Restaurants face many problems; one of which is trying to predict how many customers will visit the restaurant on each day. This is helpful so that the appropriate number of staff can be scheduled and that the right amount of food can be ordered. If these can be optimized, costs can be reduced while still serving the customers needs.

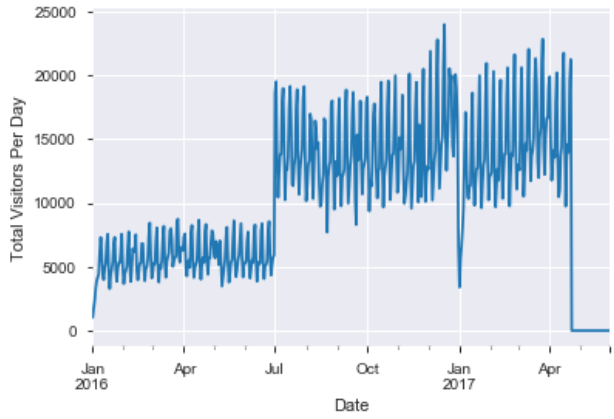
There were many files of data provided by the Kaggle competition. The first file contains a store ID, the date, and the number of visitors. I began by concatenating the test set to the training set so that I would have to merge the rest of the data only once. The file I merged to the visit data contained information about the store including genre and area name. The second file contained the day of the week for each date and whether it was a holiday.

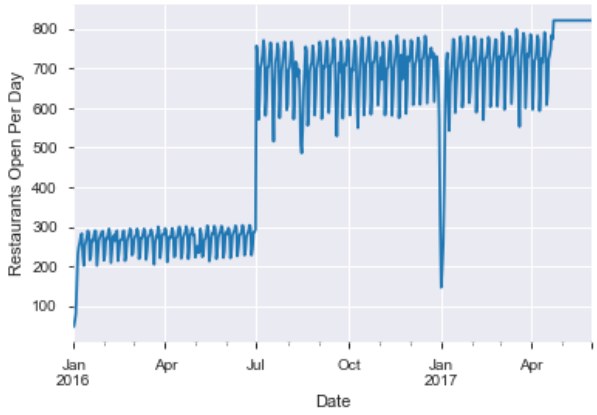
When looking at what external data other people had used for this, I found that someone had gathered weather data from 1,600+ stations across Japan as well as how far each of these stations is from each of the 108 unique latitude/longitude pairs. For many of these locations, the closest station only measured precipitation or temperature. By averaging the 5 closest stations, I was able to get temperature and precipitation measurements for every location for every day. Some of the other measurements were still missing. I filled them first by the average value in the same city on that day, and I filled the rest by the average of all measurements on the day.

To begin my exploratory analysis, I used the pandas profiling module to create a report about each of the variables. It found that four of the weather measurements were highly correlated to

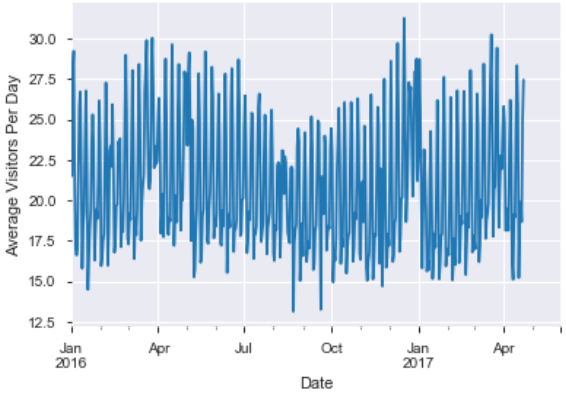
others and should be excluded: average sea pressure, average vapor pressure, high temperature, and low temperature.

