Capstone Project 1: Milestone Report

## Problem Statement

Running a thriving business can be more challenging than it seems. For many businesses, accurately forecasting the number of customers on a given day may be the difference between turning a profit and losing money. This is especially true of restaurants where inventory has a short shelf life and over- or under-staffing can be costly.

Predicting customer turnout at restaurants can be challenging due to unforeseen circumstances like weather and changing local competition. This is even harder for newer restaurants with little historical data.  
  
Recruit Holdings has access to unique datasets that may make automated future customer prediction possible. Specifically, Recruit Holdings owns AirREGI (a restaurant point of sales service), and Restaurant Board (reservation log management software).  
  
In this project, I draw from these and other data sources to predict the total number of visitors to a restaurant on a given date. This information will help restaurants be more efficient with regard to both purchasing and staffing. It will also free up resources so that restaurants can focus on creating an enjoyable dining experience for their customers.

## Data

Data for this project came from AirREGI/Restaurant Board (AIR) system. AIR is a reservation and cash register system that provides users increased convenience and flexibility as they dine out. Files from the AIR system include reservations, visits, and other information from restaurants across Japan. These time-series data span from January, 2016 through April, 2017.

In these data, the following is provided for each restaurant:  
  
**Air\_store\_id** - the restaurant's id in the air system  
**Visit\_datetime** - the time of the reservation  
**Reserve\_datetime** - the time the reservation was made  
**Reserve\_visitors** - the number of visitors for that reservation  
**Air\_genre\_name** - type of restaurant

**Air\_area\_name** - region where the store is located  
**Latitude** - approximate latitude of the restaurant  
**Longitude** - approximate longitude of the restaurant  
**Visitors** - the number of visitors to the restaurant on the date

In addition to the restaurant information described above, this project uses date-specific information (i.e. day of the week, holiday, etc.) as well as weather information linked to each restaurant. These include:

**Calendar\_date** - date of each reservation  
**Day\_of\_week** - day of the week for each calendar date

**Holiday\_flg** - indicates that the day is a holiday in Japan

**Avg\_temperature** - average daily temperature at the weather station closest to each restaurant

**High\_temperature** - daily high temperature at the weather station closest to each restaurant

**Low\_temperature** - daily low temperature at the weather station closest to each restaurant

**Precipitation** - daily precipitation total at the weather station closest to each restaurant

**Hours\_sunlight** - daily hours of sunlight at the weather station closest to each restaurant

## Data Wrangling

I began by downloading the relevant AIR files from the Kaggle API and reading them into my local Python environment. These included reservation data, visitor data, restaurant information, calendar information, and weather data. The visitor data contained the variable of interest: number of visitors for each transaction. These data were reported at the transaction-level and, therefore, contained multiple cases per restaurant, per day. Because the outcome of interest is the number of visitors per day, I chose to group the data by restaurant, resampled the data by day, and summed the number of visitors at each restaurant on each day. I repeated this process with the reservation data, but chose to count the number of reservations made for each day.

Because the visitor data contained the outcome of interest, I used a left join on the visitor data to bring in the reservation data. This ensured that all of the relevant visitor data was retained and irrelevant reservation data was excluded. I then merged the restaurant information and calendar information files using the same method.

Wrangling the weather data was more complicated because these data were stored in separate files for each weather station. Additionally, weather station ids were provided in the file names, but were not contained as data in the files. To address this, I wrote a function that read each file into an empty list while simultaneously creating an id variable generated from the id in the filename. I then concatenated these data frames to create a single weather data frame where each observation contained unique weather information for each day and for each weather station. Using a crosswalk provided by Kaggle, I then linked the restaurant data to the weather data from the nearest weather station for each day. This final data frame contained 296,279 unique observations.

