Recruit Holdings: Restaurant Visitor Forecasting

Batool Fatima | Brady Engelke | Fazel | Jie Liu | Noah Becker | Patrick Seng

Project Definition

Problem Statement

Recruit Holdings' Dining Services in Japan has uncovered an opportunity for new customer growth and increased customer retention. Using its vast amount of data on restaurant visits and reservations across Japan, they believe they can substantially increase their value in the eyes of restaurant owners by marketing their accurate forecasting capabilities as one of their major product offerings. Our team hopes to help them make enhanced predictions to grow their business and ultimately drive revenue for Recruit Holdings.

The goal of this project is to predict the number of people that will visit 829 restaurants between April 23rd, 2017 and May 31st, 2017. The provided data contains information on restaurants in various parts of Japan from January 2016 to April 2017, which will be used to train our static and dynamic models and to predict the number of visitors for each restaurant throughout the specified 39-day period.

Naturally, this can be treated as a time series forecasting problem since the predictions are dependent on the number of visitors observed on the preceding dates with an element of seasonality. Despite these convoluting factors, this problem can also be cast into a standard regression problem. It is important to note that the test data spans a holiday week in Japan called the 'Golden Week' which is intended to be an additional ambiguity the algorithm must learn to traverse. There are days in the test set where the restaurant was closed and had no visitors which are omitted from scoring.

Primary Data

The primary data consists of 8 tables gathered from two systems that maintain and collect user information: Hot Pepper Gourmet (hpg), similar to the search and reserve functionality of Yelp, and AirREGI (air), similar to the reservation control and cash register system capabilities of Square. Each file is prefixed with either air or hpg, indicating its source. It's important to note that both systems do not cover the same number of restaurants, air contains data for 829 restaurants whereas hpg covers 4,690 restaurants with only 150 restaurants common between the two.

Analysis

Overview

Meticulous data cleaning and table merging was initially conducted to ensure the team extracted as much information as possible from the primary datasets. The team then appended simple features to the dataset to provide the predictive algorithm with fundamental contextual information. Advanced features were then engineered to incorporate customer behavior variation based on weather patterns and the historical behavior of customers on similar dates.

After equipping the dataset with all of the necessary features to train a model that captures a substantial amount of our target variable's variation, dummy variables were created for categorical variables, irrelevant columns were dropped, and all columns were converted to the float data type. Next, the dataset was sliced into training and validation sets in a way that closely simulated the actual prediction process. The team concluded that the 39-day period directly preceding the actual 39-day test period would suffice.

The team experimented with both static and dynamic models to determine whether a time-series or a classic regression approach was more suitable for the use case. The algorithms were selected to provide Recruit Holdings with a variety of algorithmic approaches, some easy to interpret, others highly complex.

Data Cleaning

The disparate .csv files were merged into a master dataframe indexed on air store ID and date. Data [6] was selected as the base table since the column containing the actual daily visitor information was contained within this file. It was found that the dataset contained 829 unique air stores spanning over 517 dates. A cartesian product was then taken to construct the final dataset, resulting in 428,593 rows. Many of the stores were not open for business for the entire duration which required careful imputation later on. When merging the hpg files, data [2] and data [4], all information that didn't correspond to the 829 unique air stores could be dropped. The joining of these additional files added holiday and reservation information from air and hpg as well as the area in Japan, longitude/latitude, and the genre of food each store offers. Unfortunately, only 10% of the 428,593 rows within the base table were appended with reservation information but all rows did contain genre, longitude/latitude, holiday, and area information. Listed below are the features that we selected from the primary data:

- Hpg reservation information
- AIR system reservation information
- Genre
- Area
- Geographical information (longitude, latitude)
- Holiday

Feature Engineering

The next step was to append to the master dataset all of the contextual information required for the predictive model to produce accurate predictions. The team aimed to simulate the information a veteran forecaster would take into consideration when attempting to predict the number of visitors for a respective restaurant on a certain date. Listed below are the simple features that were appended to the master dataset:

- Day of week
- Week of month
- Month of year
- Year
- Earliest open date

- Number of days the store has been in business
- Whether the store was open, closed, or not in business for each day
- Whether it was the day before or day after a holiday

The day before and after a holiday were deemed valuable since it is likely that customer behavior demonstrably varies on such days due to holiday preparations or recovery. The earliest open date feature provided a starting point to track the number of days a store was in business, providing static models with some historical context. The earliest open date proved useful for imputation when deriving more advanced features but does have a notable drawback which is discussed later on in this report.

