of the value add for the Table for Four project, RH can now offer Yuichi a new way to view data on his restaurants. After going through an RH setup experience to select his restaurants, Yuichi can see a map showing his stores, Yuichi's Original Teppanyaki, and Yuichi's Store Two.

USER INTERFACE WALK-THROUGH: Clicking on either location will show a view of that individual restaurant or performing a multi-select will show both. In the example below, the restaurant owner is being presented with a view showing a restaurant's recent counts of actual visitors as the yellow line, in conjunction with the forecasted visitors for the next several days as the blue line. On the same page,

Yuichi is offered icons to go to a Staff Management page as well as an Inventory/Order page to allow him to optimize his staff planning and inventory based on the predicted visitor traffic.

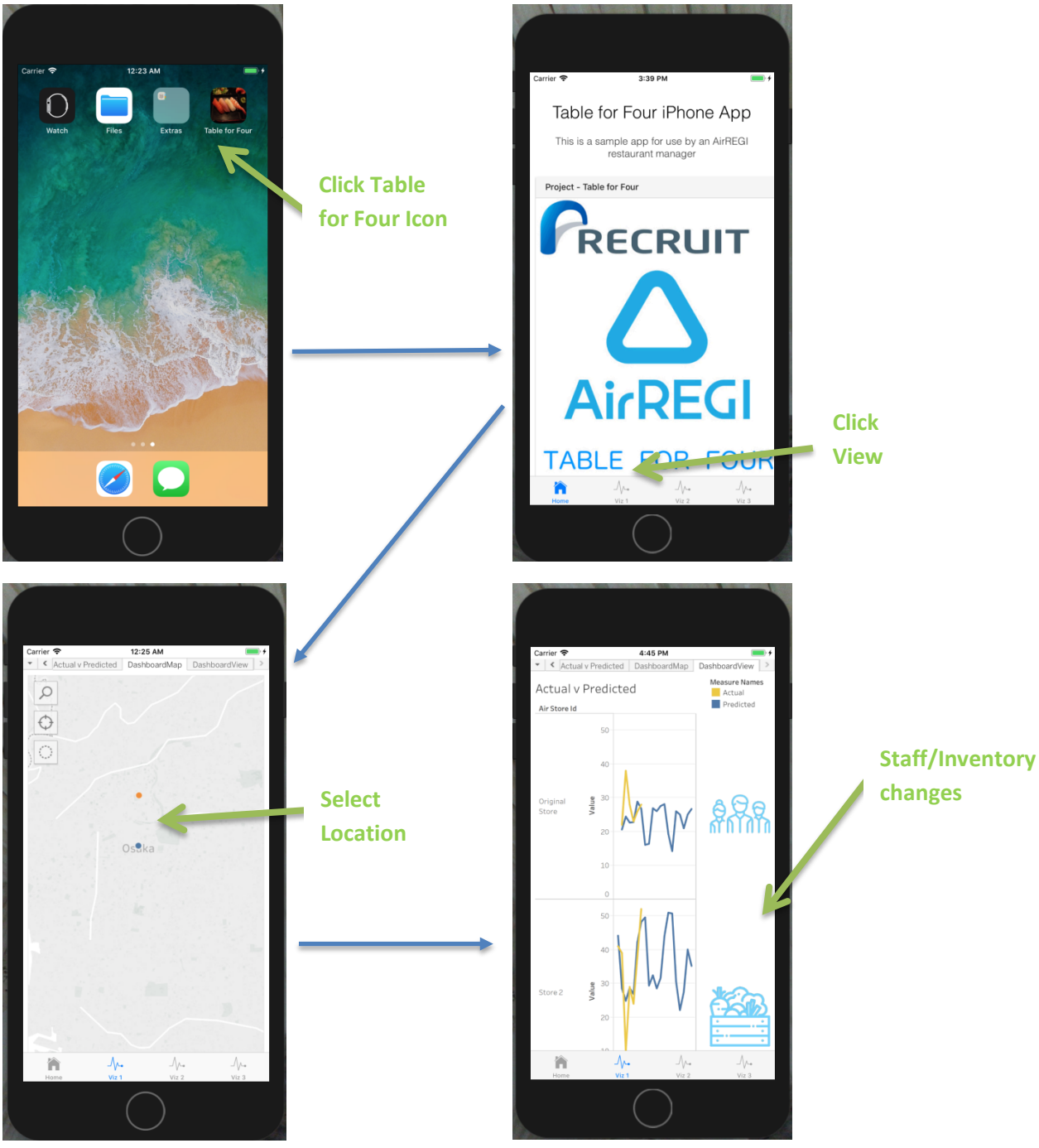

Figure 14 - AirREGI manager UI flow

DELIVERABLE 4 - RH DASHBOARD

RH PREFECTURE MANAGER SCENARIO: Himari Mizutani is a RH manager overseeing their Tokyo, Niigata, and Miyagi operations. The new dashboard from WildCATTS Analytics allows her to easily visualize predicted visitors at her stores (Figure 15). Restaurant prefectures and genres are displayed at the top of the dashboard. The size of the square represents the number of restaurants in the category. Hovering over any category displays the number of restaurants in the category as well as the average daily predicted visitors for the time horizon selected. Himari selects one or more of her prefectures. She can also select a specific genre. A line chart displays current and future predicted visitors. Himari can also scroll through individual line charts for the subset of restaurants she selected to examine their predicted volumes.

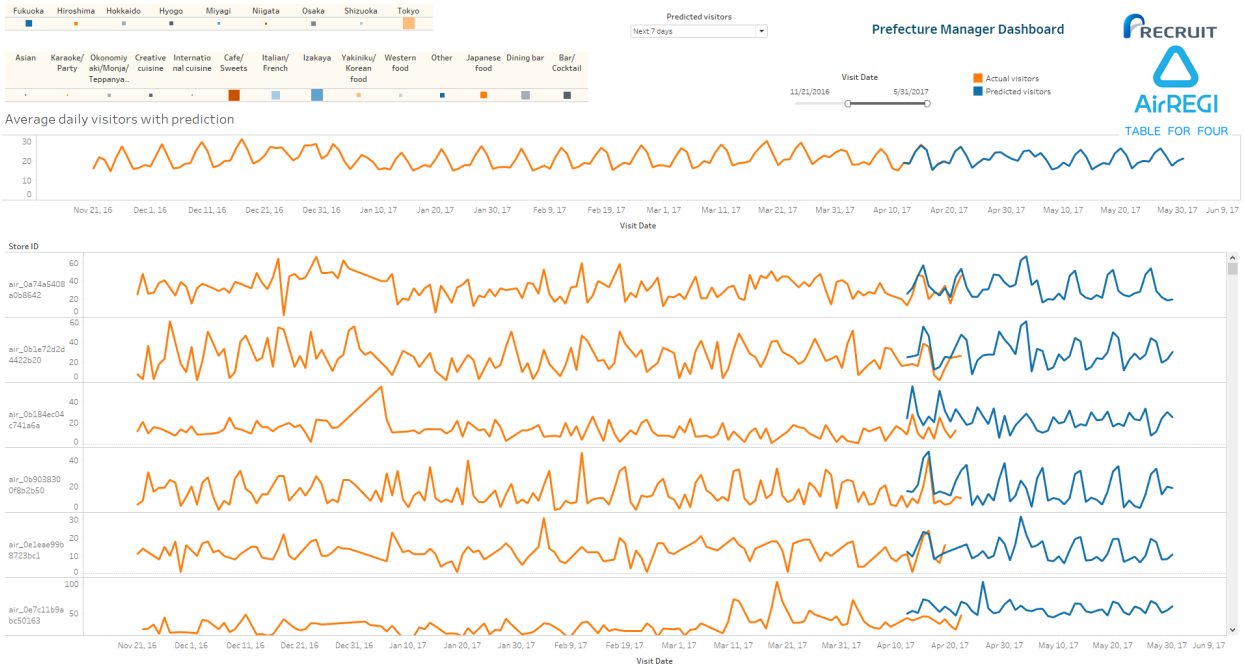

Figure 15 - RH prefecture manager dashboard

EXECUTIVE SCENARIO: Sadaharu Nomo is a RH senior executive based in Tokyo. His dashboard displays a map of Japan with prefectures highlighted that are home to AirREGI restaurants. This serves as a graphic interface point of entry for drilling down to prefectures and restaurants (Figure 16). Selecting a prefecture on the map zooms the inlaid map on the prefecture, showing each restaurant denoted by genre. Hovering on any restaurant will display restaurant characteristics and predicted visitors. Selecting a prefecture also limits the charts at the bottom to the prefecture. The chart on the left shows

total actual and predicted visitors, and the chart on the right displays reservation count by restaurant within each prefecture. This allows RH to monitor the impact of promotions its AirREGI restaurants may push to HPG to address predicted volumes.

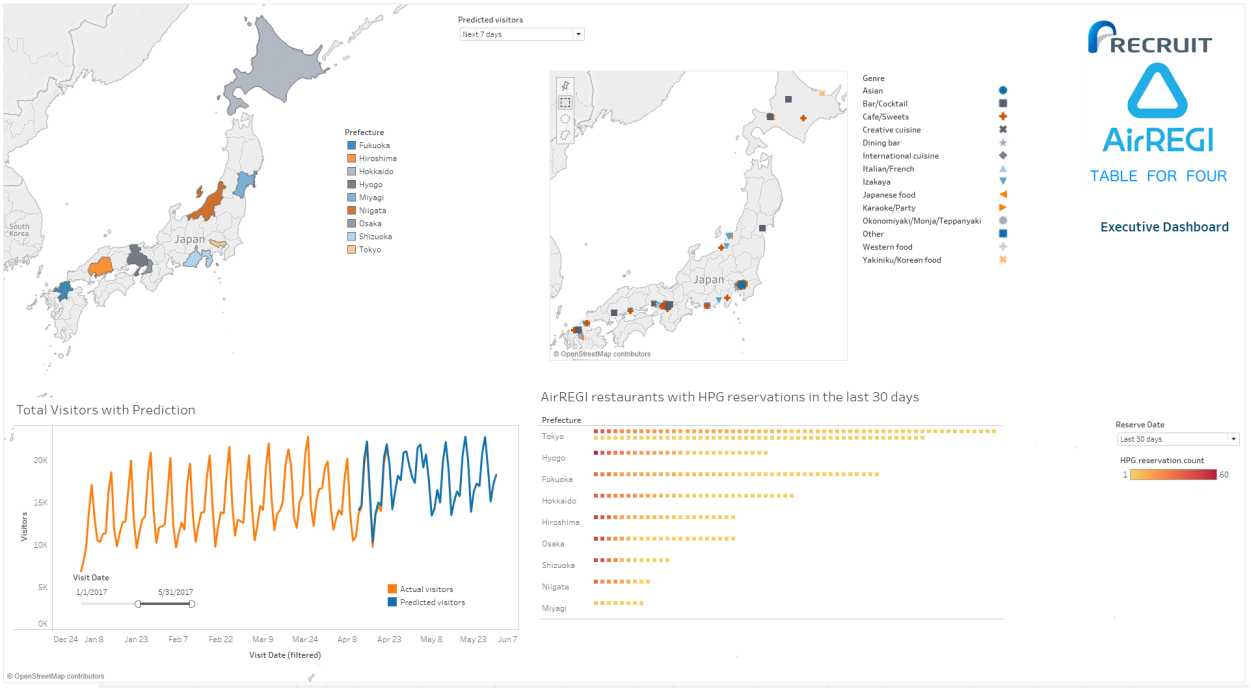

Figure 16 - RH executive dashboard

MONITORING MODEL PERFORMANCE: A third view provides insight into the performance of the prediction model (Figure 17). This dashboard displays the median daily absolute error, in visitors, of the predictions over a user-selected time horizon. Interactive bar graphs displaying the top 10 over- and under-prediction error restaurants allow instant filtering to the restaurant of interest in the line chart below. RH's technical team can monitor model accuracy in real time across all its AirREGI restaurants, and act quickly to rectify any issues. Keeping an accurate model will enhance satisfaction with the prediction-enhanced AirREGI Business Support Pack and support RH's goal of growing its AirREGI business.

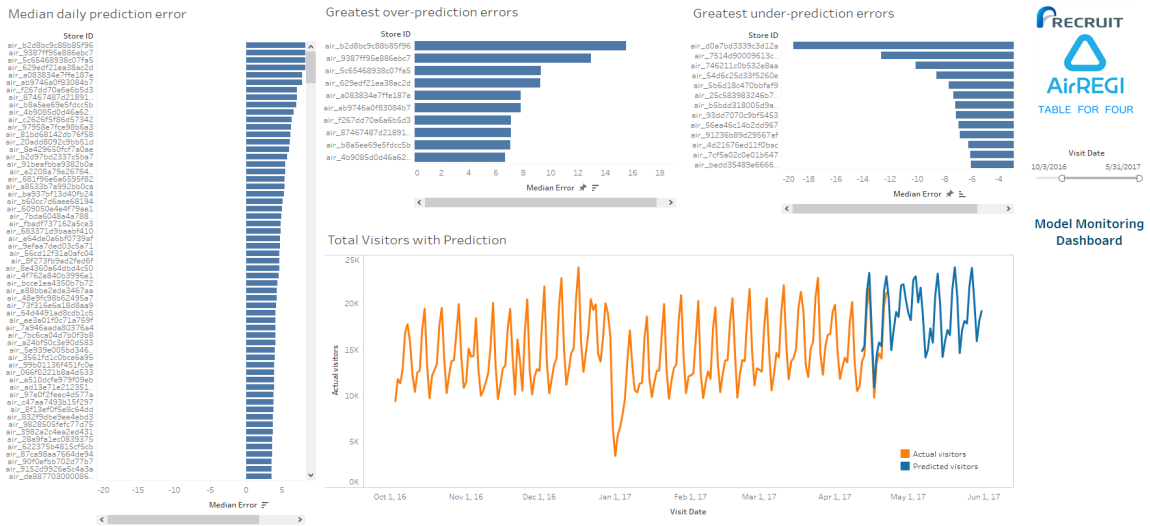

Figure 17 - RH model monitoring dashboard

DELIVERABLE 5 – RECOMMENDED MONETIZATION STRATEGY

RH does not provide details on the cost of its Business Support Pack. According to the New York Times, Yelp charges $250 per month for a reservations-only system. The additional data tools, advertising, website creation, and other services would certainly cost a premium on top of the $250 monthly fee.

From a conservative estimating standpoint, it is assumed that RH will charge $250 per month per restaurant, which equates to $3,000 per year per restaurant. Per RH's annual report, in 2016 it had sales of its Air Platform to restaurants of approximately $342 million USD. Dividing the $342 million by $3,000, it can be estimated that RH's Business Support Pack is in 114,000 restaurants in Japan. Add in an estimated 25% growth, as it is a new product, and first year revenues for the product can be estimated to be over $384 million USD. See Table 6 for calculations. Table 6 further estimates various costs associated with the system. The bottom line comes to over $278 million USD in profit for the first year for RH.

Conservatively estimate that improved ability to predict restaurant volume is providing 1% of value to the entire Business Support Pack, as mentioned above in the form of optimizing a restaurant's cost of goods, staffing, and use of incentives such as coupons to drive traffic. This project would therefore provide approximately $2.78 million USD in value to RH in its first year.

Further recall that RH has many other business lines, including the housing, bridal, educational, automobile, travel, dining, and human resources industries. RH also operates in over 60 countries. The ability to leverage the predictive algorithms and lessons learned from this project and apply elsewhere in RH can yield an exponential amount of value to RH's bottom line.

Recruit Holdings, Premium Data Services		
All financial numbers in USD		
REVENUE	**CALCULATION**	**NOTES**
annual fee per restaurant customer	$3,000	$250 per month per restaurant
estimated number of subscribers at start of year	114,000	Per annual report, $342 million USD in rev. $342mm/$3000 per year = 114,000
first year growth of new customers	28,500	25% growth in premium subscribers per YEAR
estimated revenue year one	**$384,750,000**	(starting subscription * annual fee) + (0.5 * new customers * annual fee) [0.5 new customers because half will be with RH, on average, for full year]
COSTS		
One time fixed cost of software	$10,000,000	Estimate
Ongoing software support per year	$32,062,500	$250 per restaurant per YEAR (also factors in 25% growth)
WildCATTS Analytics Fee for Current Engagement	$234,000	$325 per hour, 4 people, 20 hours per week, 9 weeks
Sales, Promotions, Entertainment, etc.	$64,125,000	$500 per restaurant, per YEAR
estimated total costs year one	**$106,421,500**	
PROFIT		
estimated margin year one	$278,328,500	For entire RH Business Support Pack
	$2,783,285	**Margin attributable to improved restaurant predictive modeling**

Table 6 - Financial Projections

BUSINESS IMPACT

The ability to accurately predict restaurant volume up to 30 days in advance is a small but important part of RH's overall data strategy. WildCATTS Analytics assistance in solving this piece of the puzzle will help RH achieve its goal: *"The more options you have, the richer your life may be. That's why here at Recruit, we are hard at work to give you as many options as possible."*

RH's Business Support Pack for restaurants, according to its annual report, "comes with six core functions: website creation, online advertising, website reservation tools, table management and reservation books, customer management, and messaging. We offer a variety of rates and plans to customers, who select the one that suits their business best." [13] RH began charging restaurants for these services in January 2017.