Once the final data frame was created, I examined the percentage of values that were missing for each variable. These are presented in the following table:

|  |  |
| --- | --- |
| **Variable** | **Percent Missing** |
| air\_store\_id | 0% |
| visit\_date | 0% |
| visitors | 0% |
| number\_of\_reservations | 76.2% |
| air\_genre\_name | 0% |
| air\_area\_name | 0% |
| latitude | 0% |
| longitude | 0% |
| day\_of\_week | 0% |
| holiday\_flg | 0% |
| station\_id | 0% |
| avg\_temperature | 9.9% |
| high\_temperature | 9.9% |
| low\_temperature | 9.9% |
| precipitation | 30.4% |
| hours\_sunlight | 14.1% |

Because the vast majority of data was missing from number\_of\_reservations, I chose to exclude it from the analysis. The only other variables with missing values were from the weather files. With each of these weather variables, I began by creating a missing flag to keep track of the observations that would contain imputed values. I then calculated the global daily median value for each variable and replaced missing values with the median value for that day. I chose the median because it will be robust to non-normal distributions.

Additionally, I examined outliers. Using descriptive statistics (i.e. min, max, mean, median, std, etc.), I determined that there were no extreme outliers in any of the predictors. However, the outcome (i.e. number of daily visitors) contained several extreme outliers. I dealt with these by first identifying observations with values greater than 3.5 standard deviations above the restaurant-specific mean. I then replaced these values with 3.5 standard deviations above the restaurant-specific mean to create a maximum value ‘cap’ for each restaurant. This eliminated the large residual values that would be created by these observations while still capturing the fact that their values were in the extremes of the distribution.

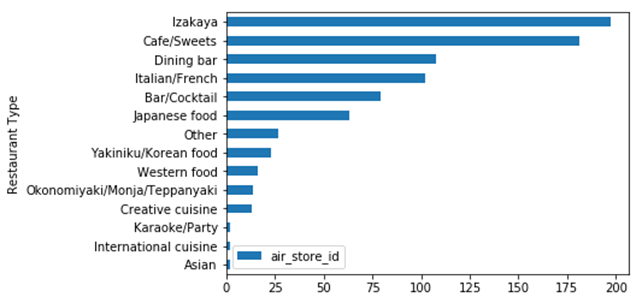
As a final step, I created ‘naive’ features from the data that are typical in time-series analyses. These included rolling means, medians, and standard deviations that were calculated by restaurant over a thirty day window. Given the context in which the resulting model will be used, I lagged these rolling features by one week and replaced missing values with the global mean/median/standard deviation for each restaurant. This was appropriate given that restaurants will likely only have data in advance of when they will need to make predictions. I also created numeric and categorical features for days of the week and days of the month. This should help address any seasonal and trend effects. Finally, I created categorical features for each restaurant genre and region in the dataset as well as the number of restaurants in each region as a proxy for restaurant “competition”.

## Exploratory Data Analysis: Descriptive Statistics and Graphical Analysis

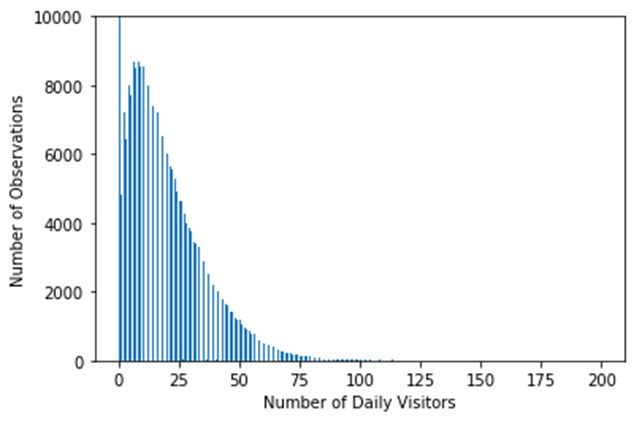
As an initial step, I conducted exploratory data analysis (EDA) to gain a better understanding of the data. In total, the data contained 296,279 unique day-restaurant observations that spanned from Friday, January 1st, 2016 through Saturday, April 22nd, 2017. These included information for 829 restaurants in 103 neighborhoods in Japan. The number of restaurants were not equally distributed across neighborhoods. The neighborhoods with the largest number of restaurants are included the following table:

|  |  |
| --- | --- |
| **Neighborhood** | **Number of Restaurants** |
| Fukuoka-ken Fukuoka-shi Daimyō | 64 |
| Tōkyō-to Shibuya-ku Shibuya | 58 |
| Tōkyō-to Minato-ku Shibakōen | 51 |
| Tōkyō-to Shinjuku-ku Kabukichō | 39 |
| Tōkyō-to Setagaya-ku Setagaya | 30 |
| Tōkyō-to Chūō-ku Tsukiji | 29 |
| Ōsaka-fu Ōsaka-shi Ōgimachi | 25 |
| Hiroshima-ken Hiroshima-shi Kokutaijimachi | 23 |
| Tōkyō-to Meguro-ku Kamimeguro | 22 |
| Hokkaidō Sapporo-shi Minami 3 Jōnishi | 21 |

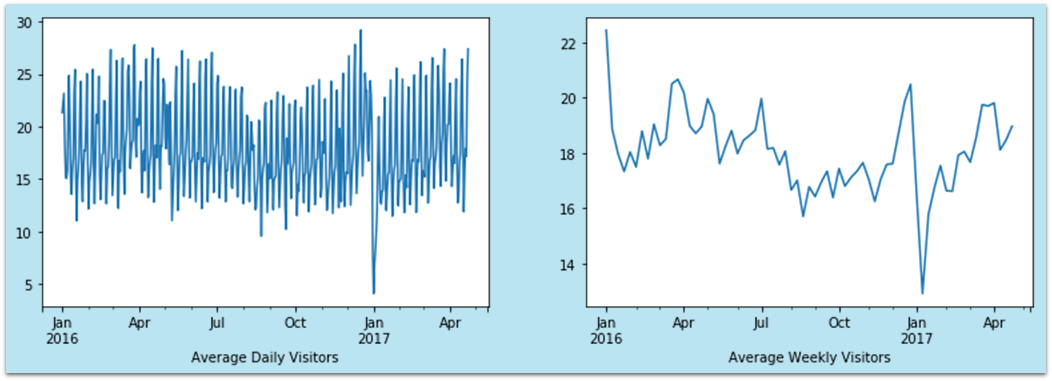
The restaurants in these data represented fourteen ‘genres’ the most frequent of which were ‘Izakaya’ and ‘Cafe/Sweets.’



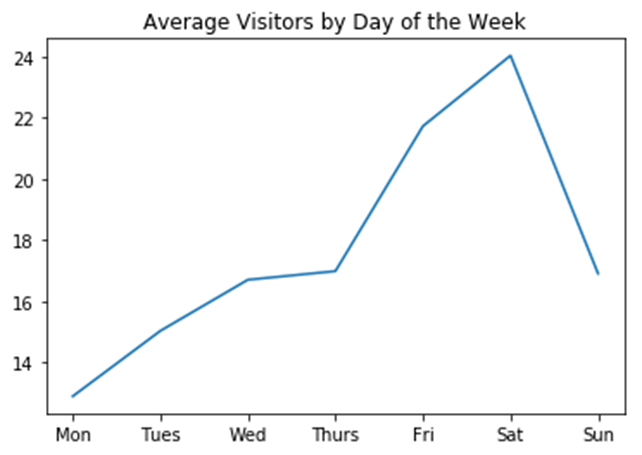
The average number of visitors across all days and all restaurants was 17.8 with a standard deviation of 16.6. The minimum value for the number of visitors was 0, the median value was 14, and the maximum value was 199.9. In this dataset, there were many observations with zero visitors (44,171).



Looking at trends in the data, there are seasonal trends both within and across months. The number of visitors is highest in April through July, December, and January. The months between August and November seem to be slowest for these restaurants.