"Look-back" Features

The following features were developed to provide the algorithm with features that would closely emulate the volume of visitors to be expected on a particular date by referencing a similar date in the past's volume of visitors. Several date intervals were debated upon such as looking back exactly one year, 6 months, 3 months, 1 month, or a combination thereof. The team concluded that looking back more than 6 weeks would ensure these features could be captured for all of the test data instances. This prevented the algorithm from making predictions based upon predictions. Furthermore, looking back a year would result in an imputation issue since a majority of stores were not open for the entire duration of the data collection period. Thus, a compromise was made to look back exactly 6, 7, 8, and 9 weeks from a respective date to compute the following features:

- Average number of visitors by store 6, 7, 8, and 9 weeks ago
- Average number of visitors by genre 6, 7, 8, and 9 weeks ago
- Average number of visitors by area 6, 7, 8, and 9 weeks ago
- Standard deviation of visitors by store 6, 7, 8, and 9 weeks ago
- Standard deviation of visitors by genre 6, 7, 8, and 9 weeks ago
- Standard deviation of visitors by area 6, 7, 8, and 9 weeks ago

Several edge cases had to be addressed when implementing the function. To elaborate on these complexities the team has provided a flowchart (figure 1) to illustrate the function's computation of the mean number of visitors by store. Note, the function's filter steps only need to be adjusted when computing the average or standard deviation for genre or area.