The ability to predict restaurant volume will aid these core functions by optimizing a restaurant's cost of goods, staffing, and use of incentives such as coupons to drive traffic.

CONCLUSIONS

To ensure RH's continued success with its restaurant-sector offerings, WildCATTS Analytics has incorporated RH's business goals to integrate a predictive model into its AirREGI POS product. Process enhancements resulting from our approach influence both AirREGI restaurants and RH and are annotated with green checkmarks in Figure 18. Managers are prompted to respond to predicted volumes using other RH applications, such as HPG, and RH can monitor the impact of the prediction-enhanced AirREGI POS system.

Clustering AirREGI restaurants based on 4 groupings of restaurant characteristics, including external data about the location, was a powerful analytic strategy. Clusters not only provided insight that can be used to develop marketing and retention strategies for AirREGI restaurants, but also improved the accuracy of the predictive model.

RH will benefit from WildCATTS Analytics' multi-tiered analytics insight dashboards and mobile experiences, benefitting individual and regional restaurant managers using the AirREGI system, RH management overseeing districts of AirREGI restaurant clients, RH executive leadership, and RH's technical team.

Finally, our project approach included foresight towards expansion and repurposing of models and business insights into RH's many lines of business. The work we presented readily translates into the restaurant, beauty, and other service sectors and RH continues to expand its global impact with data-driven solutions.

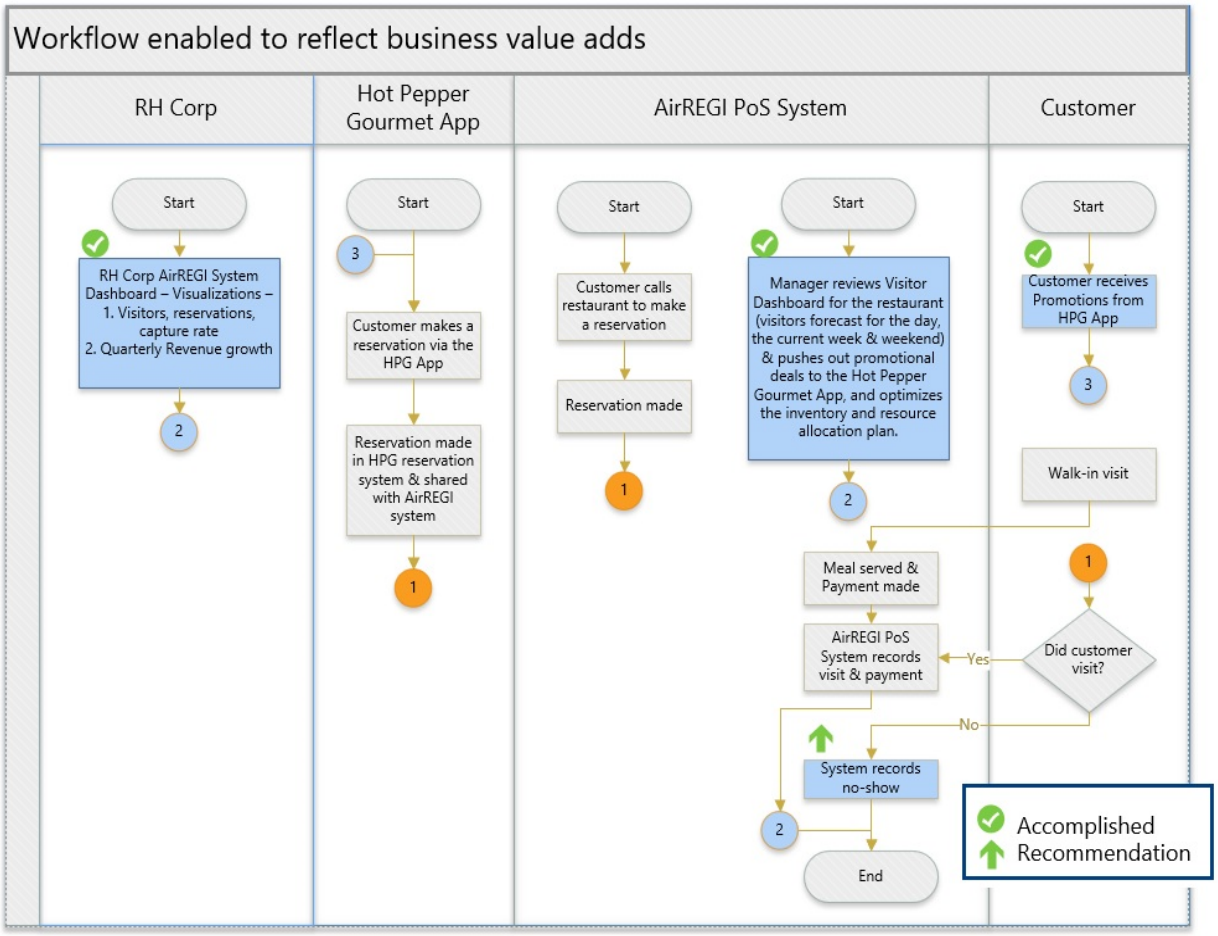

Figure 18 - Table for Four project workflow enhancements

RECOMMENDATIONS

WildCATTS Analytics proposes recommendations along three lines. RH can utilize the prediction-enhanced AirREGI product to boost adoption through restaurant success. The foundational work presented here can be expanded into other markets and other RH product lines. Finally we recommend data collection and integration steps to improve the power of the data sources.

PREDICTION-ENHANCED AIRREGI POS PRODUCT

1. RH can provide online resources providing restaurants with tips for maximizing their use of visitor predictions. Specific topics include understanding the impact of holidays, optimizing perishable food inventory, marketing and promotions, and optimizing operating hours and staffing.

2. As shown in the AirREGI-facing dashboards and app, restaurant managers acting on predicted volume can drive activity to existing RH diner-facing marketing applications. RH may consider marketing activities, such as pushing targeted promotions to HPG, to restaurant managers.
3. The AirREGI POS application can be expanded to give restaurants the ability to monitor their reservations on HPG and check their visibility if reservations decline.
4. RH should monitor the impact of its prediction-enhanced POS application on restaurants sales and use this as a marketing tool to recruit new restaurant customers for both AirREGI and HPG.
5. Insights from clustering and segmentation suggest visitor volume is associated with specific locales. Pushing promotions to local hotels and subway advertising are two of the specific recommendations we have proposed for one of RH's location clusters.

NEW MARKETS AND PRODUCT LINES

1. The predictive model for restaurant volume was developed with RH's global reach in mind. The model can be re-used or re-purposed for markets in other countries as RH expands its hospitality suite of products.
2. This project has demonstrated visitor prediction based on historical volume and a basic set of store characteristics is viable. This capability can be transferred to RH's other lines of business in the service sector.

DATA COLLECTION AND INTEGRATION

1. Collect and expose AirREGI restaurant actual volumes by time of day or by breakfast, lunch, and dinner. Predictions are currently limited to daily volume due to the aggregation of the data at the date level. Restaurant managers will benefit from more specific meal-time or time-of-day predictions by adjusting staffing, seating capacity, and open hours to maximize profitability.
2. Link reservations from both AirREGI and HPG to actual visit data from the AirREGI POS system. Analysis of no-show reservations and integration of reservations and conversions to visits is likely to enhance the accuracy of predictions.
3. Create a data pipeline for reservation data that includes data integrity checks, normalization, and integration of reservations from multiple RH source systems in the hospitality sector. Click- and referral-source information on diners will improve RH's understanding of local dining patterns. This data can also be developed into a customer relationship management tool for AirREGI restaurant clients to provide additional value.
4. Collect demographic data on visitors to the restaurants. This can further aid clustering and marketing of individual restaurants to a targeted population.

PROJECT TEAM

WILDCATS ANALYTICS

This is a team with diverse backgrounds who are passionate about building business value using Analytics. They are all graduating in MS in Data Science (Predictive Analytics) from Northwestern University this June 2018.

	*Catherine Tolley* Catherine has over 10 years of data analytics experience spanning KPI and dashboard development, strategic analysis, data source development, program evaluation, statistical analysis and multivariate modeling. She holds a B.A. from DePauw University and is a graduate of the Centers for Disease Control's Emerging Infectious Diseases fellowship.
	*Tamara Williams* Tamara's background is in Software Engineering, Software Engineering Management, and Software Quality Assurance. She has more than 20 years of experience working at Microsoft on everything from games to Enterprise software. She holds a B.S. in Mathematics from University of Washington, and an MSE from Seattle University.
	*Tom Alig* Tom's background includes 7 years in various Purchasing roles at General Motors, and 17 years in healthcare. He has a B.A. from the University of Michigan, and an MBA from Michigan State University.
	*Sheela Rao* Sheela is an Engineer (CS) with 20+ years of experience in the Software Industry in software product and services companies. She worked at Microsoft for over 13 years in a variety of Program Management functions. Her last role was that of a Patent Analyst, which got her interested in pursuing a formal education in Data Science.

WORKS CITED

[1] Wikipedia, "Japan," [Online]. Available: https://en.wikipedia.org/wiki/Japan.

[2] Statista, "Number of mobile phone users in Japan from 2013 to 2020 (in millions)," [Online]. Available: https://www.statista.com/statistics/274672/forecast-of-mobile-phone-users-in-japan/.

[3] H. McGushion, "Exhaustive Weather EDA/File Overview," [Online]. Available: https://www.kaggle.com/huntermcgushion/exhaustive-weather-eda-file-overview.

[4] Google, Inc., "Place Search," [Online]. Available: https://developers.google.com/places/web-service/search.

[5] R. Hyndman, "Package 'forecast'," [Online]. Available: https://cran.r-project.org/web/packages/forecast/forecast.pdf.

[6] D. Shaub, "Package 'forecastHybrid'," [Online]. Available: https://cran.r-project.org/web/packages/forecastHybrid/forecastHybrid.pdf.

[7] P. Ellis, "forecastxgb-r-package," [Online]. Available: https://github.com/ellisp/forecastxgb-r-package.

[8] P. Ellis, "Timeseries forecasting using extreme gradient boosting," [Online]. Available: http://freerangestats.info/blog/2016/11/06/forecastxgb.

[9] T. Chen, T. He, M. Benesty, V. Khotilovich and Y. Tang, "Package 'xgboost'," [Online]. Available: ftp://cran.r-project.org/pub/R/web/packages/xgboost/xgboost.pdf.

[10] E. Schumann, "gridSearch," [Online]. Available: https://www.rdocumentation.org/packages/NMOF/versions/1.4-1/topics/gridSearch.

[11] "Puipui", "1st Place LGB Model(public:0.470, private:0.502)," [Online]. Available: https://www.kaggle.com/pureheart/1st-place-lgb-model-public-0-470-private-0-502/forks..

[12] ReadTheDocs, "LightGBM Features," [Online]. Available: http://lightgbm.readthedocs.io/en/latest/Features.html.

[13] Recruit Holdings , "Recruit Holdings 2017 Annual Report," [Online]. Available: https://recruit-holdings.com/assets/pdf/annual/2017/annual_2017_en_all.pdf.

APPENDICES

APPENDIX A - CLUSTERING DETAILS

Below is a summary of segment characteristics, based on the analysis described in the Clustering section of this paper. Each of the 25 clusters was reviewed against 28 variables, and compared to the overall population of restaurants.

Group	Cluster. Number	Cluster. Size	Sat.perc. bzy.DOW	Fri.perc. bzy.DOW	Sun.perc. bzy.DOW	Other.DOW. perc.bzy	Pref.1 .Loc	Pref.1. perc
All	**All**	**829**	**0.48**	**0.22**	**0.21**	**0.09**	**Tokyo**	**0.54**
Availability	1	189	0.52	0.17	0.21	0.1	Tokyo	0.52
Availability	2	183	0.62	0.11	0.23	0.04	Tokyo	0.5
Availability	3	193	0.56	0.18	0.25	0.01	Tokyo	0.42
Availability	4	32	0.38	0.16	0.22	0.24	Tokyo	0.56
Availability	5	104	0.38	0.3	0.18	0.14	Tokyo	0.68
Availability	6	128	0.3	0.24	0.15	0.31	Tokyo	0.66
Location	1	31	0.68	0.13	0.1	0.09	Tokyo	0.42
Location	2	12	0.67	0.17	0.17	0	Tokyo	1
Location	3	84	0.52	0.23	0.19	0.06	Tokyo	0.23
Location	4	225	0.48	0.26	0.18	0.08	Tokyo	0.53
Location	5	53	0.26	0.43	0.15	0.16	Tokyo	0.96
Location	6	127	0.35	0.37	0.17	0.11	Tokyo	0.76
Location	7	59	0.61	0.05	0.25	0.09	Tokyo	0.53
Location	8	238	0.53	0.1	0.28	0.09	Tokyo	0.43
Reservations	1	18	0.44	0.5	0.06	0	Tokyo	0.17
Reservations	2	61	0.46	0.34	0.15	0.05	Tokyo	0.36
Reservations	3	162	0.54	0.3	0.11	0.05	Tokyo	0.52
Reservations	4	517	0.45	0.17	0.26	0.12	Tokyo	0.61
Reservations	5	71	0.59	0.21	0.15	0.05	Tokyo	0.24
Volume	1	230	0.52	0.24	0.17	0.07	Tokyo	0.52
Volume	2	47	0.19	0.51	0.13	0.17	Tokyo	0.81
Volume	3	156	0.53	0.22	0.19	0.06	Tokyo	0.47
Volume	4	134	0.47	0.2	0.3	0.03	Tokyo	0.46
Volume	5	35	0.34	0.14	0.34	0.18	Tokyo	0.51
Volume	6	227	0.5	0.15	0.2	0.15	Tokyo	0.59

		Pref.2.Loc	Pref.2.perc	Pref.3.Loc	Pref.3.perc	Mon.perc.slow.DOW	Tues.perc.slow.DOW	Sun.perc.slow.DOW	Other.DOW.perc.slow
All	**All**	**Fukuoka**	**0.15**	**Osaka**	**0.09**	**0.32**	**0.21**	**0.21**	**0.26**
Availability	1	Fukuoka	0.16	Osaka	0.11	0.31	0.22	0.22	0.25
Availability	2	Fukuoka	0.16	HyAgo	0.1	0.35	0.3	0.1	0.25
Availability	3	Fukuoka	0.2	Osaka	0.1	0.33	0.21	0.17	0.29
Availability	4	Fukuoka	0.09	HyAgo	0.09	0.25	0.38	0.09	0.28
Availability	5	Osaka	0.09	Fukuok	0.08	0.38	0.08	0.32	0.22
Availability	6	Fukuoka	0.13	Osaka	0.08	0.31	0.06	0.26	0.37
Location	1	Fukuoka	0.35	Shizuo	0.1	0.35	0.23	0.03	0.39
Location	2	x	0	x	0	0.33	0.17	0.33	0.17
Location	3	Hokaida	0.23	Osaka	0.25	0.37	0.14	0.21	0.28
Location	4	Fukuoka	0.35	Miyagi	0.08	0.32	0.25	0.26	0.17
Location	5	Hyago	0.04	x	0	0.45	0.08	0.26	0.21
Location	6	Osaka	0.23	x	0	0.31	0.12	0.33	0.24
Location	7	Hokaida	0.22	Shizuo	0.22	0.24	0.36	0.12	0.28
Location	8	Hyago	0.16	Fukuok	0.13	0.31	0.25	0.12	0.32
Reservations	1	Fukuoka	0.17	many	0.11	0.28	0.11	0.44	0.17
Reservations	2	Fukuoka	0.18	Hokaid	0.15	0.34	0.26	0.16	0.24
Reservations	3	Fukuoka	0.11	Osaka	0.1	0.36	0.19	0.25	0.2
Reservations	4	Fukuoka	0.16	Osaka	0.09	0.31	0.21	0.19	0.29
Reservations	5	Fukuoka	0.18	Hokaid	0.15	0.34	0.28	0.18	0.2
Volume	1	Fukuoka	0.15	Osaka	0.11	0.31	0.24	0.21	0.24
Volume	2	Osaka	0.11	Hokaid	0.04	0.09	0.11	0.51	0.29
Volume	3	Fukuoka	0.15	Hyago	0.1	0.37	0.21	0.15	0.27
Volume	4	Fukuoka	0.16	Hyago	0.1	0.33	0.27	0.14	0.26
Volume	5	Fukuoka	0.17	Hyago	0.11	0.29	0.26	0.06	0.39
Volume	6	Fukuoka	0.18	Osaka	0.08	0.36	0.18	0.24	0.22