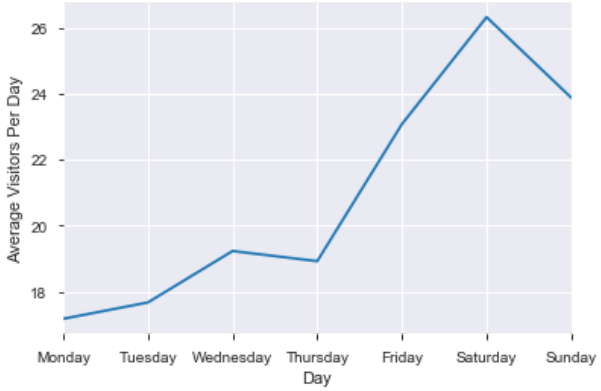
I found two interesting things when looking at the total visitors per day over the entire date range. There appears to be a weekly cycle and at the beginning of July the total visitors goes from ~50k to 150k. I thought this second change may be due to an increase in the number of restaurants, and I found that this also almost tripled in early July.





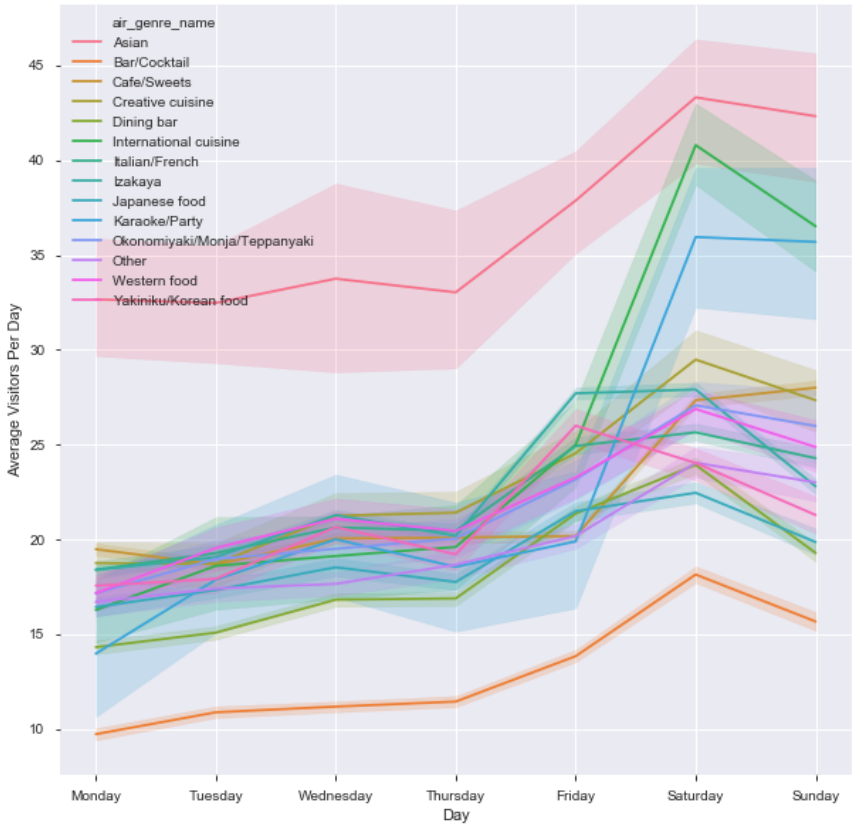
I decided to look at average visitors per day to remove the effect of a different number of restaurants being open each day. The daily data showed that there may be a seasonal variation in visitors, but the change is smaller than the day to day change. By averaging over a week and then a month, the seasonal change became more noticable.