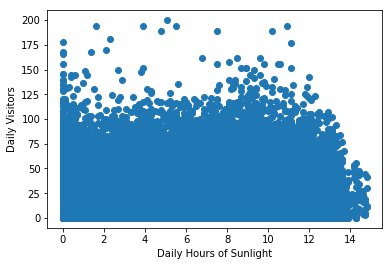


Looking at differences in average daily visitors by day of the week, Monday is the slowest day with the average number of visitors increasing steadily throughout the week and dropping sharply on Sunday.



There appears to be a slight positive relationship between average daily visitors and average daily temperature and a negative relationship between average daily visitors and daily precipitation. There does not appear to be a relationship between average daily visitors and hours of sunlight. Despite these graphical insights, formal statistical tests are needed to confirm these results.

|  |  |
| --- | --- |
|  |  |



## Exploratory Data Analysis: Statistical Analyses

I completed further EDA by conducting formal statistical tests. In all of these analyses, I examined relationships with the outcome of interest: the number of daily visitors at each restaurant.

The first question I sought to answer was whether the observed differences in average visitors by day of the week were significantly different from one another. To do this, I estimated a linear regression model with dummy variables for each day of the week as predictors and the log+1 transformed number of visitors as the outcome. In this model, I used Monday as the reference category since the graphical analysis showed that Mondays have the fewest visitors, on average. As the table below shows, each day of the week had significantly more visitors, on average, compared to Mondays. The pattern of the coefficients reflects the pattern shown in the graphical analysis.

|  |  |
| --- | --- |
| **Day of Week** | **Coefficient/Std. Err** |
| Tuesday | 0.30\*\*\*  (0.01) |
| Wednesday | 0.42\*\*\*  (0.01) |
| Thursday | 0.49\*\*\*  (0.01) |
| Friday | 0.80\*\*\*  (0.01) |
| Saturday | 0.83\*\*\*  (0.01) |
| Sunday | 0.10\*\*\*  (0.01) |
| R-squared = .05\*\*\* | |
| \* = *p* < .05, \*\* = *p* < .01, \*\*\* = *p* < .001 | |

I also examined correlations between each of the numeric features and the untransformed outcome. As the table below shows, all variables except for high-temperature were significantly related to the number of restaurant visitors. By far, the strongest relationships were with the lagged ‘naive’ features with correlations ranging .48 to .64.

|  |  |
| --- | --- |
| **Variable** | **Correlation with Number of Visitors** |
| avg\_temperature | -.01\*\* |
| high\_temperature | -.00 |
| low\_temperature | -.01\*\*\* |
| precipitation | -.04\*\*\* |
| hours\_sunlight | .04\*\*\* |
| day\_of\_month | .03\*\*\* |
| rolling\_mean\_lag | .61\*\*\* |
| rolling\_median\_lag | .58\*\*\* |
| rolling\_std\_lag | .48\*\*\* |
| visitors\_lag | .64\*\*\* |
| competition | -.03\*\*\* |
| \* = *p* < .05, \*\* = *p* < .01, \*\*\* = *p* < .001 | |

## Model Building and Selection

For modeling, I chose to use extreme gradient boosting using the XGBoost package in Python. Extreme gradient boosting is a variant of gradient boosting which takes a series of “weak learners” (i.e. models with modest predictive capabilities) and combines their results into a single “strong learner” (i.e. a model that is strongly predictive). In gradient boosting, a series of models are fit on several random samples of observations and features. The results from these models are then averaged to give the final prediction. This is similar to the ‘bagging’ approach used in many machine learning algorithms. Where gradient boosting diverges from bagging is in its treatment of misclassified observations. Specifically, at each iteration, misclassified observations from the previous models are given greater weight. This allows each successive model to learn from the ‘weaknesses’ of the previous models thus providing more accurate results.