		Num.Days.Wk.Open	Genre1	Genre.1.perc	Genre2	Genre.2.perc	Genre3	Genre.3.perc
All	**All**	**3.79**	**Izakaya**	**0.24**	**Cafe/Sweets**	**0.22**	**Dining Bar**	**0.13**
Availability	1	3.31	Cafe/Sweets	0.27	Izakaya	0.13	Dining Bar	0.13
Availability	2	3.83	Izakaya	0.27	Cafe/Sweets	0.21	Dining Bar	0.16
Availability	3	5.83	Izakaya	0.27	Cafe/Sweets	0.18	Dining Bar	0.15
Availability	4	1.2	Izakaya	0.34	Cafe/Sweets	0.19	Italian/French	0.19
Availability	5	5.05	Izakaya	0.26	Cafe/Sweets	0.25	Japanese	0.13
Availability	6	3.04	Izakaya	0.25	Cafe/Sweets	0.2	Italian/French	0.16
Location	1	3.91	Izakaya	0.65	Dining Bar	0.16	Cafe/Sweets	0.13
Location	2	3.91	Izakaya	0.33	Bar/Cocktail	0.33	Dining Bar	0.17
Location	3	3.87	Cafe/Sweets	0.24	Izakaya	0.2	Bar/Cocktail	0.18
Location	4	5.65	Cafe/Sweets	0.19	Izakaya	0.17	Dining Bar	0.16
Location	5	3.91	Italian/French	0.26	Izakaya	0.25	Bar/Cocktail	0.15
Location	6	3.41	Cafe/Sweets	0.2	Izakaya	0.2	Italian/French	0.15
Location	7	3.68	Dining Bar	0.27	Izakaya	0.25	Cafe/Sweets	0.2
Location	8	5.33	Cafe/Sweets	0.3	Izakaya	0.27	Dining Bar	0.13
Reservations	1	5.25	Izakaya	0.5	Italian/French	0.17	Yakiniku/Kore	0.11
Reservations	2	3.73	Izakaya	0.28	Italian/French	0.25	Japanese	0.18
Reservations	3	3.47	Izakaya	0.38	Italian/French	0.18	Dining Bar	0.16
Reservations	4	3.87	Cafe/Sweets	0.32	Izakaya	0.15	Bar/Cocktail	0.13
Reservations	5	3.85	Izakaya	0.41	Dining Bar	0.15	Italian/French	0.13
Volume	1	5.41	Izakaya	0.23	Cafe/Sweets	0.21	Italian/French	0.15
Volume	2	3.36	Izakaya	0.23	Italian/French	0.23	Cafe/Sweets	0.21
Volume	3	3.75	Izakaya	0.26	Cafe/Sweets	0.21	Dining Bar	0.16
Volume	4	3.88	Izakaya	0.34	Cafe/Sweets	0.28	Italian/French	0.11
Volume	5	3.23	Cafe/Sweets	0.31	Izakaya	0.23	Italian/French	0.2
Volume	6	5.63	Bar/Cocktail	0.2	Cafe/Sweets	0.19	Izakaya	0.17

		Num.Tourist.Attr.Nearby	Median.Daily.Reserv	Median.Res.Vis	Total.Visitors.Day	Num.Subway.Stops.Nearby	Nightlife.Nearby	Median.Num.Hol.w.Visits
All	**All**	**3**	**0**	**0**	**17**	**1**	**7**	**16**
Availability	1	3	0	0	16	1	5	14
Availability	2	3	0	0	19	1	7	18
Availability	3	3	0	0	18	1	7	24
Availability	4	3	1	4	29	1	7	2
Availability	5	3	0	0	16	1	7	12
Availability	6	4	0	0	15	1	7	7
Location	1	2	1	4	14	0	19	17
Location	2	0	0	0	11	1	20	17
Location	3	4	0	0	16	1	3	16
Location	4	6	0	0	17	1	20	17
Location	5	12	0	0	17	3	3	13
Location	6	2	0	0	19	2	7	13
Location	7	3	1	3	20	0	7	17
Location	8	1	0	0	19	0	1	16
Reservations	1	3	4	22	32	0	7	20
Reservations	2	4	3	12	23	1	7	16
Reservations	3	3	1	5	16	1	7	16
Reservations	4	3	0	0	16	1	7	15
Reservations	5	3	2	7	21	0	4	18
Volume	1	3	0	0	15	1	7	16
Volume	2	3	0	0	33	1	3	9
Volume	3	3	0	0	22	1	5	17
Volume	4	3	0	0	31	1	6	17
Volume	5	3	0	0	43	1	5	14
Volume	6	3	0	0	8	1	7	14

APPENDIX B- COMPLETE DATA EXPLORATION

The RH data span three main categories: reservations made via the AirREGI system, reservation made via the HPG system, and store specific data like visits and store location. We looked at each category in turn.

AIRREGI RESTAURANT DATA

The AirREGI restaurants are labeled with the restaurant genre, the geographic area, and the latitude and longitude. There are 14 distinct restaurant genre categories (Figure 19), with extremely unequal distribution. The most common genres are Izayaka and cafe/sweets.

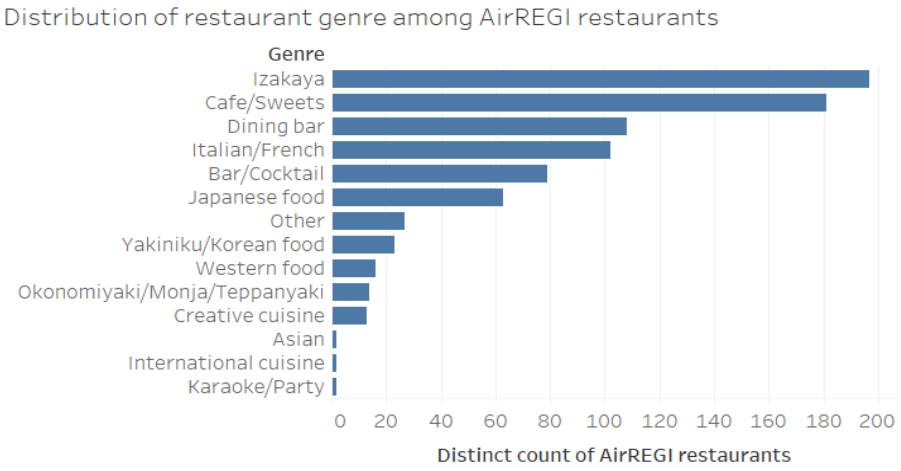

Figure 19 - Restaurant genres

-

Several restaurants share the same latitude and longitude pairs as a result of intentional obfuscation of individual restaurant locations and identities, resulting in 108 unique coordinate pairs among the 829 restaurants. Restaurants are located across Japan (Figure 20). The area variable demonstrates high cardinality with 103 distinct geographic areas.

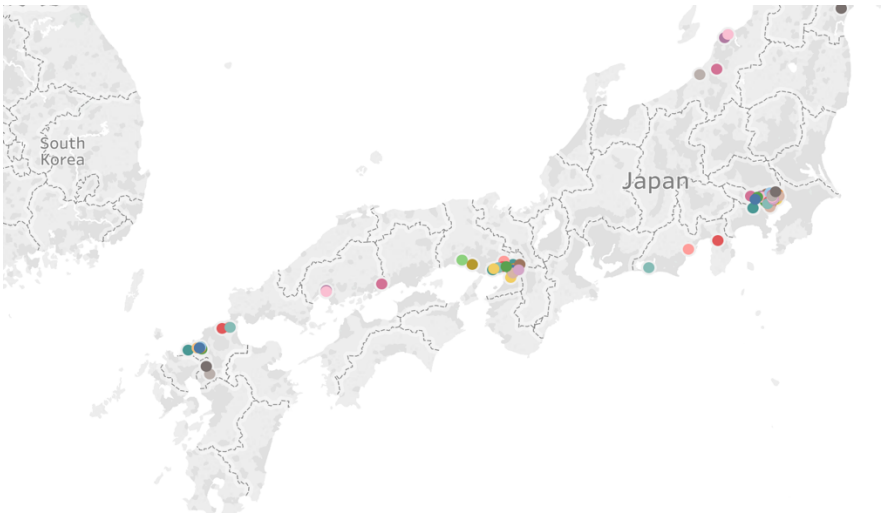

Figure 20 - Map of AirREGI restaurant locations

The distribution of the prediction target, visits per day per restaurant per day, is centered around 20 visits, shown in orange. (Figure 21). There are a few rare outlier days with more than 100 visitors per restaurant. It is difficult to explore outlier daily visits in the absence of hourly visits and restaurant capacity data.

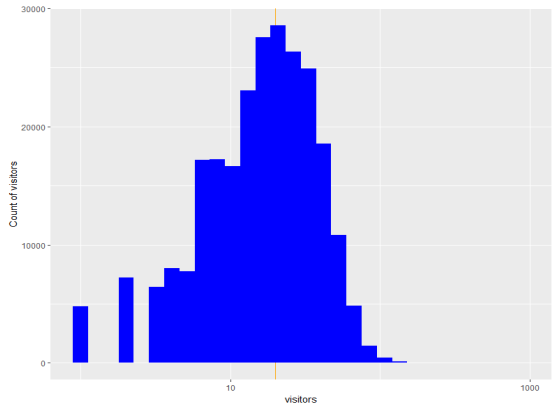

Figure 21 - Distribution of visitors per day

While a weekend holiday has little impact on the visitor numbers, and even decreases them slightly, there is a much more pronounced effect for the weekdays, especially Monday and Tuesday (Figure 22).

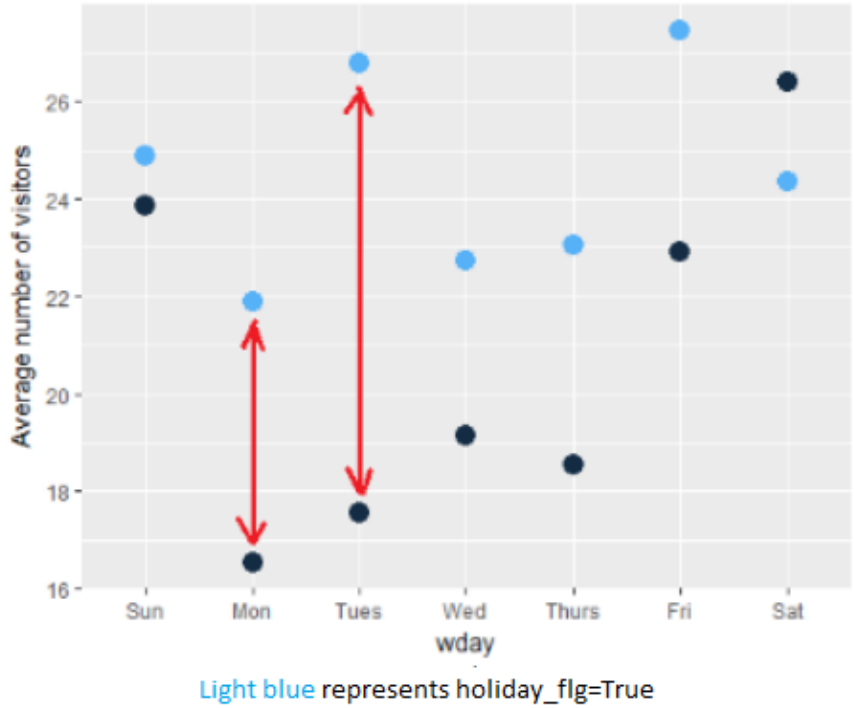

Figure 22 - Impact of holidays on visitor count

Historical visitor totals across all restaurants are presented in Figure 23. There is a periodic pattern that most likely corresponds to a weekly cycle. An outlier date with very few visits is noted on New Year's Day, likely due to a large number of restaurants closed for the holiday. Notice that there is an increase in volume starting July 2016. The plot also demonstrates seasonality and a slight upward trend, with more visits in December and in the most recent months of the data.

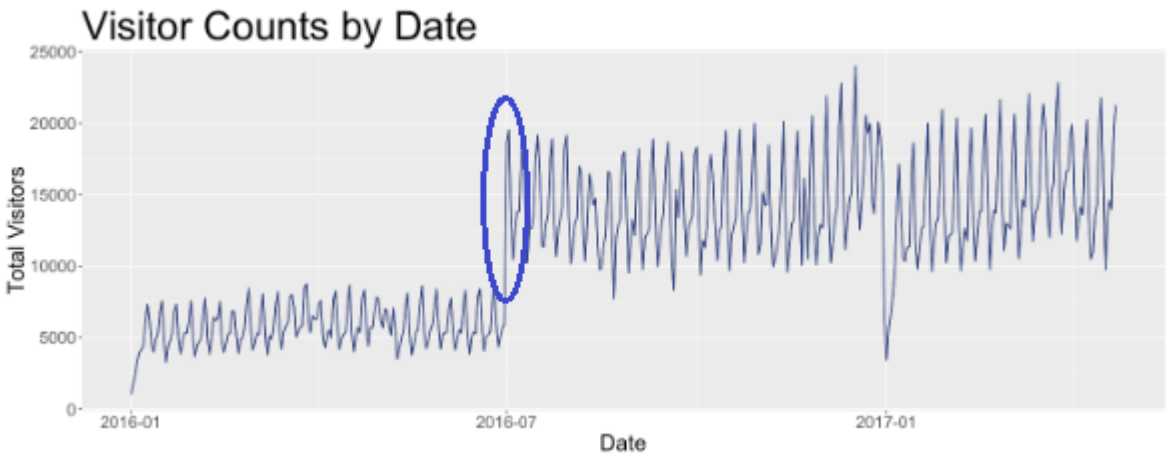

Figure 23 - Plot of Total Number of Visitors Per Date

The sudden increase in visitors in July 2016 is explained by the addition of approximately 400 restaurants to the data. In the upper panel of Figure 24, the dark blue trend shows the number of restaurants with visitor counts captured by date. The grey trend shows the number of restaurants with no visitor count captured by date. The sudden increase in restaurants with visitor count may be explained by the onboarding of new restaurants or increased data capture. Most of the restaurants have visitor counts in the last 10 months of the training data.

The variation in visitor count by date also demonstrates a periodic weekly cycle. Furthermore, there are dips in the number of restaurants with visitor counts associated with holidays, indicated by the holiday flags in the lower panel of Figure 24. This suggests the periodic absence of visitor counts may be due to restaurant closures.

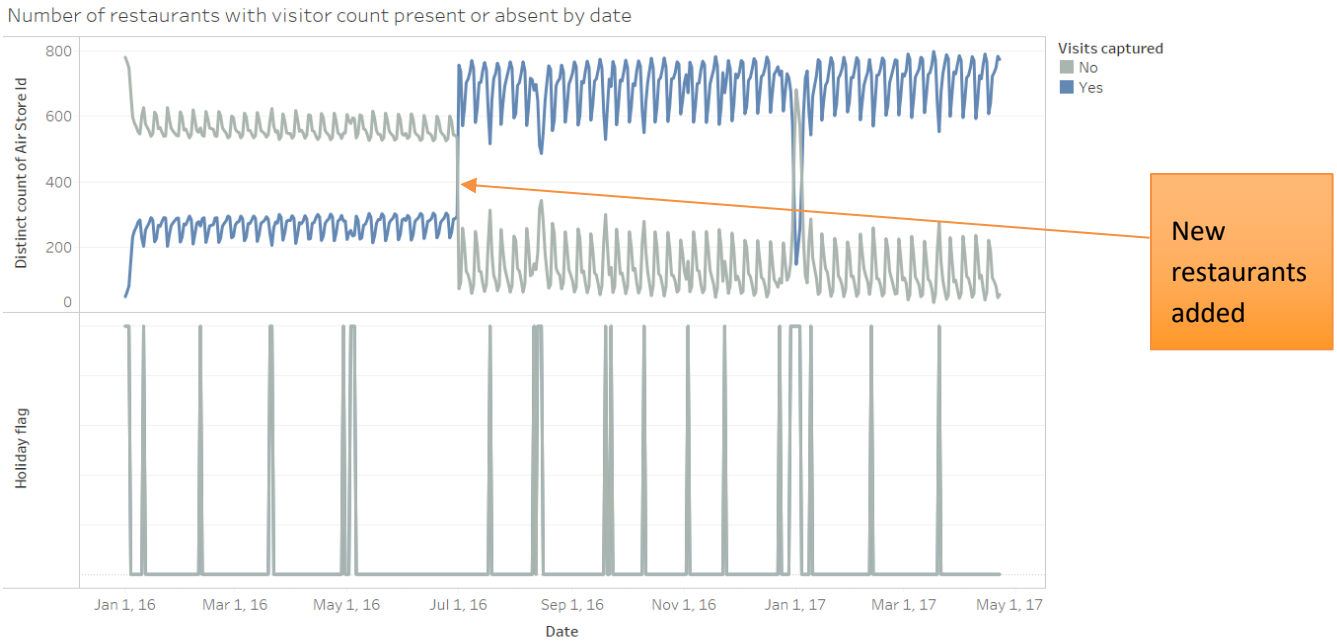

Figure 24 - Number of restaurants with visitor count captured by date, with holiday flags

Upon further investigation, individual restaurants had data for 4 days per week on average, with a bimodal distribution of restaurants with 3-4 days per week and restaurants with 5-6 days per week (Figure 25). This pattern was similar across restaurant genres and prefectures (not shown).