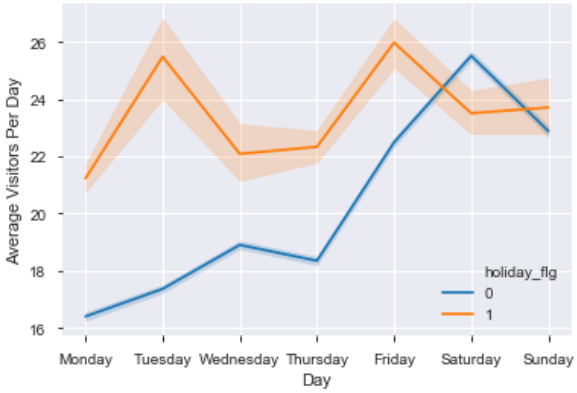


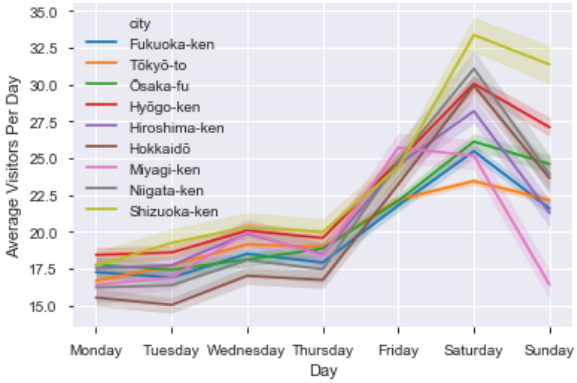
I next wanted to investigate whether total visitors per day graph was showing a weekly trend or if it was just noise. When looking at the average visitors per day based on the day of the week, Monday through Thursday have a much smaller average than Friday through Sunday. Saturday is the maximum with an average 50% more than Monday.

Looking at this weekly trend and separating by the other categorical features yielded some interesting results. Most of the 14 genres of restaurant are fairly tightly clustered with only 4 that



stand out. The ‘Asian’ genre has a consistently much larger number of customers while the ‘BarCocktail’ genre is consistently much lower. The ‘International cuisine’ and ‘Karaoke/Party’ genres blend in with the rest on Monday through Friday, but have a much larger increase on Saturday and Sunday. If there is a holiday on a weekday, the number of visitors increases, but on Saturday and Sunday, there appears to be no effect. The city of the restaurant has no clear effect on the number of visitors.





I then ran an F-test on each of the categorical features and found that all have statistically significant statistics. This means that for each feature there is at least one pair of categories that have different means. The only feature where this is surprising is 'city.' This can be made sense of by realizing that even though they are clustered together, the highest mean may be significantly different from the lowest because there is so much data.

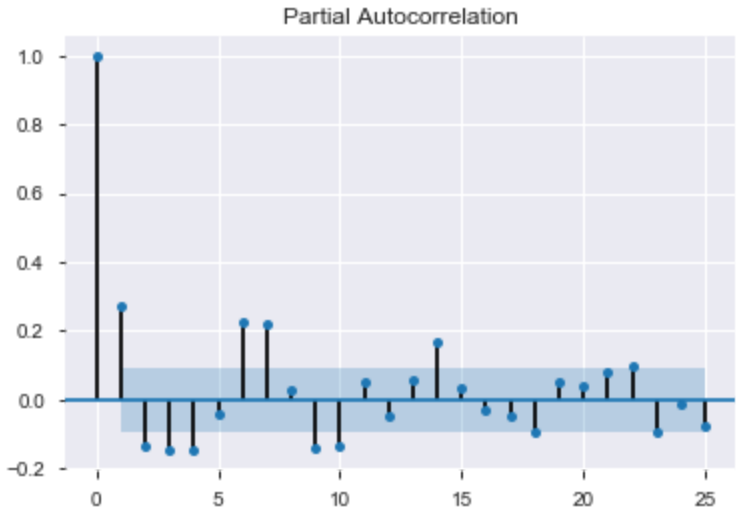
With the exploratory analysis finished, I began building models and creating derived features. I used the Kaggle test set and other competitors’ scores to assess the performance of my models. The best public score was a root mean squared log error of 0.465, the 25th percentile was 0.479, the 50th percentile was 0.489, and the 75th percentile was 0.521. My goal for this project was to have a score that was not in the bottom quartile.

The first model I trained was a random forest using only the original features described so far; this scored near 0.786. Since the competition is over, I was able to look through a few competitors