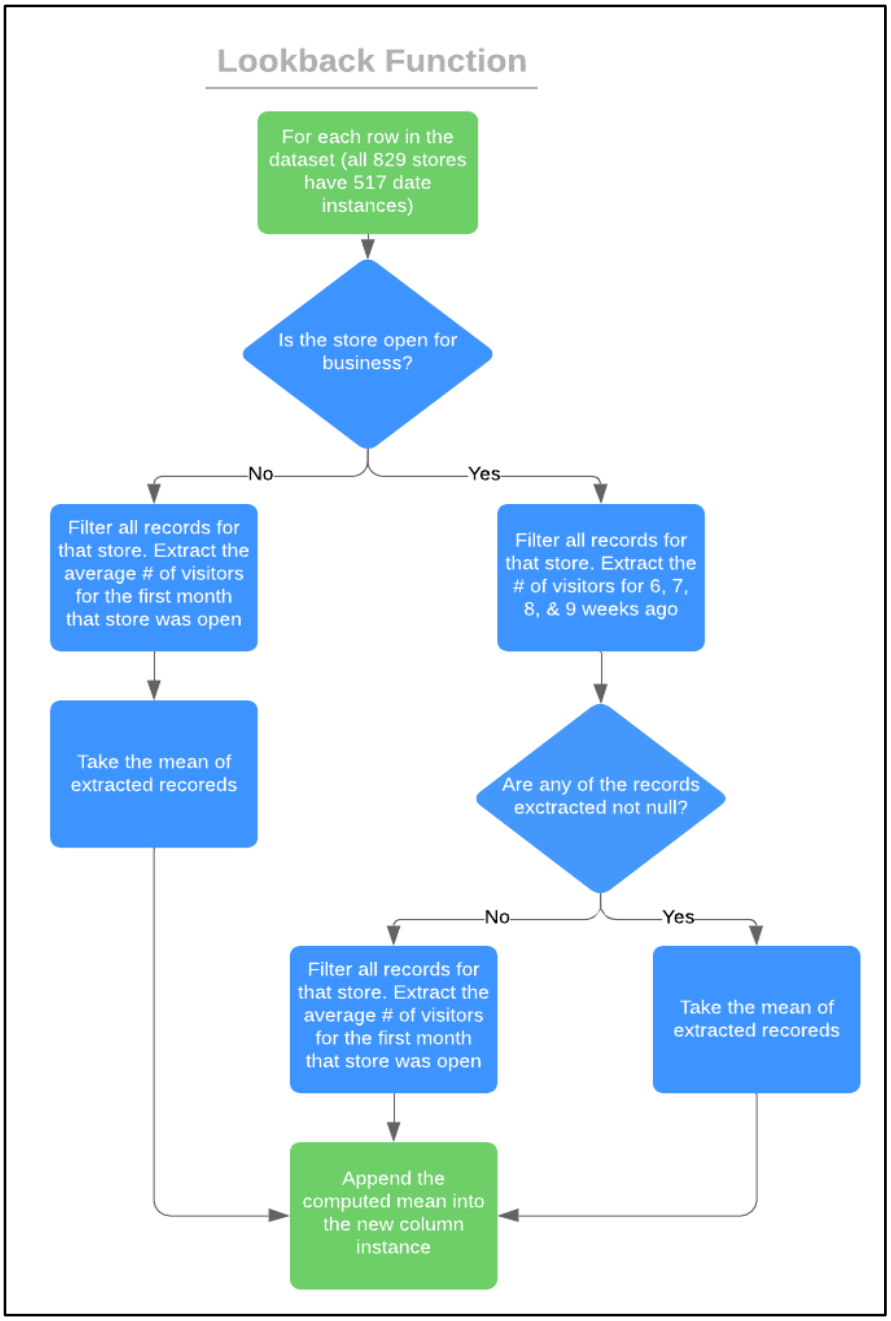

Figure 1: "Look-back" Function

Weather Features

In addition to the primary dataset, the team accessed supplementary data regarding weather across Japan from the Japan Meteorological Agency compiled by Hunter McGushion, data [8]. To combine this data with the master dataset, the team used the daily weather recorded at the nearest weather station to each restaurant, data [9]. A wide variety of features have been recorded by these weather stations, although it is not consistent from station to station; some stations have data on every feature while others may have information on just one. Relying upon intuition for informativeness, explainability. and the amount of data present, the team extracted the following features:

- Average, low, and high temperature
- Amount of precipitation
- Hours of sunlight (without cloud cover)
- Average wind speed
- Average humidity

Each of these features could logically affect the amount of traffic to a restaurant. Although features with extremely large amounts of missing values were excluded, around 10-30% of data remained missing amongst the selected features. To address this, the team first attempted to impute the mean value for each of the 7 weather features grouped by the area in which the restaurant is located. Unfortunately, it was discovered that there were many areas present in the primary dataset and missing values were widespread. To give a better estimation than simply taking a national mean each day, the team made use of the latitude and longitude. Each day, missing values were filled using the average value of its 3 nearest not-null neighbors in Euclidean distance using the k-nearest neighbors algorithm. Euclidean distance was used since it was the best approximation of the distance between two restaurants and provided the most accurate representation of geographic proximity.

Modeling Preparation

With all of the features in hand, the team began manipulating the dataset into a digestible format for the predictive models. First, all irrelevant columns were dropped. These included: earliest open date and day of year. These columns were deemed irrelevant since other columns better captured the variation in the target variable and care had to be taken to determine how many columns to convert to dummy variables. Then, all nulls in the air reservations, hpg reservations, and visitors columns were filled with 0 and dummy variables were created for the categorical features: genre name, area name, month, year, day of week, week of month, and air store id. The calendar date column was then utilized to split up the dataset into training, validation, and testing sets. The filters for each set are as follows:

- Training: all dates before March 15th, 2017
- Validation: dates inclusively between March 15th, 2017 and April 22nd, 2017
- Test: dates after April 22nd, 2017

The team reasoned that the most accurate validation period would be a consecutive 39-day period at the end of the training set since this most closely emulated the true prediction time interval which was also a consecutive 39-day period. Finally, the calendar date column was dropped from the datasets, all variables were converted to float64 type, and the visitors column was extracted to serve as the target variable.

Static Modeling

The following models were selected as candidate models:

- Linear Model
- K-NN
- Gradient Boosting Regression Tree
- Light Gradient Boosting Regression Tree

The candidate model list was selected based upon sequential complexity to provide the stakeholders interpretable as well as accurate results. Normalization was applied to the training and testing sets before fitting the K-NN Regression model to ensure no one feature dominated the calibration process. GridSearchCV was employed to tune the hyper-parameters within each model. Unfortunately, Gradient Boosting Regression Tree faced an intractable computation task when implementing GridSearchCV. This led the team to utilize a Light Gradient Boosting Regression Tree so we could tune the hyper-parameters within a reasonable amount of time.

After each model successfully predicted the number of visitors for the 3- days of the validation and test sets, the team imputed 0 for any negative predictions made by the algorithms, since these predictions were simply not feasible for the use case. The models were then compared based upon three metrics: Mean Squared Error, Root Mean Squared Log Error, and R-Squared. The client deemed the root mean squared log error as the most important metric before the project. Therefore, this was the metric that carried the most weight when selecting the final model.

Dynamic Modeling

Next, the team used a recurrent neural network (RNN) to explore a time series approach. Seq2Seq refers to the sequence-to-sequence architecture of the neural network fit. This architecture enables mapping arbitrary length sequences to other arbitrary length sequences. This ability helps Seq2Seq shine in performing many tasks including language translation, image captioning, and time series prediction. RNN's ability to successfully learn from data with long range temporal dependencies makes it a natural choice for time series prediction. However, because the RNN follows a vastly different structure to a traditional regression approach, a detailed explanation is necessary.