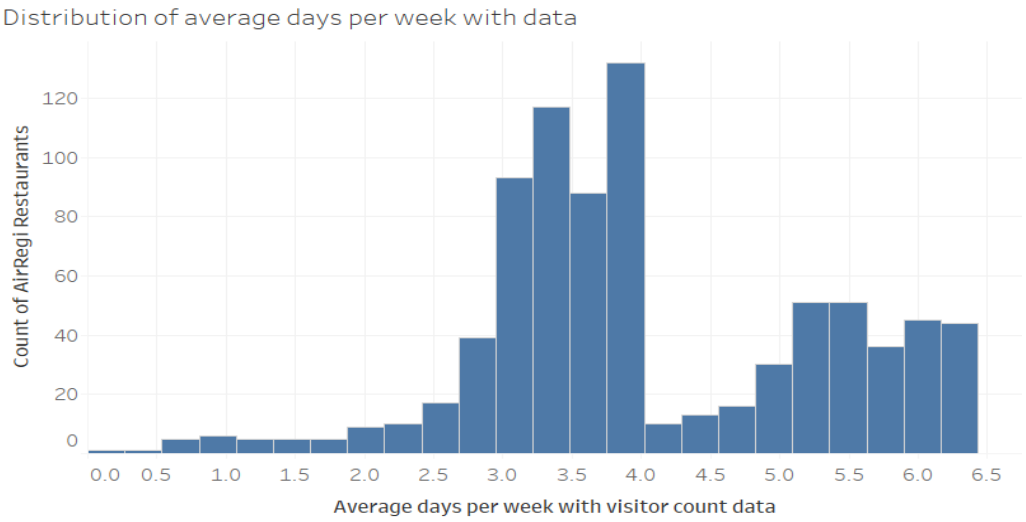

Figure 25 - Restaurant average days per week with visitors

These observations will be taken into account when imputing missing historical visitor counts for individual restaurant-date combinations prior to modeling. The majority of missing dates are likely the result of weekly planned restaurant closures as part of their normal operating schedule.

Holidays account for both missing visitor counts due to restaurant closures and impacts to visitor counts due to changes in diner patterns. The holiday period known as Golden Week: April 29 - May 5 is an important example. The plot in Figure 26 shows visitor counts around this date range as well as a smoothing fit in blue. Notice the blue line trending downward around the end of the first quarter in 2016. This is the negative impact of Golden Week on the restaurant business. Holiday information can be used effectively for modeling special cases.

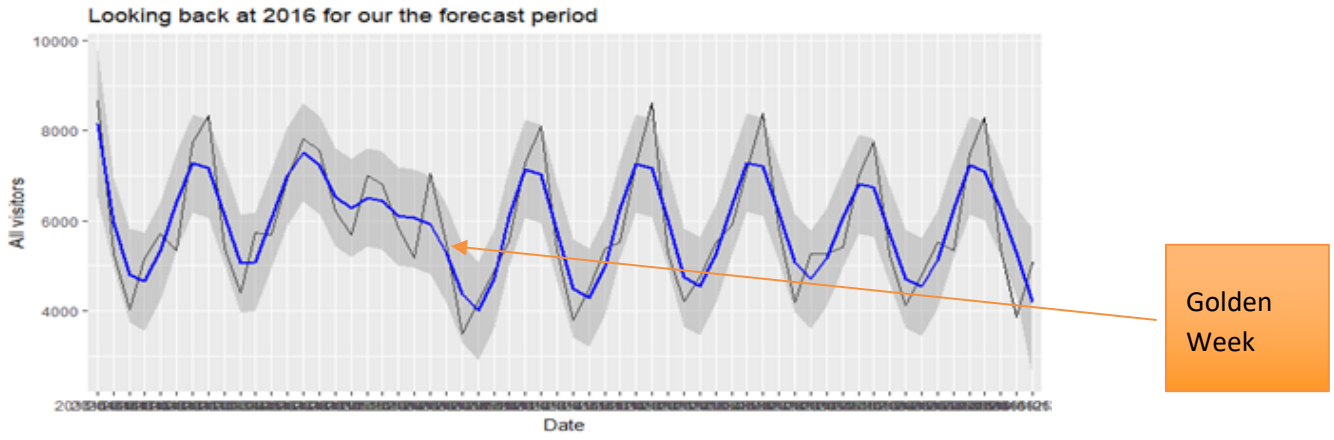

Figure 26 - Impact of Golden Week

The remaining unexplained missing dates are attributed to abnormal restaurant closure, failure to capture visit counts within AirREGI for the date, dates prior to the restaurant's use of AirREGI, or other technical issues leading to missing data.

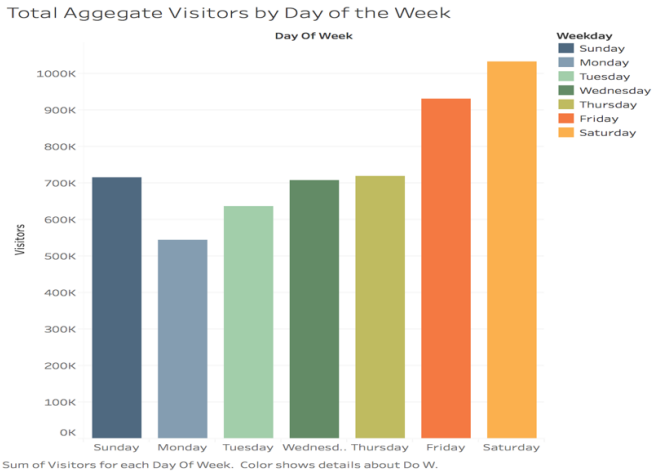

Figure 27 - Total Visitors by the Day of the Week

Drilling down on the periodicity of restaurant visits, a weekly pattern emerges that is intuitive for the restaurant sector: the highest volume is on Saturdays, followed by Fridays, and the lowest volume is on Monday (Figure 27). Visits also vary by month, with the highest volume in March and the lowest volume in May and June.

An important consideration for modeling time series data is stationarity. The training series was tested for stationarity using the Augmented Dickey-Fuller test and confirmed to be stationary. This indicates the data can be modeled using time series forecasting-class models.

In addition to general patterns for all restaurants, individual restaurants demonstrated widely varying patterns of visit volume. Figures 28 and 29 show the time series and autocorrelation of two restaurants. The plot of visitor count against date shows a unique pattern for each. The autocorrelation patterns shown below each time series plot demonstrate the periodicity of the data. Restaurant 101 has a random pattern of visit counts, while restaurant 201 has a clear weekly pattern of visit counts. Predictive models should account for individual restaurant characteristics, and RH will benefit from analysis of both general patterns and specific restaurant patterns in tailoring its AirREGI feature offerings.

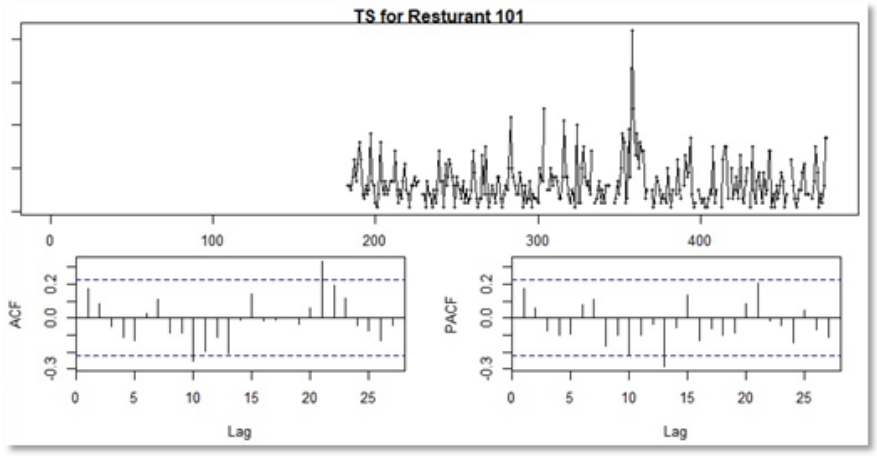

Figure 28 - Restaurant 101

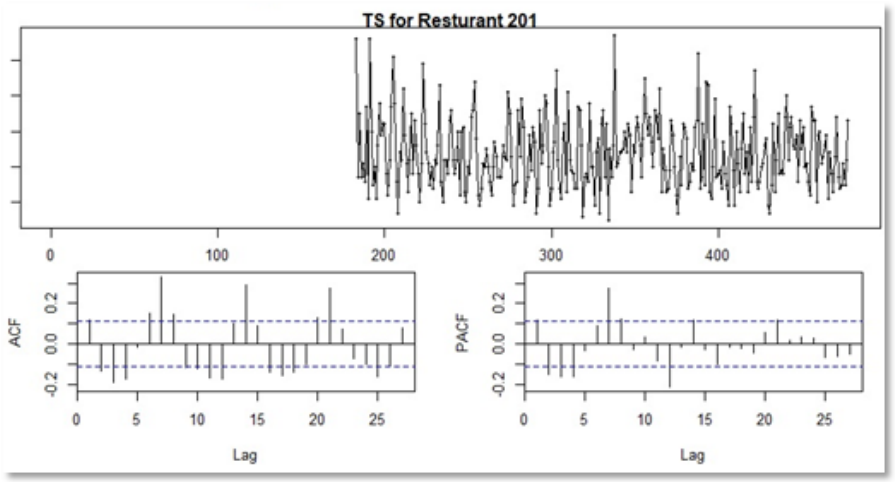

Figure 29 - Restaurant 201

RESERVATIONS DATA FROM AIRREGI AND HPG SYSTEMS

RH provided reservation data from the AirREGI POS system as well as the HPG mobile app. Figure 30 shows the number of visitors reserved through each system over time. Fewer reservations were made in 2016 through the AirREGI system and none at all for a long stretch of time between July 2016 and early October 2016. The volume only increased during the end of that year. Reserved visits were more consistent in 2017. The artificial decline we see after the first quarter of 2017 is most likely related to these reservations being at the end of the training time frame, which means that long-term reservations would not be part of this data set. For HPG reservations, reserved visits follow a more consistent pattern, with a clear spike in December 2016.

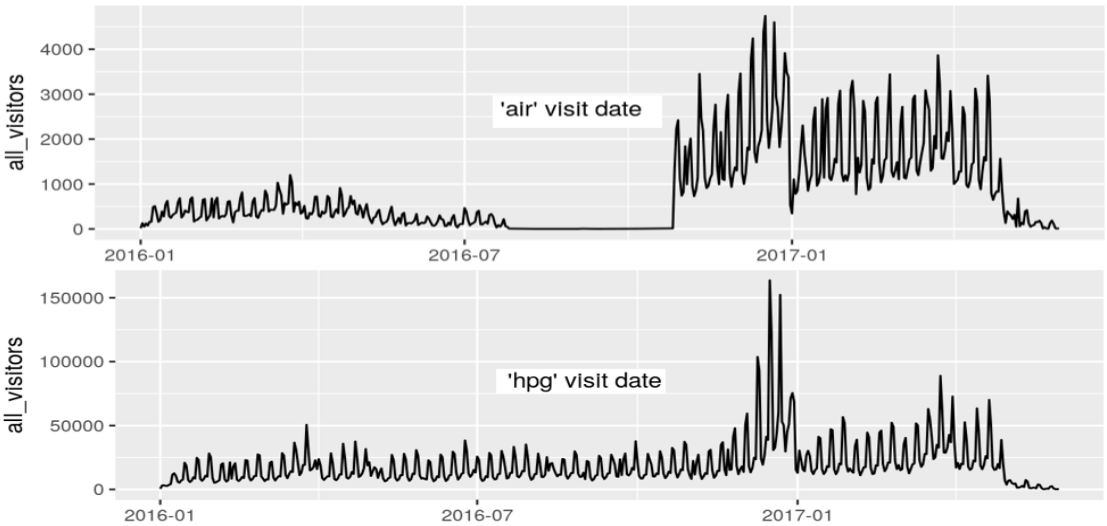

Figure 30 - Time Distribution of Reservation Data

ADDITIONAL FINDINGS

Outliers. The view of total visitors per month (Figure 31) shows evidence of outliers in red, which may impact the models.

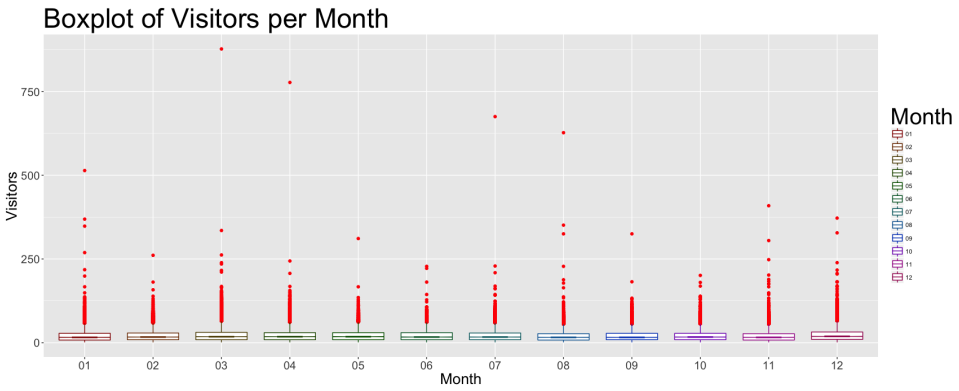

Figure 31 - Boxplot of Total Visitors Showing Outliers by Month

There was a significant outlier in the reservation data. A single restaurant had an abnormally high number of reservations for a single date. Upon examination, the reservations were made from February 2016 up until 5 days prior to the visit date. The source of this anomaly is unclear.

Number of Reservations	Number of Visitors	Earliest Reserved Date	Latest Reserved Date
362	2,241	2/4/16	11/5/16

Table 7 - Reservations for AirREGI store air_a17f0778617c76e2 for 11/10/2016

Potential Clusters. The visitors by city and genre view of the data in Figure 32 reveals there are natural clusters in the data.

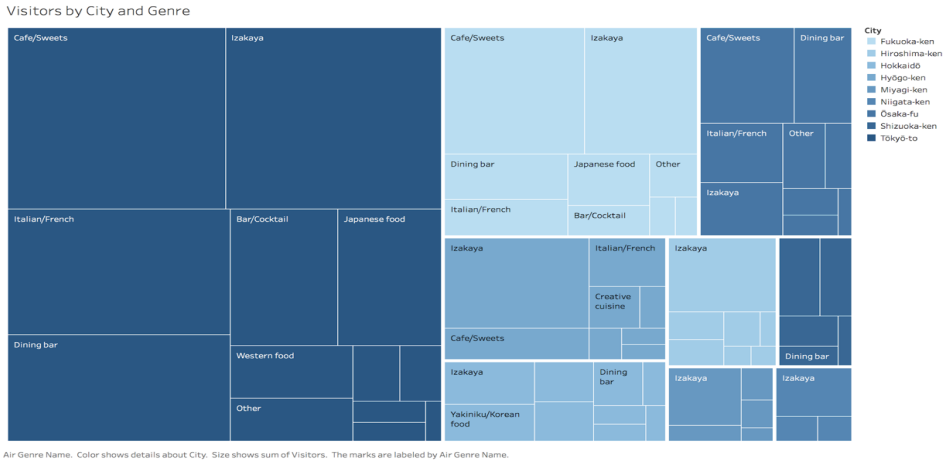

Figure 32 - Potential Clusters, Visits Aggregated by Prefecture and Genre. Size = Visits

Prefectures. The first portion of the the high-cardinality restaurant area name text variable was common across many restaurants. The text was parsed resulting in 9 distinct regions. These labels and the coordinates were referenced against Japanese geographic names and determined to be prefectures, the Japanese equivalent of counties. As shown in Figure 33, the majority of AirREGI restaurants fall in the Tokyo Prefecture. Prefectures will be used as a lower-cardinality descriptor of restaurant location for clustering, modeling and dashboards.