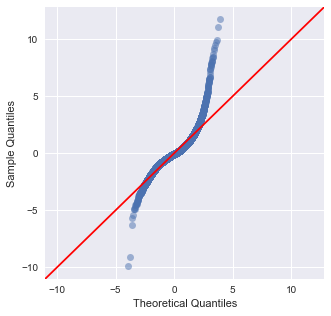
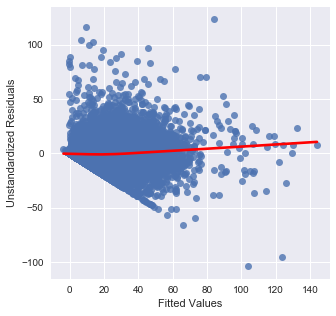
As a base-learner in my XGBoost model, I chose random forests with the number of visitors on a given day as the target. Although some features were more highly-correlated with the target than others, and several features were highly correlated with each other, the random forests used in the XGBoost algorithm are robust to these issues. Therefore, I chose to include all of the features described above in the model.

My initial model used all the features described in the previous sections to predict the number of visitors at each restaurant on a given day (i.e. the target). I began by dividing my data into training and test sets. The training data consisted of all observation from January 1st, 2016 through March 27th, 2017. The test data, on the other hand, consisted of all observations between March 28th, 2017 and April 22nd, 2017. The features used in this model should not present contamination issues between the training and test sets as data from the test set was not used to calculate the features in the training set. It should also be noted that in the training data, I used a version of the target variable where extreme outliers had been capped at 3.5 standard deviations above the restaurant-specific mean. This should reduce the influence of outliers on the model and, therefore, reduce its variance across future samples. However, for the test data, I chose to include extreme outliers in the target variable to simulate real-world data.

Next, I tuned the models hyperparameters on the training data using scitkitlearn’s RandomizedSearchCV using the default 3-fold cross-validation. Specifically, I tuned the following hyperparameters: subsample (the fraction to be sampled at each tree), n\_estimators (the number of trees in each random forest), colsample\_bytree (the fraction of features to be selected for each tree in the random forest), learning rate, and max\_depth (the maximum number of branches in each tree).

I then fit the model with the tuned hyperparameters to the test data. To gauge the improvement of the model over a null-model, I calculated the RMSE with the median number of visitors from the test set as the predicted value. This baseline value was 18.94. The initial model, by comparison, produced an RMSE value of 13.9. The residuals indicated the model was systematically over- and under-predicting the number of visitors for low- and high-traffic days, respectively. To address this, I calculated additional features that were meant to better predict these values. These included the average number of visitors per restaurant, the average number of visitors per restaurant per day of the week, as well as dummy indicators for high-, medium-, and low-traffic restaurants. These new features were calculated from the training data and imputed into the test data to ensure that there was no contamination between datasets.

I then re-tuned the hyperparameters and fit the model. The addition of these features added significantly to the model’s predictive power with an RMSE of 11.8. After examining the residuals, two extreme outliers were identified and removed from the data as they had a strong influence on the RMSE. The revised baseline and model-estimated RMSEs were 18.1 and 10.5, respectively. In other words, the model reduced the average misprediction by 42 percent when compared to an empty model. The residuals are shown below plotted against the fitted values (with a loess line superimposed) and compared to a normal distribution via a Q-Q plot.



From these, we see that there is no trend in the residuals and that the residuals are normally distributed near the mean, but deviate from normality at the tails of the distribution. As a final check on model quality, I checked for overfitting by examining the RMSE when the model was fit to the training data and compared that to the test data. The training RMSE was 9.2 which is fairly close to the test RMSE of 10.5 suggesting that the model did not overfit the training data.

To address the non-normality in the residuals, I also estimated a model on a log-transformed outcome following all of the procedures described above. This also conforms to best practices in traditional time-series analysis. This model produced an RMSE of 12.1 on the untransformed target variable which is higher than the RMSE produced by the previous model. Therefore, I selected the model for the untransformed target as my final model.