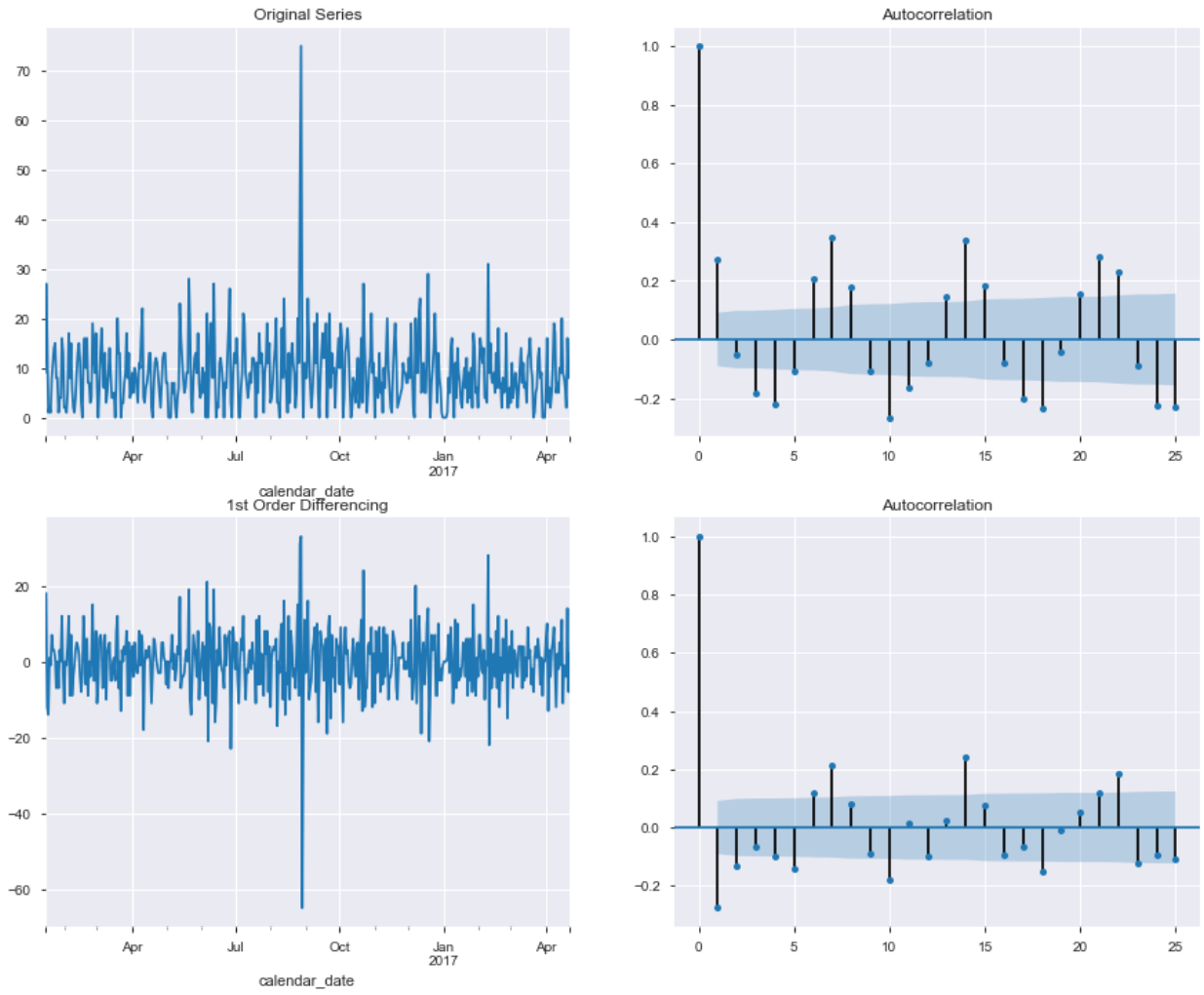
solutions. I found that many of the used the time series nature of this data to extract statistics about the number of visitors on a rolling basis. The function I used was largely based on the one used by Max Halford here. I extracted statistics for each restaurant and for each restaurant grouped by day. The score of a random forest with these features was 0.530 which was a large decrease and almost to my goal.

The location of each restaurant was given at three levels: city, prefecture, and subprefecture. The pandas profiling report gave a warning for the latter two for high cardinality, 55 and 99 distinct values respectively. Because of this warning, I tested removing the subprefecture column (0.53115) and both prefecture and subprefecture columns (0.53318). Having all levels of detail slightly improved the performance of the model.

I then began testing hyperparameters for the random forest and gradient boosted regressors. Increasing the number of trees in the forest had an unpredictable effect on score; 200 trees performed worse than 100, but 300 performed better than 100. For the gradient boosted regressor, I found that incrementally increasing the number of trees from 100 to 1,000 improved the RMSLE continuously. A maximum depth of 5 for each regressor performed better than 3 for each number of estimators.