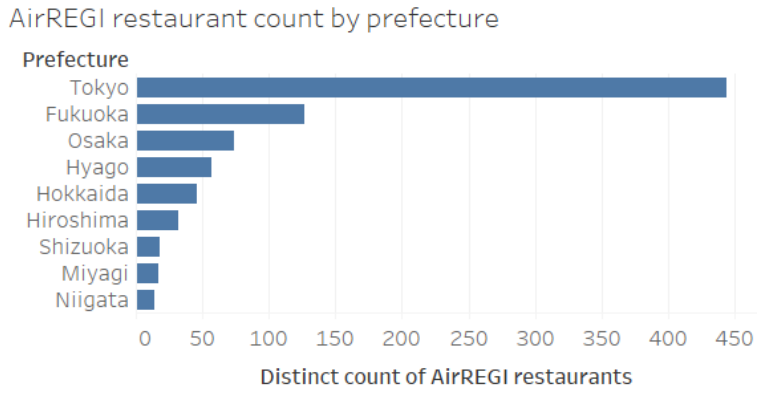

Figure 33 - Restaurant prefectures

APPENDIX C – LIST OF MODELS BUILT

Submission Name	Private Score	Public Score
TW_xgb_plain_no_class_rnd.csv	0.591	0.571
SR_TW_ensemble_top2.csv	0.799	0.862
TW_xgb_plain_no_class_tst.csv	0.635	0.611
TW_xgb_plain_holiday_shorter4.csv	0.591	0.57
TW_glm_gauss.csv	**0.876**	**0.878**
TW_pois2B.csv	0.876	0.878
TW_xgb_plain_holiday_shorter3.csv	0.777	0.836
TW_xgb_plain_holiday_shorter2.csv	0.688	0.699
TW_xgb_plain_holiday_shorter.csv	0.688	0.72
TW_xgb_plain_holiday.csv	0.687	0.707
TW_xgb_plus_holiday4.csv	1.649	1.529
TW_xgb_plus_holiday2.csv	1.87	1.875
TW_xgb_plus_holiday.csv	2.066	2.202
TW_full4_ensemble.csv	0.996	1.041
TW_xgb_full_meal_deal4.csv	1.93	1.934
SR_TW_ensemble.csv	0.798	0.809
TW_xgb_full_meal_deal3.csv	2.236	2.37
TW_xgb_weather_clusters3.csv	0.709	0.704
TW_xgb_full_meal_deal.csv	2.394	2.458
TW_xgb_full_meal_deal.csv	1.008	1.613
TW_xgb_1_by_1.csv	0.884	0.877
TW_xgb_kaggle_data_only.csv	**0.834**	**0.852**
TW_xgb_weather_clusters3.csv	0.709	0.704
TW_xgb_weather_clusters2.csv	0.714	0.704
TW_xgb_weather_clusters1.csv	0.789	0.784
TW_xgb_s_weather_1.csv	0.783	0.768
TW_xgb_s_weather_1.csv	0.887	0.879
TW_ts_xgb_lag1.csv	1.295	1.016
TW_ts_xgb_lag1e7.csv	1.08	0.971
TW_xgb_weather_m1.csv	**0.752**	**0.734**
TW_metro_xgb2.csv	0.798	0.807
TW_metro_xgb.csv	0.86	0.876

TW_xgb_wthr_rnd.csv	0.753	0.735
TW_xgb_ts_1.csv	0.874	0.848

Submission Name	Private Score	Public Score
SR_ScoreFile_hybridForecastModel.csv	**0.593**	**0.573**
SR_ScoreFile_neuralnet_a.csv	0.620	0.576
SR_ScoreFile_AA.csv	**0.613**	**0.587**
SR_ScoreFile_autoARIMA_b.csv	0.621	0.592
SR_ScoreFile_ETS.csv	0.616	0.601
SR_ScoreFile_autoARIMA.csv	0.722	0.684
SR_ScoreFile_AA_xreg_b.csv	1.222	1.172

APPENDIX D - R CODE FOR TOP MODEL AND CLUSTERING

EDA, other models, and clustering(?) code produced for the project will be included in the project zip file. The top model code is shown below:

BEST MODEL CODE

```
################################################################################
# NWU Capstone 498
#
# Table for Four Project team
# Advisor: Don Wedding
#
# Team: Tom Alig, Sheela Rao, Catherine Tolley, Tamara Williams
#
# Script Author: Tamara Williams
#
# This script uses an input file which contains the base Kaggle data, joined to the
# weather data from Hunter McGushion's kernal
# (https://www.kaggle.com/huntermcgushion/exhaustive-weather-eda-file-overview)
#
# The data are then additionally augmented by adding cluster info and the is.golden.week
# flag feature
################################################################################

# For the sake of good programming hygiene, start with a clean workspace
#------------------------------------------------------------------
# clear Workspace, then clear console
rm(list=ls())
cat("\014")

# Get location script, and set to working directory
#------------------------------------------------------------------
working.dir = dirname(rstudioapi::getSourceEditorContext()$path)
setwd(working.dir)
```

```
data.path = # your path
out.path = # your path
train.file = "final_train_dataset.csv"
test.file = "final_test_dataset.csv"

# include required packages
#----------------------------------------------------------------
library(reshape2)
library(dplyr)
library(zoo)
library(magrittr)
library(xgboost)
library(lubridate)

#############################################################################
######              Read and prep data block          ##########
#############################################################################
# get the data
df=read.csv(paste0(data.path,train.file),header=T,stringsAsFactors = T)
test.df = read.csv(paste0(data.path,test.file), header = T, stringsAsFactors = T)

# get Kaggle competition file to extract store ids you do NOT submit
submit_only_id=read.csv(paste0(data.path,"sample_submission 3.csv"),header=T, stringsAsFactors = FALSE)

df[is.na(df)] = 0
test.df[is.na(test.df)] = 0

# move visitors to position 1 for easy of processing
refcols  =  c("visitors", "is.golden.week","air_store_id", "visit_date","clus.avail","clus.loc",
        "clus.res","clus.vol", "holiday.name","is.weekend" )
df = df[, c(refcols, setdiff(names(df), refcols))]

refcols  =  c("is.golden.week","air_store_id", "visit_date","clus.avail","clus.loc",
```

```
            "clus.res","clus.vol", "holiday.name","is.weekend")
test.df = test.df[, c(refcols, setdiff(names(test.df), refcols))]

df$visit_date = as.Date(df$visit_date)
test.df$visit_date = as.Date(test.df$visit_date)

# dropping columns which are not useful
df = df[-c(13,14)]
test.df = test.df[-c(12,13)]

df.overlap = df[df$visit_date >= "2017-04-12",]
df.minus = df[df$visit_date < "2017-04-12",]
tmp = df.overlap[-c(1)]
df.chart.it = rbind(tmp, test.df)

#================================================================================
# Function to output the forecast in the right format for Kaggle
#================================================================================
make_submission <- function (my_forecast,m_name){
 print("entered function")
 #  my_forecast <- pred_df

 new_sub_id = paste0(test.df$air_store_id,"_", test.df$visit_date)
 submission = cbind(new_sub_id, my_forecast)

 names(submission) = c('id', 'visitors')
 submission$visitors = pmax(submission$visitors,0)
 submission$visitors = round(submission$visitors,0)

 # get rid of the id's NOT in the submission file
 print("starting the extraction of good ids")
 good_id = submit_only_id$id
 all_id = submission$id
```

```
 submission = submission[submission$id %in% good_id, ]

 #getNow <- as.integer(now())
 submissionFile = paste0(out.path,"TW_", m_name, ".csv")
 write.csv(submission, submissionFile, quote=FALSE, row.names = FALSE)
}

make_test <- function (my_forecast,m_name){
 print("entered function")
 #  my_forecast <- pred_df

 new_sub_id = test.df$air_store_id
 new_date = test.df$visit_date
 submission = cbind(new_sub_id, new_date, my_forecast)

 names(submission) = c('id', 'visit_date', 'visitors')
 submission$visitors = pmax(submission$visitors,0)
 #  submission$visitors = round(submission$visitors,0)

 # get rid of the id's NOT in the submission file
 print("starting the extraction of good ids")
 good_id = submit_only_id$id
 all_id = submission$id
 submission = submission[submission$id %in% good_id, ]

 submissionFile = paste0(out.path,"TW_", m_name, ".csv")
 write.csv(submission, submissionFile, quote=FALSE, row.names = FALSE)
}

#============================================================================
#  Setup and train the model
#============================================================================
set.seed(13)
```

```
df_m = df[,-c(1)]
df_m = data.matrix(df_m, rownames.force = F)

test_m = data.matrix(test.df, rownames.force = F)

target=(df$visitors)

#train model
#-----------------------------------------------------------------
bst_df = xgboost(data = df_m,
                 label = target,
                 eta = 0.3,
                 max_depth = 15,
                 nrounds = 35,
                 nfold = 10,
                 subsample = 0.7,
                 colsample_bytree = 0.6,
                 gamma = .2,
                 min_child = 1,
                 eval_metric = 'rmse',
                 objective = "reg:linear"
)

my.pred =  predict(bst_df, test_m)
pred_df = as.data.frame(my.pred)

make_submission(pred_df, "your-file-name")

# make the charting data so you have a 10-day overlap to show
# actual versus predicted.
#
# Do the actual combining of predicted and actual
```

```
# data in Excel, it's faster.


test_m2 = data.matrix(df.chart.it, rownames.force = F)

chart.pred =  predict(bst_df, test_m2)

tmp = df.chart.it[c('air_store_id', 'visit_date')]

colnames(tmp) = c()

make.chart.data = cbind(tmp, chart.pred)

write.csv(make.chart.data, paste0(data.path, "predicts_for_chart.csv"), quote=FALSE, row.names = FALSE)
```