The Seq2Seq architecture is comprised of two parts, an encoder and a decoder. The team designed an encoder which looks into 400 days of historical data and a decoder to predict 39 days of future visitor traffic. The same features used for static modeling were then inputted into the encoder with the intention that it would pick up patterns embedded within the sequential 400 days' worth of data. The decoder was fed stationary information about the 39 upcoming days, such as flags for holidays, area, and genre.

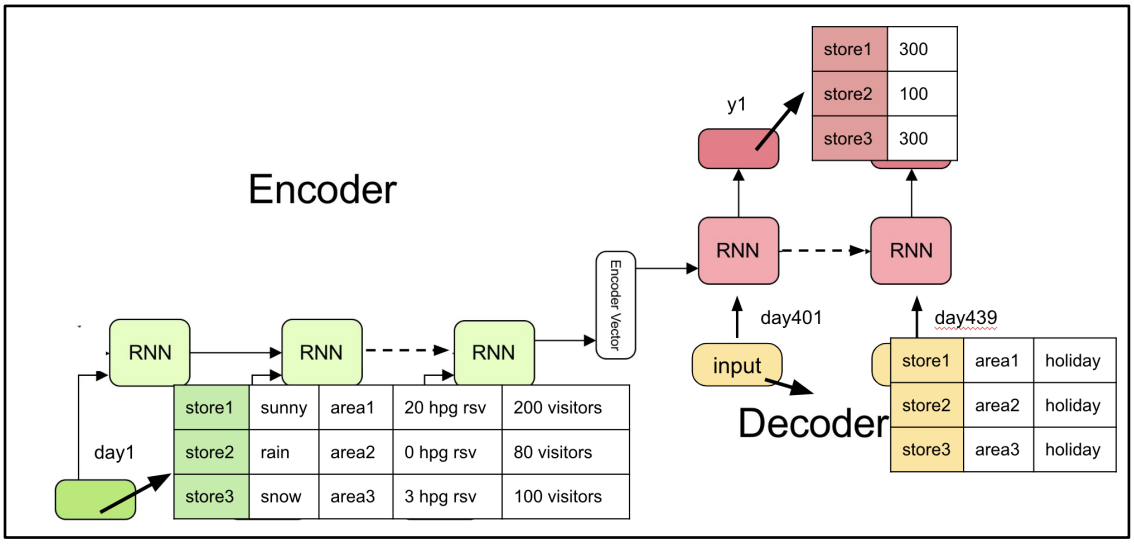

Figure 2: Seq2Seq Architecture Illustrating Encoder & Decoder

The team adopted a walk forward train and validation split method to grid search for optimal parameters. To do so, data from January 1st, 2016 to March 14th, 2017 was used for training and data from February 9th, 2016 to April 22nd, 2017 was used as a validation set.