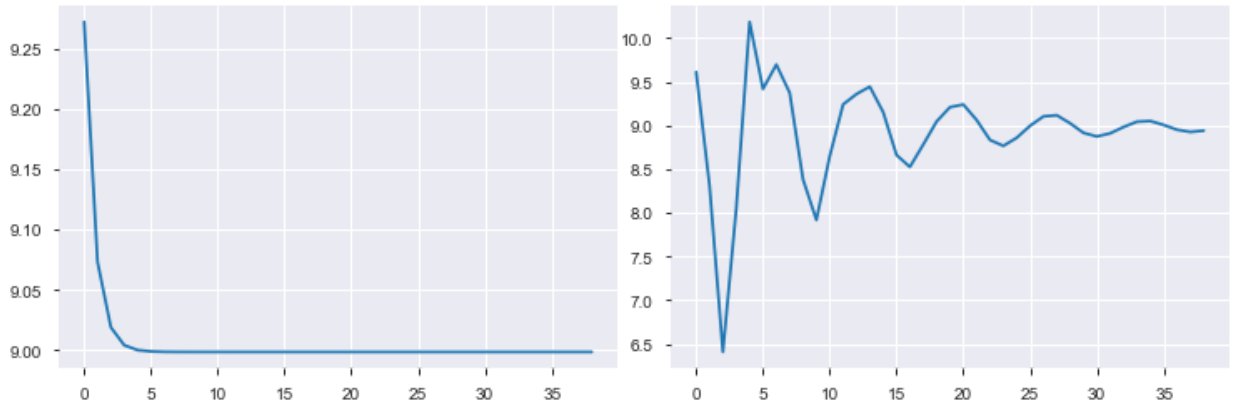


To use the time-series nature of these data in a more direct way, I decided to use a separate ARIMA model for each restaurant to make the predictions. There are three parameters that must be determined to build an ARIMA model which I estimated for a random restaurant and assumed they would be the same for all of them. The first is the AutoRegressive term which can be estimated using a plot of the partial autocorrelation. In the plot above, most of the lags less than or equal to seven are significant. This makes sense since there is a weekly variation in the number of visitors. The second is the Integrated term which is given by the number of times the data need to be differenced to make the series stationary. The original series appears to be stationary which would mean that the second parameter is zero. This is confirmed by plotting the autocorrelation of the first order differencing and seeing that the first few terms are significantly negative. The final is the



Moving Average term which is estimated from the autocorrelation plot with the correct order of differencing. Since the second term is not significant, I set this parameter to one initially. From this I started with an ARIMA(7, 0, 1).

The ARIMA model summary gives the significance for each term and I used this to determine the final values for the three parameters. The term with the largest p-value was the MA term so I removed it. For the next step, the second and third AR terms were not significant, but the higher terms were. To determine if I should use 1 or 7 for the AR term, I made predictions for the restaurant with each value. The first graph shows the predictions with AR = 1 and the second shows them with AR = 7. Since there is almost no variation in the first, the final model I used was ARIMA(7,0, 0). With the parameters set, I trained a model for each restaurant and used them to forecast the future visitors. Even without any other information, these models did much better than the very first model I trained, but worse than the best so far with an RMSLE of 0.56343.



I found that there is a type of ARIMA model that accepts other features in addition to the main time series. I tried to train models with all of the additional features, but the program would always crash on a random restaurant. This is likely because the summary showed that almost none of the features were significant. I still wanted to use these predictions, so I added them as a feature for the gradient boosted regressor; however, this did not end up improving the score.

Since I had seen continuous improvement in the score as the number of estimators increased, I decided to train a model with 10,000. In this case, the score was worse which shows that bigger models do not always mean better accuracy.

While working on this project, I learned some important concepts. The time-series nature of this dataset made it easy to create new features about the number of visitors, and this greatly improved the accuracy of the predictions. Sometimes simple models can perform fairly well such as an ARIMA model looking only at the number of visitors each day to each restaurant. My final take-away is similar and is that larger models are not always necessary.