CLUSTERING CODE

```
# Predict 498, Capstone
# Clustering/Segmenting
# May 27, 2018

library(cluster)
library(ggplot2)
library(useful)
library(Hmisc)
library(HSAUR)
library(MVA)
library(HSAUR2)
library(fpc)
library(mclust)
library(lattice)
library(car)
library(tidyverse)
library(plot3D)
library(maptree)
library(proxy)
library(corrplot)
library(reshape)
library(NbClust)
library(devtools)
library(factoextra)
library(clues)
library(dplyr)

# read in data
rest.info <- read.csv ("restaurant_info.csv")

############################################################
# Standardize the variables with mean of 0 and std dev of 1
```

```
##########################################################

## Availability

head(rest.info$dates.with.visits)
length(rest.info$dates.with.visits)
rest.info$scale.dates.w.visits <- scale(rest.info$dates.with.visits)
head(rest.info$scale.dates.with.visits)

rest.info$scale.hol.w.visits <- scale(rest.info$holidays.with.visits)
rest.info$scale.meandays.perwk <- scale(rest.info$mean.days.per.week)
rest.info$scale.num.wks.open <- scale(rest.info$number.weeks.open)

head(rest.info)

subset.names.avail <- subset(rest.info, select = c("scale.dates.w.visits",
                                "scale.hol.w.visits",
                                "scale.meandays.perwk",
                                "scale.num.wks.open"))

wssplot.avail <- function(subset.names.avail, nc=15, seed=1234) {
 wss <- (nrow(subset.names.avail)-1)*sum(apply(subset.names.avail,2,var))
 for (i in 2:nc) {
  set.seed(seed)
  wss[i] <- sum(kmeans(subset.names.avail, centers=i)$withinss)}
 plot(1:nc, wss, type="b", xlab="Number of Clusters",
    ylab="Within groups sum of squares")}

wssplot.avail(subset.names.avail)

####

# KMeans w Six Clusters

####

##########################################################
### Create a Kmeans with 6 clusters with derived data #
##########################################################
clusterresults.k6.avail <- kmeans(subset.names.avail,6)
names(clusterresults.k6.avail)
rsquare.k6.avail <- clusterresults.k6.avail$betweenss/clusterresults.k6.avail$totss
rsquare.k6.avail
avail.cluster.plot <- plot(clusterresults.k6.avail, data=subset.names.avail,
   main = "K Means 6 Clusters, Availability Data")
avail.cluster.plot
dev.print(png, 'Availability.Clusters.png', width=480, height=360)

class(subset.names.avail)
str(subset.names.avail)
head(subset.names.avail)
newdf.k6.avail <- cbind(rest.info,clusterresults.k6.avail$cluster)
```

```
head(newdf.k6.avail)
newdf.k6.availb <- newdf.k6.avail

# Re-name last column
names(newdf.k6.availb)[names(newdf.k6.availb) == 'clusterresults.k6.avail$cluster'] <- 'k6clust'
head(newdf.k6.availb)

tablek6 <- table(newdf.k6.availb$k6clust)
tablek6

k61 <- tablek6[1] # size of Cluster 1
k62 <- tablek6[2]
k63 <- tablek6[3]
k64 <- tablek6[4]
k65 <- tablek6[5] # size of Cluster 5
k66 <- tablek6[6]

####################################

# LOCATION

####################################
rest.info$scale.subwayStops <- scale(rest.info$subwayStops)
head(rest.info$scale.subwayStops)

rest.info$scale.touristAct <- scale(rest.info$touristAct)
rest.info$scale.nightLife <- scale(rest.info$nightLife)
head(rest.info)

subset.names.loc <- subset(rest.info, select = c("scale.subwayStops",
                                    "scale.touristAct",
                                    "scale.nightLife"))

wssplot.loc <- function(subset.names.loc, nc=15, seed=1234) {
 wss <- (nrow(subset.names.loc)-1)*sum(apply(subset.names.loc,2,var))
 for (i in 2:nc) {
  set.seed(seed)
  wss[i] <- sum(kmeans(subset.names.loc, centers=i)$withinss)}
 plot(1:nc, wss, type="b", xlab="Number of Clusters",
    ylab="Within groups sum of squares")}

wssplot.loc(subset.names.loc)




#########################################################
### Create a Kmeans with 8 clusters for LOCATION
#########################################################
clusterresults.k8.loc <- kmeans(subset.names.loc,8)
names(clusterresults.k8.loc)
clusterresults.k8.loc
```

```
rsquare.k8.loc <- clusterresults.k8.loc$betweenss/clusterresults.k8.loc$totss
rsquare.k8.loc

loc.cluster.plot <- plot(clusterresults.k8.loc, data=subset.names.loc,
   main = "K Means 8 Clusters, location Data")

loc.cluster.plot
dev.print(png, 'Location.Clusters.png', width=480, height=360)

class(subset.names.loc)
str(subset.names.loc)
head(subset.names.loc)
dim(subset.names.loc)

newdf.k8.loc <- cbind(rest.info,clusterresults.k8.loc$cluster)
head(newdf.k8.loc)
newdf.k8.locb <- newdf.k8.loc

# Re-name last column
names(newdf.k8.locb)[names(newdf.k8.locb) == 'clusterresults.k8.loc$cluster'] <- 'k8clust'
head(newdf.k8.locb)

tablek8 <- table(newdf.k8.locb$k8clust)
tablek8

k81 <- tablek8[1] # size of Cluster 1
k82 <- tablek8[2]
k83 <- tablek8[3]
k84 <- tablek8[4]
k85 <- tablek8[5] # size of Cluster 5
k86 <- tablek8[6]
k87 <- tablek8[7]
k88 <- tablek8[8]

####################################

# RESERVATIONS Standardize

####################################

rest.info$reservations <- (rest.info$dates.with.res +
                    rest.info$median.daily.reservations +
                    rest.info$median.daily.res.visitors +
                    rest.info$median.res.lead.hours +
                    rest.info$air.reservations +
                    rest.info$hpg.reservations)/ 6

rest.info$scale.dates.with.res <- scale(rest.info$dates.with.res)
head(rest.info$scale.dates.with.res)

rest.info$scale.median.daily.reservations <- scale(rest.info$median.daily.reservations)
rest.info$scale.median.daily.res.visitors <- scale(rest.info$median.daily.res.visitors)
```

```
rest.info$scale.median.res.lead.hours <- scale(rest.info$median.res.lead.hours)
rest.info$scale.air.reservations <- scale(rest.info$air.reservations)
rest.info$scale.hpg.reservations <- scale(rest.info$hpg.reservations)

head(rest.info)

subset.names.res <- subset(rest.info, select = c("scale.dates.with.res",
                                  "scale.median.daily.reservations",
                                  "scale.median.daily.res.visitors",
                                  "scale.median.res.lead.hours",
                                  "scale.air.reservations",
                                 "scale.hpg.reservations"))

wssplot.res <- function(subset.names.res, nc=15, seed=1234) {
 wss <- (nrow(subset.names.res)-1)*sum(apply(subset.names.res,2,var))
 for (i in 2:nc) {
  set.seed(seed)
  wss[i] <- sum(kmeans(subset.names.res, centers=i)$withinss)}
 plot(1:nc, wss, type="b", xlab="Number of Clusters",
    ylab="Within groups sum of squares")}

wssplot.res(subset.names.res)


########################################################
### Create a Kmeans with 5 clusters for RESERVATION
########################################################
clusterresults.k5.res <- kmeans(subset.names.res,5)
names(clusterresults.k5.res)
rsquare.k5.res <- clusterresults.k5.res$betweenss/clusterresults.k5.res$totss
rsquare.k5.res

res.cluster.plot <- plot(clusterresults.k5.res, data=subset.names.res,
   main = "K Means 7 Clusters, Reservation Data")

res.cluster.plot
dev.print(png, 'Reservation.Clusters.png', width=480, height=360)


class(subset.names.res)
str(subset.names.res)
head(subset.names.res)

newdf.k5.res <- cbind(rest.info,clusterresults.k5.res$cluster)
head(newdf.k5.res)
newdf.k5.resb <- newdf.k5.res

# Re-name last column
names(newdf.k5.resb)[names(newdf.k5.resb) == 'clusterresults.k5.res$cluster'] <- 'k5clust'
head(newdf.k5.resb)
dim(newdf.k5.resb)
```

```
tablek5.res <- table(newdf.k5.resb$k5clust)
tablek5.res

k51 <- tablek5.res[1] # size of Cluster 1
k52 <- tablek5.res[2]
k53 <- tablek5.res[3]
k54 <- tablek5.res[4]
k55 <- tablek5.res[5] # size of Cluster 5

points.in.clus.1 <- newdf.k5.resb[which(newdf.k5.resb$k5clust == '1'),]
points.in.clus.1 # 50, 528, 750
# 50 has median reserv visitors of 52, BUT only 1 date with reservations
#   reservation was made 49 days in advance
#   Had 292 dates w visitors

# 528 has 50 dates w reserv, but only 47 dates w visitors
#   Has 56 day median reservation lead time
#   Has 4791 air reservations

# 750 had only 1 date w reservations, for 11 people, made 39 days in advance
#   Had 421 dates w visitors

points.in.clus.5 <- newdf.k5.resb[which(newdf.k5.resb$k5clust == '5'),]
points.in.clus.5 # 753, has MEDIAN reservation lead time of 131 DAYS!

dim(newdf.k5.resb)
newdf.k5.resb[750:753, 38:44]

##############################

## VOLUME

##############################
rest.info$volume <- (rest.info$median.daily.visitors +
                rest.info$busy.DOW.mean.visits +
                rest.info$slow.DOW.mean.visits)/ 3

head(rest.info$median.daily.visitors)
length(rest.info$median.daily.visitors)
rest.info$scale.median.daily.visitors <- scale(rest.info$median.daily.visitors)
head(rest.info$scale.median.daily.visitors)

rest.info$scale.busy.DOW.mean.visits <- scale(rest.info$busy.DOW.mean.visits)
rest.info$scale.slow.DOW.mean.visits <- scale(rest.info$slow.DOW.mean.visits)

subset.names.vol <- subset(rest.info, select = c("scale.median.daily.visitors",
                                "scale.busy.DOW.mean.visits",
                                "scale.slow.DOW.mean.visits"))

wssplot.vol <- function(subset.names.vol, nc=15, seed=1234) {
 wss <- (nrow(subset.names.vol)-1)*sum(apply(subset.names.vol,2,var))
 for (i in 2:nc) {
```

```
  set.seed(seed)
  wss[i] <- sum(kmeans(subset.names.vol, centers=i)$withinss)}
 plot(1:nc, wss, type="b", xlab="Number of Clusters",
    ylab="Within groups sum of squares")}

wssplot.vol(subset.names.vol)


########################################################
### Create a Kmeans with 6 clusters with derived data #
########################################################
clusterresults.k6.vol <- kmeans(subset.names.vol,6)
clusterresults.k6.vol

names(clusterresults.k6.vol)
rsquare.k6.vol <- clusterresults.k6.vol$betweenss/clusterresults.k6.vol$totss
rsquare.k6.vol
vol.cluster.plot <- plot(clusterresults.k6.vol, data=subset.names.vol,
   main = "K Means 6 Clusters, volume Data")

vol.cluster.plot
dev.print(png, 'Volume.Clusters.png', width=480, height=360)

class(subset.names.vol)
str(subset.names.vol)
head(subset.names.vol)

newdf.k6.vol <- cbind(rest.info,clusterresults.k6.vol$cluster)
head(newdf.k6.vol)
newdf.k6.volb <- newdf.k6.vol

# Re-name last column
names(newdf.k6.volb)[names(newdf.k6.volb) == 'clusterresults.k6.vol$cluster'] <- 'k6clust'
head(newdf.k6.volb)

tablek6.vol <- table(newdf.k6.volb$k6clust)
tablek6.vol

# Cluster 5 has only five members, and they all seem to be outliers.
#   Let's investigate further
#   First, we'll find which restaurants are in Cluster 5

points.in.clus.5.vol <- newdf.k6.volb[which(newdf.k6.volb$k6clust == '5'),]
head(points.in.clus.5.vol)
# restaurants 72, 104, 454, 511, 722 are in cluster 5
# let's check them out
rest.info[71:73,]
rest.info[103:105,]
rest.info[453:455,]
rest.info[510:512,]
rest.info[721:723,]
```

```
# Cluster 5 closest to Cluster 4
#  Let's check out a little about Cluster 4
points.in.clus.4.vol <- newdf.k6.volb[which(newdf.k6.volb$k6clust == '4'),]
head(points.in.clus.4.vol)

# change factors to numeric in cluster 4 investigation
points.in.clus.4.vol.num <- lapply(points.in.clus.4.vol, as.numeric)
head(points.in.clus.4.vol.num)
max(points.in.clus.4.vol.num$scale.median.daily.visitors) # 3.09
max(points.in.clus.4.vol.num$scale.busy.DOW.mean.visits) # 3.65
max(points.in.clus.4.vol.num$scale.slow.DOW.mean.visits) # 3.40

# Impute the 3 scaled variables in Volume cluster
#  Goal is for the 5 restaurants in Cluster 5 to become part of Cluster 4
#  This will be done by imputing values for those five restaurants to be the
#  max values of Cluster 4 members. Will then re-run info to determine optimal
#  number of clusters

# First, copy existing variables into variable impute_x
head(subset.names.vol)
subset.names.vol$imp.scale.median.daily.visitors <- subset.names.vol$scale.median.daily.visitors
subset.names.vol$imp.scale.busy.DOW.mean.visits <- subset.names.vol$scale.busy.DOW.mean.visits
subset.names.vol$imp.scale.slow.DOW.mean.visits <- subset.names.vol$scale.slow.DOW.mean.visits

# Create new data frame w  only the imputed values
subset.names.vol.2 <- subset(subset.names.vol, select = c("imp.scale.median.daily.visitors",
                                  "imp.scale.busy.DOW.mean.visits",
                                  "imp.scale.slow.DOW.mean.visits"))

# Now set each of the outliers to the max of Cluster 4
subset.names.vol.2[c(72, 103, 454, 511, 722), 1] <- 3.09
subset.names.vol.2[c(72, 103, 454, 511, 722), 2] <- 3.65
subset.names.vol.2[c(72, 103, 454, 511, 722), 3] <- 3.40

# Now, how many clusters do we need? Check out skree plot
wssplot.vol.2 <- function(subset.names.vol.2, nc=15, seed=1234) {
 wss <- (nrow(subset.names.vol.2)-1)*sum(apply(subset.names.vol.2,2,var))
 for (i in 2:nc) {
  set.seed(seed)
  wss[i] <- sum(kmeans(subset.names.vol.2, centers=i)$withinss)}
 plot(1:nc, wss, type="b", xlab="Number of Clusters",
    ylab="Within groups sum of squares")}

wssplot.vol.2(subset.names.vol.2)

# It once again appears that 6 clusters is best. Let's investigate further

########################################################
### Create a Kmeans with 6 clusters with derived data #
########################################################
clusterresults.k6.vol.2 <- kmeans(subset.names.vol.2,6)
head(clusterresults.k6.vol.2)
```

```
head(clusterresults.k6.vol.2$cluster)
names(clusterresults.k6.vol.2)
rsquare.k6.vol.2 <- clusterresults.k6.vol.2$betweenss/clusterresults.k6.vol.2$totss
rsquare.k6.vol.2
plot(clusterresults.k6.vol.2, data=subset.names.vol.2,
   main = "K Means 6 Clusters, volume Data")

# So far, looks OK

newdf.k6.vol.2 <- cbind(rest.info,clusterresults.k6.vol.2$cluster)
head(newdf.k6.vol.2)
names(newdf.k6.vol.2)
newdf.k6.volb.2 <- newdf.k6.vol.2

# Re-name last column
names(newdf.k6.volb.2)[names(newdf.k6.volb.2) == 'clusterresults.k6.vol.2$cluster'] <- 'k6clust'
head(newdf.k6.volb.2)
dim(newdf.k6.volb.2)

tablek6.vol.2 <- table(newdf.k6.volb.2$k6clust)
tablek6.vol.2

k61.vol <- tablek6.vol.2[1] # size of Cluster 1
k62.vol <- tablek6.vol.2[2]
k63.vol <- tablek6.vol.2[3]
k64.vol <- tablek6.vol.2[4]
k65.vol <- tablek6.vol.2[5] # size of Cluster 5
k66.vol <- tablek6.vol.2[6]


###################################

# Data Frame of Store ID and Cluster

###################################
store.id.w.clust.avail <- data.frame(newdf.k6.availb$air_store_id, newdf.k6.availb$k6clust)
head(store.id.w.clust.avail)

colnames(store.id.w.clust.avail) <- c("air_store_id", "cluster.avail")
head(store.id.w.clust.avail)
write.csv(store.id.w.clust, file = "store.id.avail.clust.csv")

store.id.w.clust.loc <- data.frame(newdf.k8.locb$air_store_id, newdf.k8.locb$k8clust)
head(store.id.w.clust.loc)

colnames(store.id.w.clust.loc) <- c("air_store_id", "cluster.loc")
head(store.id.w.clust.loc)

store.id.w.clust.res <- data.frame(newdf.k5.resb$air_store_id, newdf.k5.resb$k5clust)
head(store.id.w.clust.res)

colnames(store.id.w.clust.res) <- c("air_store_id", "cluster.res")
```

```
head(store.id.w.clust.res)

store.id.w.clust.vol <- data.frame(newdf.k6.volb$air_store_id, newdf.k6.volb$k6clust)
head(store.id.w.clust.vol)

colnames(store.id.w.clust.vol) <- c("air_store_id", "cluster.vol")
head(store.id.w.clust.vol)

store.id.w.clust.vol.2 <- data.frame(newdf.k6.volb.2$air_store_id, newdf.k6.volb.2$k6clust)
head(store.id.w.clust.vol.2)

colnames(store.id.w.clust.vol.2) <- c("air_store_id", "cluster.vol.2")
head(store.id.w.clust.vol.2)


new.clusters <- cbind(store.id.w.clust.avail,
                      store.id.w.clust.loc$cluster.loc,
                      store.id.w.clust.res$cluster.res,
                      store.id.w.clust.vol$cluster.vol)

head(new.clusters)
colnames(new.clusters) <- c("air_store_id", "clus.avail",
                            "clus.loc", "clus.res",
                            "clus.vol")
head(new.clusters)
dim(new.clusters)

write.csv(new.clusters, file = "clusters2.csv")