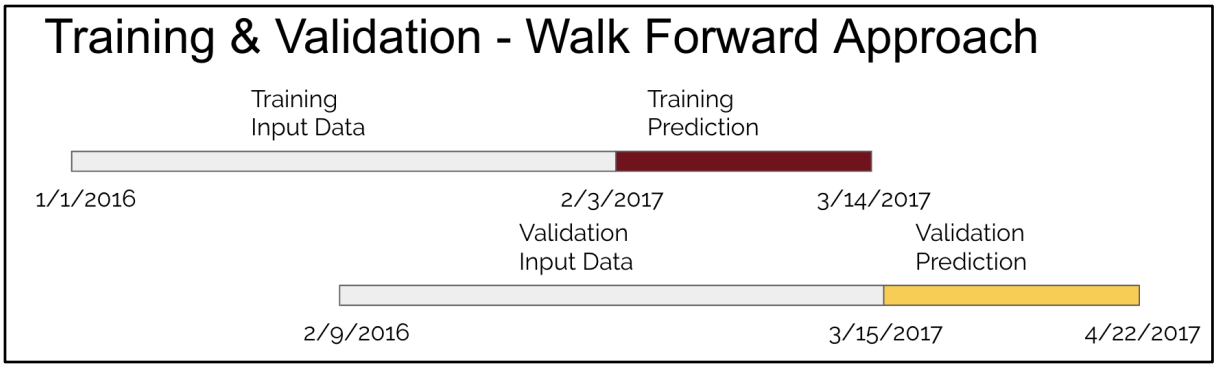

Figure 3: Training & Validation Splits

First, the model structure was tuned. The number of neurons in the LSTM cells and the depth of LSTM layers are known to have a significant effect on model performance. Then, the learning rate was tuned for the task at hand. The learning rate is of major concern for hyperparameter tuning because it affects the speed of convergence substantially and the ability to escape the local minimum solutions for the global minimum solutions. The team found that a structure with 2 LSTM layers and 256 neurons per LSTM cell using a 0.001 learning rate Adam optimizer resulted in the best performance. A sigmoid activation function was used for LSTM layers using ReLU as the activation function resulted in NaN as the loss, a sign of the 'Dying ReLU' problem. Finally, the team searched multiple dropout rates to help reduce the overfitting problem. A description of the final neural network architecture after tuning is provided below (Figure 4).

Layer (type)	Output Shape	Param #	Connected to
input_23 (InputLayer)	(None, None, 146)	0	
input_24 (InputLayer)	(None, None, 142)	0	
encoder (RNN)	[(None, 256), (None,	937984	input_23[0][0]
decoder (RNN)	[(None, None, 256),	933888	input_24[0][0] encoder[0][1] encoder[0][2] encoder[0][3] encoder[0][4]
dense_23 (Dense)	(None, None, 100)	25700	decoder[0][0]
time_distributed_12 (TimeDistri	(None, None, 1)	101	dense_23[0][0]

Total params: 1,897,673
Trainable params: 1,897,673
Non-trainable params: 0

Figure 4: Neural Network Summary

After 40 epochs of training, the model performance stagnated at around 1.4 RMSLE. The final performance upon the validation set was 1.240 RMSLE and 20.097 RMSE. This was by far inferior to the performance of the Light Gradient Boosting Tree. The prediction RMSLE as evaluated by Kaggle was 0.842, still much poorer than that of the best static model.

The RNN model predictions range from 9 to 20 visitors per day, implying that it may have learned to generalize too much, losing the variance of the visitor traffic. Also, the dummy variables for categorical data substantially increased the dimensions of the input data. The team conjectures that if there are more neurons per LSTM cell, the model might be able to pick up the latent pattern better. However, due to technical limitations, the Kaggle GPU ran out of memory space when training on 1024 neurons and the team was not able to implement more complex structures.

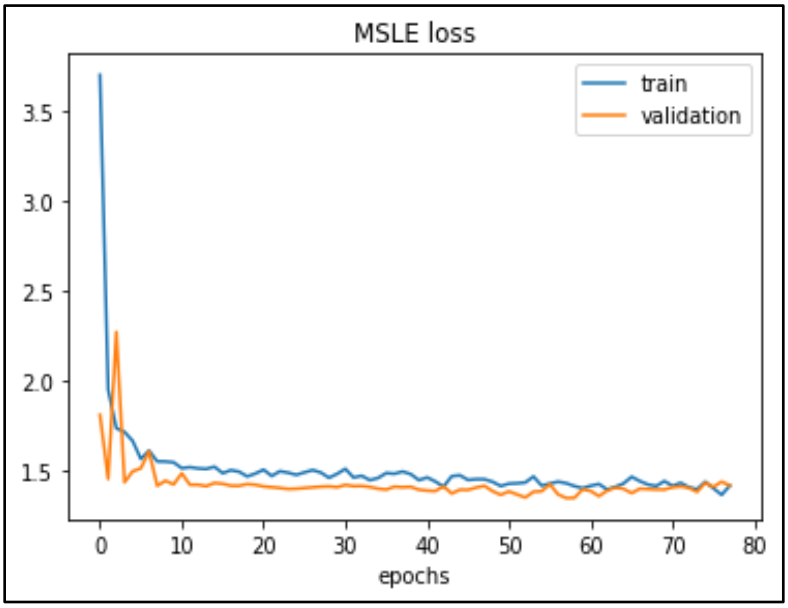

Figure 5: Seq2Seq Learning Curve

Model Comparison

Finally, the team was ready to compare the models. Below is a table (figure 6) of each models' performance regarding the metrics chosen for evaluation as well as a plot (figure 7) illustrating the Mean Absolute Error of each model over the validation period.