# Adjusting for new volume clusters
new.clusters.514 <- cbind(store.id.w.clust.avail,
                      store.id.w.clust.loc$cluster.loc,
                      store.id.w.clust.res$cluster.res,
                      store.id.w.clust.vol.2$cluster.vol.2)

head(new.clusters.514)
colnames(new.clusters.514) <- c("air_store_id", "clus.avail",
                            "clus.loc", "clus.res",
                            "clus.vol")
head(new.clusters.514)
dim(new.clusters.514)

write.csv(new.clusters.514, file = "clusters.3.csv")


#########################

# BUSY DAY OF WEEK

#########################
newdf.k6.availb$busy.DOW2 <- factor(newdf.k6.avail$busy.DOW,
```

```
                    levels= c("Sunday", "Monday",
                          "Tuesday", "Wednesday",
                          "Thursday", "Friday", "Saturday"))

head(newdf.k6.avail)
head(newdf.k6.availb)

dim(newdf.k6.avail)
dim(newdf.k6.availb)

#dim(newdf.k6.avail$k6clust)
#newdf.k6.avail$k6clust


tablek6.busyday <- table(newdf.k6.availb$k6clust, newdf.k6.availb$busy.DOW2)
tablek6.busyday

# Function for all groups, for busy days
busy.clus <- function(group, variable, row, Clus)
{
 round(table(group, variable)[row,]/Clus,2)
}

busy.clus(newdf.k8.locb$k8clust, newdf.k8.locb$busy.DOW, 1, k81)
busy.clus(newdf.k8.locb$k8clust, newdf.k8.locb$busy.DOW, 2, k82)
busy.clus(newdf.k8.locb$k8clust, newdf.k8.locb$busy.DOW, 3, k83)
busy.clus(newdf.k8.locb$k8clust, newdf.k8.locb$busy.DOW, 4, k84)
busy.clus(newdf.k8.locb$k8clust, newdf.k8.locb$busy.DOW, 5, k85)
busy.clus(newdf.k8.locb$k8clust, newdf.k8.locb$busy.DOW, 6, k86)
busy.clus(newdf.k8.locb$k8clust, newdf.k8.locb$busy.DOW, 7, k87)
busy.clus(newdf.k8.locb$k8clust, newdf.k8.locb$busy.DOW, 8, k88)

busy.clus(newdf.k5.resb$k5clust, newdf.k5.resb$busy.DOW, 1, k51)
busy.clus(newdf.k5.resb$k5clust, newdf.k5.resb$busy.DOW, 2, k52)
busy.clus(newdf.k5.resb$k5clust, newdf.k5.resb$busy.DOW, 3, k53)
busy.clus(newdf.k5.resb$k5clust, newdf.k5.resb$busy.DOW, 4, k54)
busy.clus(newdf.k5.resb$k5clust, newdf.k5.resb$busy.DOW, 5, k55)

busy.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$busy.DOW, 1, k61.vol)
busy.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$busy.DOW, 2, k62.vol)
busy.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$busy.DOW, 3, k63.vol)
busy.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$busy.DOW, 4, k64.vol)
busy.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$busy.DOW, 5, k65.vol)
busy.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$busy.DOW, 6, k66.vol)




busyday.cluster1 <- tablek6.busyday[1,]/k61
round(busyday.cluster1,2)
```

```
busyday.cluster2 <- tablek6.busyday[2,]/k62
round(busyday.cluster2,2)

busyday.cluster3 <- tablek6.busyday[3,]/k63
round(busyday.cluster3,2)

busyday.cluster4 <- tablek6.busyday[4,]/k64
round(busyday.cluster4,2)

busyday.cluster5 <- tablek6.busyday[5,]/k65
round(busyday.cluster5,2)

busyday.cluster6 <- tablek6.busyday[5,]/k66
round(busyday.cluster6,2)

color <-  c("darkblue", "red", "gray0", "chocolate", "violet",
          "lightblue", 'black')


par(mfrow=c(2,3))
#plot(newdf.k6b.avail$busy.DOW2, main = "Busy Day of Week for All",
#     xlab = "Grouping", ylab = "Frequency",
#     col = color)
barplot(busyday.cluster1, main = "Cluster 1 Busy Day of Week",
      ylab = "Frequency", ylim = c(0,1))
barplot(busyday.cluster2, main = "Cluster 2 Busy Day of Week",
      ylab = "Frequency", ylim = c(0,1))
barplot(busyday.cluster3, main = "Cluster 3 Busy Day of Week",
      ylab = "Frequency", ylim = c(0,1))
barplot(busyday.cluster4, main = "Cluster 4 Busy Day of Week",
      ylab = "Frequency", ylim = c(0,1))
barplot(busyday.cluster5, main = "Cluster 5 Busy Day of Week",
      ylab = "Frequency", ylim = c(0,1))
barplot(busyday.cluster6, main = "Cluster 6 Busy Day of Week",
      ylab = "Frequency", ylim = c(0,1))
par(mfrow=c(1,1))


#########################

# SLOW DAY OF WEEK

#########################
newdf.k6.availb$slow.DOW2 <- factor(newdf.k6.avail$slow.DOW,
                          levels= c("Sunday", "Monday",
                                "Tuesday", "Wednesday",
                                "Thursday", "Friday", "Saturday"))


# Function for all groups, for busy days
slow.clus <- function(group, variable, row, Clus)
```

```
{
 round(table(group, variable)[row,]/Clus,2)
}

slow.clus(newdf.k8.locb$k8clust, newdf.k8.locb$slow.DOW, 1, k81)
slow.clus(newdf.k8.locb$k8clust, newdf.k8.locb$slow.DOW, 2, k82)
slow.clus(newdf.k8.locb$k8clust, newdf.k8.locb$slow.DOW, 3, k83)
slow.clus(newdf.k8.locb$k8clust, newdf.k8.locb$slow.DOW, 4, k84)
slow.clus(newdf.k8.locb$k8clust, newdf.k8.locb$slow.DOW, 5, k85)
slow.clus(newdf.k8.locb$k8clust, newdf.k8.locb$slow.DOW, 6, k86)
slow.clus(newdf.k8.locb$k8clust, newdf.k8.locb$slow.DOW, 7, k87)
slow.clus(newdf.k8.locb$k8clust, newdf.k8.locb$slow.DOW, 8, k88)

slow.clus(newdf.k5.resb$k5clust, newdf.k5.resb$slow.DOW, 1, k51)
slow.clus(newdf.k5.resb$k5clust, newdf.k5.resb$slow.DOW, 2, k52)
slow.clus(newdf.k5.resb$k5clust, newdf.k5.resb$slow.DOW, 3, k53)
slow.clus(newdf.k5.resb$k5clust, newdf.k5.resb$slow.DOW, 4, k54)
slow.clus(newdf.k5.resb$k5clust, newdf.k5.resb$slow.DOW, 5, k55)

slow.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$slow.DOW, 1, k61.vol)
slow.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$slow.DOW, 2, k62.vol)
slow.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$slow.DOW, 3, k63.vol)
slow.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$slow.DOW, 4, k64.vol)
slow.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$slow.DOW, 5, k65.vol)
slow.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$slow.DOW, 6, k66.vol)


tablek6.slowday <- table(newdf.k6.availb$k6clust, newdf.k6.availb$slow.DOW2)
tablek6.slowday

slowday.cluster1 <- tablek6.slowday[1,]/k61
round(slowday.cluster1,2)

slowday.cluster2 <- tablek6.slowday[2,]/k62
round(slowday.cluster2,2)

slowday.cluster3 <- tablek6.slowday[3,]/k63
round(slowday.cluster3,2)

slowday.cluster4 <- tablek6.slowday[4,]/k64
round(slowday.cluster4,2)

slowday.cluster5 <- tablek6.slowday[5,]/k65
round(slowday.cluster5,2)

slowday.cluster6 <- tablek6.slowday[5,]/k66
round(slowday.cluster6,2)

########################

# AVAILABILITY
```

```
########################
tablek6.pref <- table(newdf.k6.availb$k6clust, newdf.k6.availb$Prefecture)
tablek6.pref

# Function for all groups, for number of prefecture
pref.clus <- function(group, variable, row, Clus)
{
 round(table(group, variable)[row,]/Clus,2)
}

pref.clus(newdf.k8.locb$k8clust, newdf.k8.locb$Prefecture, 1, k81)
pref.clus(newdf.k8.locb$k8clust, newdf.k8.locb$Prefecture, 2, k82)
pref.clus(newdf.k8.locb$k8clust, newdf.k8.locb$Prefecture, 3, k83)
pref.clus(newdf.k8.locb$k8clust, newdf.k8.locb$Prefecture, 4, k84)
pref.clus(newdf.k8.locb$k8clust, newdf.k8.locb$Prefecture, 5, k85)
pref.clus(newdf.k8.locb$k8clust, newdf.k8.locb$Prefecture, 6, k86)
pref.clus(newdf.k8.locb$k8clust, newdf.k8.locb$Prefecture, 7, k87)
pref.clus(newdf.k8.locb$k8clust, newdf.k8.locb$Prefecture, 8, k88)

pref.clus(newdf.k5.resb$k5clust, newdf.k5.resb$Prefecture, 1, k51)
pref.clus(newdf.k5.resb$k5clust, newdf.k5.resb$Prefecture, 2, k52)
pref.clus(newdf.k5.resb$k5clust, newdf.k5.resb$Prefecture, 3, k53)
pref.clus(newdf.k5.resb$k5clust, newdf.k5.resb$Prefecture, 4, k54)
pref.clus(newdf.k5.resb$k5clust, newdf.k5.resb$Prefecture, 5, k55)

pref.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$Prefecture, 1, k61.vol)
pref.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$Prefecture, 2, k62.vol)
pref.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$Prefecture, 3, k63.vol)
pref.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$Prefecture, 4, k64.vol)
pref.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$Prefecture, 5, k65.vol)
pref.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$Prefecture, 6, k66.vol)

prefect.cluster1.avail <- tablek6.pref[1,]/k61
round(prefect.cluster1.avail,2)

prefect.cluster2.avail <- tablek6.pref[2,]/k62
round(prefect.cluster2.avail,4)

prefect.cluster3.avail <- tablek6.pref[3,]/k63
round(prefect.cluster3.avail,2)

prefect.cluster4.avail <- tablek6.pref[4,]/k64
round(prefect.cluster4.avail,4)

prefect.cluster5.avail <- tablek6.pref[5,]/k65
round(prefect.cluster5.avail,2)

prefect.cluster6.avail <- tablek6.pref[6,]/k66
round(prefect.cluster6.avail,2)

########################
```

```
# GENRE

########################
tablek6.genre <- table(newdf.k6.availb$k6clust, newdf.k6.availb$Genre)
tablek6.genre

# Function for all groups, for genre
genre.clus <- function(group, variable, row, Clus)
                    {
 round(table(group, variable)[row,]/Clus,2)
}

genre.clus(newdf.k6.availb$k6clust, newdf.k6.availb$Genre, 2, k62)
genre.clus(newdf.k8.locb$k8clust, newdf.k8.locb$Genre, 1, k81)
genre.clus(newdf.k8.locb$k8clust, newdf.k8.locb$Genre, 2, k82)
genre.clus(newdf.k8.locb$k8clust, newdf.k8.locb$Genre, 3, k83)
genre.clus(newdf.k8.locb$k8clust, newdf.k8.locb$Genre, 4, k84)
genre.clus(newdf.k8.locb$k8clust, newdf.k8.locb$Genre, 5, k85)
genre.clus(newdf.k8.locb$k8clust, newdf.k8.locb$Genre, 6, k86)
genre.clus(newdf.k8.locb$k8clust, newdf.k8.locb$Genre, 7, k87)
genre.clus(newdf.k8.locb$k8clust, newdf.k8.locb$Genre, 8, k88)

genre.clus(newdf.k5.resb$k5clust, newdf.k5.resb$Genre, 1, k51)
genre.clus(newdf.k5.resb$k5clust, newdf.k5.resb$Genre, 2, k52)
genre.clus(newdf.k5.resb$k5clust, newdf.k5.resb$Genre, 3, k53)
genre.clus(newdf.k5.resb$k5clust, newdf.k5.resb$Genre, 4, k54)
genre.clus(newdf.k5.resb$k5clust, newdf.k5.resb$Genre, 5, k55)

genre.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$Genre, 1, k61.vol)
genre.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$Genre, 2, k62.vol)
genre.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$Genre, 3, k63.vol)
genre.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$Genre, 4, k64.vol)
genre.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$Genre, 5, k65.vol)
genre.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$Genre, 6, k66.vol)

genre.cluster1.avail <- tablek6.genre[1,]/k61
round(genre.cluster1.avail,2)

genre.cluster2.avail <- tablek6.genre[2,]/k62
round(genre.cluster2.avail,2)

genre.cluster3.avail <- tablek6.genre[3,]/k63
round(genre.cluster3.avail,2)

genre.cluster4.avail <- tablek6.genre[4,]/k64
round(genre.cluster4.avail,4)

genre.cluster5.avail <- tablek6.genre[5,]/k65
round(genre.cluster5.avail,2)

genre.cluster6.avail <- tablek6.genre[6,]/k66
round(genre.cluster6.avail,2)
```

```
#########################

# HOLIDAY

#########################
tablek6.hol <- table(newdf.k6.availb$k6clust, newdf.k6.availb$holidays.with.visits)
tablek6.hol

# Function for all groups, for holiday
hol.clus <- function(group, variable, row, Clus)
{
 round(table(group, variable)[row,]/Clus,2)
}

hol.clus(newdf.k8.locb$k8clust, newdf.k8.locb$holidays.with.visits, 1, k81)
hol.clus(newdf.k8.locb$k8clust, newdf.k8.locb$holidays.with.visits, 2, k82)
hol.clus(newdf.k8.locb$k8clust, newdf.k8.locb$holidays.with.visits, 3, k83)
hol.clus(newdf.k8.locb$k8clust, newdf.k8.locb$holidays.with.visits, 4, k84)
hol.clus(newdf.k8.locb$k8clust, newdf.k8.locb$holidays.with.visits, 5, k85)
hol.clus(newdf.k8.locb$k8clust, newdf.k8.locb$holidays.with.visits, 6, k86)
hol.clus(newdf.k8.locb$k8clust, newdf.k8.locb$holidays.with.visits, 7, k87)
hol.clus(newdf.k8.locb$k8clust, newdf.k8.locb$holidays.with.visits, 8, k88)

hol.clus(newdf.k5.resb$k5clust, newdf.k5.resb$holidays.with.visits, 1, k51)
hol.clus(newdf.k5.resb$k5clust, newdf.k5.resb$holidays.with.visits, 2, k52)
hol.clus(newdf.k5.resb$k5clust, newdf.k5.resb$holidays.with.visits, 3, k53)
hol.clus(newdf.k5.resb$k5clust, newdf.k5.resb$holidays.with.visits, 4, k54)
hol.clus(newdf.k5.resb$k5clust, newdf.k5.resb$holidays.with.visits, 5, k55)

hol.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$holidays.with.visits, 1, k61.vol)
hol.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$holidays.with.visits, 2, k62.vol)
hol.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$holidays.with.visits, 3, k63.vol)
hol.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$holidays.with.visits, 4, k64.vol)
hol.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$holidays.with.visits, 5, k65.vol)
hol.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$holidays.with.visits, 6, k66.vol)
sum(tablek6.hol[,1:16])

hol.cluster1.avail <- tablek6.hol[1,]/k61
round(hol.cluster1.avail,2)

hol.cluster2.avail <- tablek6.hol[2,]/k62
round(hol.cluster2.avail,2)

hol.cluster3.avail <- tablek6.hol[3,]/k63
round(hol.cluster3.avail,2)

hol.cluster4.avail <- tablek6.hol[4,]/k64
round(hol.cluster4.avail,2)

hol.cluster5.avail <- tablek6.hol[5,]/k65
round(hol.cluster5.avail,2)
```

```
hol.cluster6.avail <- tablek6.hol[6,]/k66
round(hol.cluster6.avail,2)


########################

# MEDIAN DAILY VISITORS

########################
tablek6.visit <- table(newdf.k6.availb$k6clust, newdf.k6.availb$median.daily.visitors)
tablek6.visit

sum(tablek6.visit[,1:23])

# Function for all groups, for daily visitors
vis.clus <- function(group, variable, row, Clus)
{
 round(table(group, variable)[row,]/Clus,2)
}

vis.clus(newdf.k8.locb$k8clust, newdf.k8.locb$median.daily.visitors, 1, k81)
vis.clus(newdf.k8.locb$k8clust, newdf.k8.locb$median.daily.visitors, 2, k82)
vis.clus(newdf.k8.locb$k8clust, newdf.k8.locb$median.daily.visitors, 3, k83)
vis.clus(newdf.k8.locb$k8clust, newdf.k8.locb$median.daily.visitors, 4, k84)
vis.clus(newdf.k8.locb$k8clust, newdf.k8.locb$median.daily.visitors, 5, k85)
vis.clus(newdf.k8.locb$k8clust, newdf.k8.locb$median.daily.visitors, 6, k86)
vis.clus(newdf.k8.locb$k8clust, newdf.k8.locb$median.daily.visitors, 7, k87)
vis.clus(newdf.k8.locb$k8clust, newdf.k8.locb$median.daily.visitors, 8, k88)

vis.clus(newdf.k5.resb$k5clust, newdf.k5.resb$median.daily.visitors, 1, k51)
vis.clus(newdf.k5.resb$k5clust, newdf.k5.resb$median.daily.visitors, 2, k52)
vis.clus(newdf.k5.resb$k5clust, newdf.k5.resb$median.daily.visitors, 3, k53)
vis.clus(newdf.k5.resb$k5clust, newdf.k5.resb$median.daily.visitors, 4, k54)
vis.clus(newdf.k5.resb$k5clust, newdf.k5.resb$median.daily.visitors, 5, k55)

vis.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$median.daily.visitors, 1, k61.vol)
vis.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$median.daily.visitors, 2, k62.vol)
vis.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$median.daily.visitors, 3, k63.vol)
vis.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$median.daily.visitors, 4, k64.vol)
vis.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$median.daily.visitors, 5, k65.vol)
vis.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$median.daily.visitors, 6, k66.vol)

visit.cluster1.avail <- tablek6.visit[1,]/k61
round(visit.cluster1.avail,2)
sum(visit.cluster1.avail[1:21])

visit.cluster2.avail <- tablek6.visit[2,]/k62
round(visit.cluster2.avail,2)
sum(visit.cluster2.avail[1:26])

visit.cluster3.avail <- tablek6.visit[3,]/k63
```

```
round(visit.cluster3.avail,2)
sum(visit.cluster3.avail[1:25])

visit.cluster4.avail <- tablek6.visit[4,]/k64
round(visit.cluster4.avail,2)
sum(visit.cluster4.avail[1:43])

visit.cluster5.avail <- tablek6.visit[5,]/k65
round(visit.cluster5.avail,2)
sum(visit.cluster5.avail[1:22])

visit.cluster6.avail <- tablek6.visit[6,]/k66
round(visit.cluster6.avail,2)
sum(visit.cluster6.avail[1:20])

########################

# MEDIAN DATES W VISITS

########################
tablek6.visit.days.week <- table(newdf.k6.availb$k6clust, newdf.k6.availb$mean.days.per.week)
tablek6.visit.days.week

sum(tablek6.visit.days.week[,1:132])

# Function for all groups, for days w visitors
days.clus <- function(group, variable, row, Clus)
{
 round(table(group, variable)[row,]/Clus,2)
}

days.clus(newdf.k8.locb$k8clust, newdf.k8.locb$mean.days.per.week, 1, k81)
sum(days.clus(newdf.k8.locb$k8clust, newdf.k8.locb$mean.days.per.week, 1, k81)[1:141])
sum(days.clus(newdf.k8.locb$k8clust, newdf.k8.locb$mean.days.per.week, 2, k82)[1:140])
sum(days.clus(newdf.k8.locb$k8clust, newdf.k8.locb$mean.days.per.week, 3, k83)[1:138])
sum(days.clus(newdf.k8.locb$k8clust, newdf.k8.locb$mean.days.per.week, 4, k84)[1:230])
sum(days.clus(newdf.k8.locb$k8clust, newdf.k8.locb$mean.days.per.week, 5, k85)[1:142])
sum(days.clus(newdf.k8.locb$k8clust, newdf.k8.locb$mean.days.per.week, 6, k86)[1:105])
sum(days.clus(newdf.k8.locb$k8clust, newdf.k8.locb$mean.days.per.week, 7, k87)[1:124])
sum(days.clus(newdf.k8.locb$k8clust, newdf.k8.locb$mean.days.per.week, 8, k88)[1:209])

days.clus(newdf.k5.resb$k5clust, newdf.k5.resb$mean.days.per.week, 1, k51)
sum(days.clus(newdf.k5.resb$k5clust, newdf.k5.resb$mean.days.per.week, 1, k51)[1:202])
sum(days.clus(newdf.k5.resb$k5clust, newdf.k5.resb$mean.days.per.week, 2, k52)[1:129])
sum(days.clus(newdf.k5.resb$k5clust, newdf.k5.resb$mean.days.per.week, 3, k53)[1:109])
sum(days.clus(newdf.k5.resb$k5clust, newdf.k5.resb$mean.days.per.week, 4, k54)[1:140])
sum(days.clus(newdf.k5.resb$k5clust, newdf.k5.resb$mean.days.per.week, 5, k55)[1:139])

days.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$mean.days.per.week, 1, k61.vol)
sum(days.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$mean.days.per.week, 1, k61.vol)[1:215])
sum(days.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$mean.days.per.week, 2, k62.vol)[1:101])
sum(days.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$mean.days.per.week, 3, k63.vol)[1:131])
```

```
sum(days.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$mean.days.per.week, 4, k64.vol)[1:139])
sum(days.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$mean.days.per.week, 5, k65.vol)[1:91])
sum(days.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$mean.days.per.week, 6, k66.vol)[1:229])

visit.days.week.cluster1.avail <- tablek6.visit.days.week[1,]/k61
round(visit.days.week.cluster1.avail,2)
sum(visit.days.week.cluster1.avail[1:97])

visit.days.week.cluster2.avail <- tablek6.visit.days.week[2,]/k62
round(visit.days.week.cluster2.avail,2)
sum(visit.days.week.cluster2.avail[1:136])

visit.days.week.cluster3.avail <- tablek6.visit.days.week[3,]/k63
round(visit.days.week.cluster3.avail,2)
sum(visit.days.week.cluster3.avail[1:239])

visit.days.week.cluster4.avail <- tablek6.visit.days.week[4,]/k64
round(visit.days.week.cluster4.avail,2)
sum(visit.days.week.cluster4.avail[1:16])

visit.days.week.cluster5.avail <- tablek6.visit.days.week[5,]/k65
round(visit.days.week.cluster5.avail,2)
sum(visit.days.week.cluster5.avail[1:189])

visit.days.week.cluster6.avail <- tablek6.visit.days.week[6,]/k66
round(visit.days.week.cluster6.avail,2)
sum(visit.days.week.cluster6.avail[1:76])

########################

# MEDIAN SUBWAY STOPS NEARBY

########################
tablek6.subway.stops <- table(newdf.k6.availb$k6clust, newdf.k6.availb$subwayStops)
tablek6.subway.stops

# Function for all groups, for number of subway stops
sub.clus <- function(group, variable, row, Clus)
{
 round(table(group, variable)[row,]/Clus,2)
}

sub.clus(newdf.k8.locb$k8clust, newdf.k8.locb$subwayStops, 1, k81)
sub.clus(newdf.k8.locb$k8clust, newdf.k8.locb$subwayStops, 2, k82)
sub.clus(newdf.k8.locb$k8clust, newdf.k8.locb$subwayStops, 3, k83)
sub.clus(newdf.k8.locb$k8clust, newdf.k8.locb$subwayStops, 4, k84)
sub.clus(newdf.k8.locb$k8clust, newdf.k8.locb$subwayStops, 5, k85)
sub.clus(newdf.k8.locb$k8clust, newdf.k8.locb$subwayStops, 6, k86)
sub.clus(newdf.k8.locb$k8clust, newdf.k8.locb$subwayStops, 7, k87)
sub.clus(newdf.k8.locb$k8clust, newdf.k8.locb$subwayStops, 8, k88)

sub.clus(newdf.k5.resb$k5clust, newdf.k5.resb$subwayStops, 1, k51)
```

```
sub.clus(newdf.k5.resb$k5clust, newdf.k5.resb$subwayStops, 2, k52)
sub.clus(newdf.k5.resb$k5clust, newdf.k5.resb$subwayStops, 3, k53)
sub.clus(newdf.k5.resb$k5clust, newdf.k5.resb$subwayStops, 4, k54)
sub.clus(newdf.k5.resb$k5clust, newdf.k5.resb$subwayStops, 5, k55)

sub.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$subwayStops, 1, k61.vol)
sub.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$subwayStops, 2, k62.vol)
sub.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$subwayStops, 3, k63.vol)
sub.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$subwayStops, 4, k64.vol)
sub.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$subwayStops, 5, k65.vol)
sub.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$subwayStops, 6, k66.vol)

sum(tablek6.subway.stops)

subway.stops.cluster1.avail <- tablek6.subway.stops[1,]/k61
round(subway.stops.cluster1.avail,2)
sum(subway.stops.cluster1.avail[1:2])

subway.stops.cluster2.avail <- tablek6.subway.stops[2,]/k62
round(subway.stops.cluster2.avail,2)

subway.stops.cluster3.avail <- tablek6.subway.stops[3,]/k63
round(subway.stops.cluster3.avail,2)

subway.stops.cluster4.avail <- tablek6.subway.stops[4,]/k64
round(subway.stops.cluster4.avail,2)

subway.stops.cluster5.avail <- tablek6.subway.stops[5,]/k65
round(subway.stops.cluster5.avail,2)
sum(subway.stops.cluster5.avail[1:2])

subway.stops.cluster6.avail <- tablek6.subway.stops[6,]/k66
round(subway.stops.cluster6.avail,2)

########################

# MEDIAN NIGHTLIFE NEARBY

########################
tablek6.nightlife <- table(newdf.k6.availb$k6clust, newdf.k6.availb$nightLife)
tablek6.nightlife

sum(tablek6.nightlife)

# Function for all groups, for number of nightlife nearby
night.clus <- function(group, variable, row, Clus)
{
 round(table(group, variable)[row,]/Clus,2)
}

night.clus(newdf.k8.locb$k8clust, newdf.k8.locb$nightLife, 1, k81)
night.clus(newdf.k8.locb$k8clust, newdf.k8.locb$nightLife, 2, k82)
```

```
night.clus(newdf.k8.locb$k8clust, newdf.k8.locb$nightLife, 3, k83)
night.clus(newdf.k8.locb$k8clust, newdf.k8.locb$nightLife, 4, k84)
night.clus(newdf.k8.locb$k8clust, newdf.k8.locb$nightLife, 5, k85)
night.clus(newdf.k8.locb$k8clust, newdf.k8.locb$nightLife, 6, k86)
night.clus(newdf.k8.locb$k8clust, newdf.k8.locb$nightLife, 7, k87)
night.clus(newdf.k8.locb$k8clust, newdf.k8.locb$nightLife, 8, k88)

night.clus(newdf.k5.resb$k5clust, newdf.k5.resb$nightLife, 1, k51)
night.clus(newdf.k5.resb$k5clust, newdf.k5.resb$nightLife, 2, k52)
night.clus(newdf.k5.resb$k5clust, newdf.k5.resb$nightLife, 3, k53)
night.clus(newdf.k5.resb$k5clust, newdf.k5.resb$nightLife, 4, k54)
night.clus(newdf.k5.resb$k5clust, newdf.k5.resb$nightLife, 5, k55)

night.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$nightLife, 1, k61.vol)
night.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$nightLife, 2, k62.vol)
night.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$nightLife, 3, k63.vol)
night.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$nightLife, 4, k64.vol)
night.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$nightLife, 5, k65.vol)
night.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$nightLife, 6, k66.vol)

nightlife.cluster1.avail <- tablek6.nightlife[1,]/k61
round(nightlife.cluster1.avail,2)
sum(nightlife.cluster1.avail[1:5])

nightlife.cluster2.avail <- tablek6.nightlife[2,]/k62
round(nightlife.cluster2.avail,2)
sum(nightlife.cluster2.avail[1:7])

nightlife.cluster3.avail <- tablek6.nightlife[3,]/k63
round(nightlife.cluster3.avail,2)
sum(nightlife.cluster3.avail[1:8])

nightlife.cluster4.avail <- tablek6.nightlife[4,]/k64
round(nightlife.cluster4.avail,2)
sum(nightlife.cluster4.avail[1:7])

nightlife.cluster5.avail <- tablek6.nightlife[5,]/k65
round(nightlife.cluster5.avail,2)
sum(nightlife.cluster5.avail[1:8])

nightlife.cluster6.avail <- tablek6.nightlife[6,]/k66
round(nightlife.cluster6.avail,2)
sum(nightlife.cluster6.avail[1:7])

########################

# MEDIAN NUMBER OF DAILY RESERVATIONS

########################
tablek6.num.res <- table(newdf.k6.availb$k6clust, newdf.k6.availb$median.daily.reservations)
tablek6.num.res
```

```
# Function for all groups, for number of daily reservations
daily.res.clus <- function(group, variable, row, Clus)
{
 round(table(group, variable)[row,]/Clus,2)
}

daily.res.clus(newdf.k8.locb$k8clust, newdf.k8.locb$median.daily.reservations, 1, k81)
daily.res.clus(newdf.k8.locb$k8clust, newdf.k8.locb$median.daily.reservations, 2, k82)
daily.res.clus(newdf.k8.locb$k8clust, newdf.k8.locb$median.daily.reservations, 3, k83)
daily.res.clus(newdf.k8.locb$k8clust, newdf.k8.locb$median.daily.reservations, 4, k84)
daily.res.clus(newdf.k8.locb$k8clust, newdf.k8.locb$median.daily.reservations, 5, k85)
daily.res.clus(newdf.k8.locb$k8clust, newdf.k8.locb$median.daily.reservations, 6, k86)
daily.res.clus(newdf.k8.locb$k8clust, newdf.k8.locb$median.daily.reservations, 7, k87)
daily.res.clus(newdf.k8.locb$k8clust, newdf.k8.locb$median.daily.reservations, 8, k88)

daily.res.clus(newdf.k5.resb$k5clust, newdf.k5.resb$median.daily.reservations, 1, k51)
daily.res.clus(newdf.k5.resb$k5clust, newdf.k5.resb$median.daily.reservations, 2, k52)
daily.res.clus(newdf.k5.resb$k5clust, newdf.k5.resb$median.daily.reservations, 3, k53)
daily.res.clus(newdf.k5.resb$k5clust, newdf.k5.resb$median.daily.reservations, 4, k54)
daily.res.clus(newdf.k5.resb$k5clust, newdf.k5.resb$median.daily.reservations, 5, k55)

daily.res.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$median.daily.reservations, 1, k61.vol)
daily.res.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$median.daily.reservations, 2, k62.vol)
daily.res.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$median.daily.reservations, 3, k63.vol)
daily.res.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$median.daily.reservations, 4, k64.vol)
daily.res.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$median.daily.reservations, 5, k65.vol)
daily.res.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$median.daily.reservations, 6, k66.vol)

num.res.cluster1.avail <- tablek6.num.res[1,]/k61
round(num.res.cluster1.avail,2)
sum(num.res.cluster1.avail[1:5])

num.res.cluster2.avail <- tablek6.num.res[2,]/k62
round(num.res.cluster2.avail,2)
sum(num.res.cluster2.avail[1:7])

num.res.cluster3.avail <- tablek6.num.res[3,]/k63
round(num.res.cluster3.avail,2)
sum(num.res.cluster3.avail[1:8])

num.res.cluster4.avail <- tablek6.num.res[4,]/k64
round(num.res.cluster4.avail,2)
sum(num.res.cluster4.avail[1:7])

num.res.cluster5.avail <- tablek6.num.res[5,]/k65
round(num.res.cluster5.avail,2)
sum(num.res.cluster5.avail[1:8])

num.res.cluster6.avail <- tablek6.num.res[6,]/k66
round(num.res.cluster6.avail,2)
sum(num.res.cluster6.avail[1:7])
```

```
########################

# MEDIAN NUMBER OF DAILY VISITORS FROM RESERVATIONS

########################
tablek6.num.res.vis <- table(newdf.k6.availb$k6clust, newdf.k6.availb$median.daily.res.visitors)
tablek6.num.res.vis

# Function for all groups, for number of daily visitors from reservations
daily.res.vis.clus <- function(group, variable, row, Clus)
{
 round(table(group, variable)[row,]/Clus,2)
}

daily.res.vis.clus(newdf.k8.locb$k8clust, newdf.k8.locb$median.daily.res.visitors, 1, k81)
daily.res.vis.clus(newdf.k8.locb$k8clust, newdf.k8.locb$median.daily.res.visitors, 2, k82)
daily.res.vis.clus(newdf.k8.locb$k8clust, newdf.k8.locb$median.daily.res.visitors, 3, k83)
daily.res.vis.clus(newdf.k8.locb$k8clust, newdf.k8.locb$median.daily.res.visitors, 4, k84)
daily.res.vis.clus(newdf.k8.locb$k8clust, newdf.k8.locb$median.daily.res.visitors, 5, k85)
daily.res.vis.clus(newdf.k8.locb$k8clust, newdf.k8.locb$median.daily.res.visitors, 6, k86)
daily.res.vis.clus(newdf.k8.locb$k8clust, newdf.k8.locb$median.daily.res.visitors, 7, k87)
daily.res.vis.clus(newdf.k8.locb$k8clust, newdf.k8.locb$median.daily.res.visitors, 8, k88)

daily.res.vis.clus(newdf.k5.resb$k5clust, newdf.k5.resb$median.daily.res.visitors, 1, k51)
daily.res.vis.clus(newdf.k5.resb$k5clust, newdf.k5.resb$median.daily.res.visitors, 2, k52)
daily.res.vis.clus(newdf.k5.resb$k5clust, newdf.k5.resb$median.daily.res.visitors, 3, k53)
daily.res.vis.clus(newdf.k5.resb$k5clust, newdf.k5.resb$median.daily.res.visitors, 4, k54)
daily.res.vis.clus(newdf.k5.resb$k5clust, newdf.k5.resb$median.daily.res.visitors, 5, k55)

daily.res.vis.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$median.daily.res.visitors, 1, k61.vol)
daily.res.vis.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$median.daily.res.visitors, 2, k62.vol)
daily.res.vis.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$median.daily.res.visitors, 3, k63.vol)
daily.res.vis.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$median.daily.res.visitors, 4, k64.vol)
daily.res.vis.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$median.daily.res.visitors, 5, k65.vol)
daily.res.vis.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$median.daily.res.visitors, 6, k66.vol)

num.res.vis.cluster1.avail <- tablek6.num.res.vis[1,]/k61
round(num.res.vis.cluster1.avail,2)
sum(num.res.vis.cluster1.avail[1:5])

num.res.vis.cluster2.avail <- tablek6.num.res.vis[2,]/k62
round(num.res.vis.cluster2.avail,2)
sum(num.res.vis.cluster2.avail[1:7])

num.res.vis.cluster3.avail <- tablek6.num.res.vis[3,]/k63
round(num.res.vis.cluster3.avail,2)
sum(num.res.vis.cluster3.avail[1:8])

num.res.vis.cluster4.avail <- tablek6.num.res.vis[4,]/k64
round(num.res.vis.cluster4.avail,2)
sum(num.res.vis.cluster4.avail[1:7])
```

```
num.res.vis.cluster5.avail <- tablek6.num.res.vis[5,]/k65
round(num.res.vis.cluster5.avail,2)
sum(num.res.vis.cluster5.avail[1:8])

num.res.vis.cluster6.avail <- tablek6.num.res.vis[6,]/k66
round(num.res.vis.cluster6.avail,2)
sum(num.res.vis.cluster6.avail[1:7])

########################

# MEDIAN NUMBER OF TOURIST ATTRACT NEARBY

########################
tablek6.tourist.att <- table(newdf.k6.availb$k6clust, newdf.k6.availb$touristAct)
tablek6.tourist.att

# Function for all groups, for number of tourist attractions nearby
tourist.clus <- function(group, variable, row, Clus)
{
 round(table(group, variable)[row,]/Clus,2)
}

tourist.clus(newdf.k8.locb$k8clust, newdf.k8.locb$touristAct, 1, k81)
tourist.clus(newdf.k8.locb$k8clust, newdf.k8.locb$touristAct, 2, k82)
tourist.clus(newdf.k8.locb$k8clust, newdf.k8.locb$touristAct, 3, k83)
tourist.clus(newdf.k8.locb$k8clust, newdf.k8.locb$touristAct, 4, k84)
tourist.clus(newdf.k8.locb$k8clust, newdf.k8.locb$touristAct, 5, k85)
tourist.clus(newdf.k8.locb$k8clust, newdf.k8.locb$touristAct, 6, k86)
tourist.clus(newdf.k8.locb$k8clust, newdf.k8.locb$touristAct, 7, k87)
tourist.clus(newdf.k8.locb$k8clust, newdf.k8.locb$touristAct, 8, k88)

tourist.clus(newdf.k5.resb$k5clust, newdf.k5.resb$touristAct, 1, k51)
tourist.clus(newdf.k5.resb$k5clust, newdf.k5.resb$touristAct, 2, k52)
tourist.clus(newdf.k5.resb$k5clust, newdf.k5.resb$touristAct, 3, k53)
tourist.clus(newdf.k5.resb$k5clust, newdf.k5.resb$touristAct, 4, k54)
tourist.clus(newdf.k5.resb$k5clust, newdf.k5.resb$touristAct, 5, k55)

tourist.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$touristAct, 1, k61.vol)
tourist.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$touristAct, 2, k62.vol)
tourist.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$touristAct, 3, k63.vol)
tourist.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$touristAct, 4, k64.vol)
tourist.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$touristAct, 5, k65.vol)
tourist.clus(newdf.k6.volb.2$k6clust, newdf.k6.volb.2$touristAct, 6, k66.vol)

tourist.att.cluster1.avail <- tablek6.tourist.att[1,]/k61
round(tourist.att.cluster1.avail,2)
sum(tourist.att.cluster1.avail[1:3])

tourist.att.cluster2.avail <- tablek6.tourist.att[2,]/k62
round(tourist.att.cluster2.avail,2)
sum(tourist.att.cluster2.avail[1:3])
```

```
tourist.att.cluster3.avail <- tablek6.tourist.att[3,]/k63
round(tourist.att.cluster3.avail,2)
sum(tourist.att.cluster3.avail[1:3])

tourist.att.cluster4.avail <- tablek6.tourist.att[4,]/k64
round(tourist.att.cluster4.avail,2)
sum(tourist.att.cluster4.avail[1:3])

tourist.att.cluster5.avail <- tablek6.tourist.att[5,]/k65
round(tourist.att.cluster5.avail,2)
sum(tourist.att.cluster5.avail[1:3])

tourist.att.cluster6.avail <- tablek6.tourist.att[6,]/k66
round(tourist.att.cluster6.avail,2)
sum(tourist.att.cluster6.avail[1:5])
```