Model	MSE	RMSLE	R^2
LGBR	**135.39**	**0.536**	**0.613**
Gradient Boost	143.981	0.541	0.588
Linear Regression	177.479	0.843	0.493
k-NN	272.347	0.982	0.221
RNN	403.608	1.230	0.201

Figure 6: Model Performance Comparison

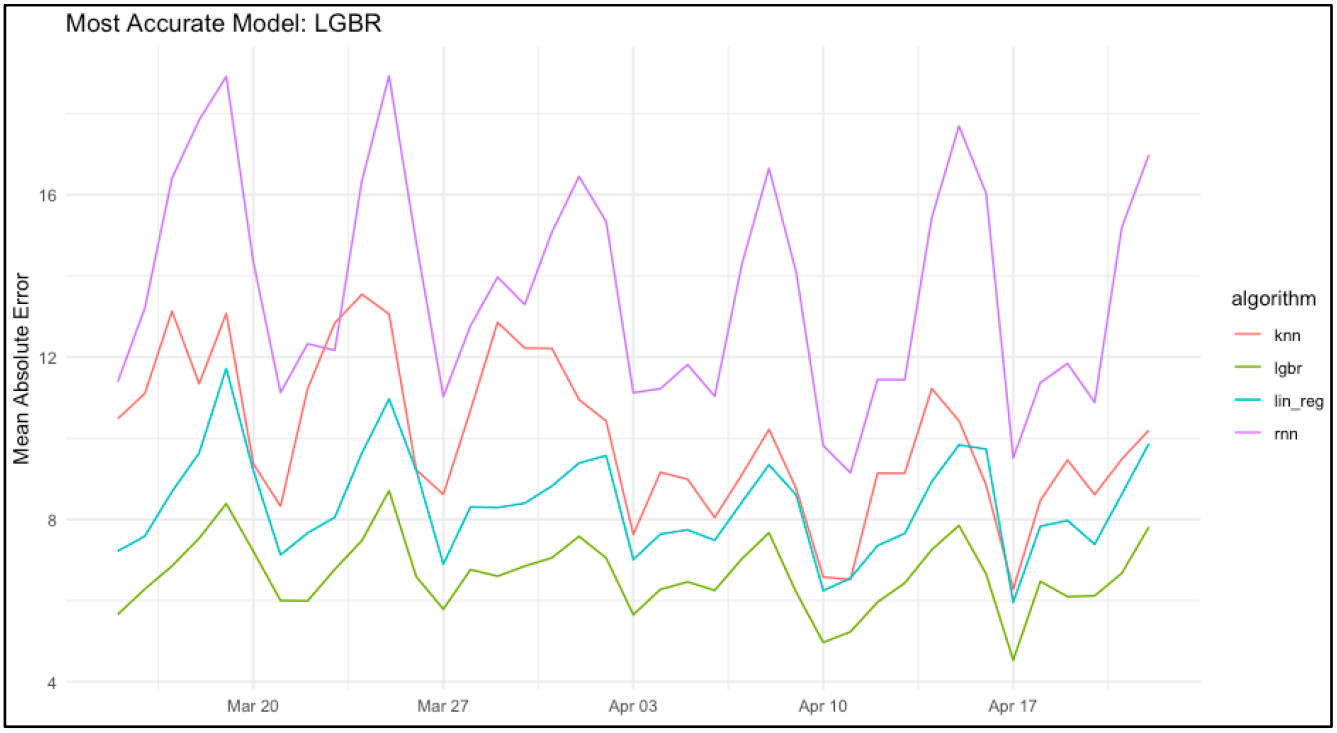

Figure 7: MAE Over Validation Period

LGBR outperformed all other models on every single day of the validation set, a tribute to the beauty of residual gradient descent. Linear regression performed incredibly well despite its simplicity and assumption of linearity. k-NN seemed to be hindered by the curse of dimensionality since the creation of dummy variables exploded our feature space.

Insights and Implementation

Recommended Implementation

The team suggests that Recruit Holdings utilize the Light Gradient Boosting Regression Tree as the primary means for making its restaurant visitor predictions. To continue using this model for predictions on future dates while maintaining similar accuracy measures, the data must be updated frequently to capture any changes in patterns that might be affecting restaurant visitors. New data in the same format as the current data set must first be fed through the feature engineering process outlined above and included with the original data when re-training the models. While the Light Gradient Boosting Regression Tree offered the best predictive power, the team recommends that the linear regression model be maintained in parallel. While this model did not produce the strongest predictive power, its simplicity fosters business understanding and would enable restaurant owners to gain a better understanding of which features in the data might be affecting traffic. Additionally, both models can be easily adjusted if more features become available to Recruit Holdings.

To further increase prediction accuracy in the future, Recruit Holdings should consider gathering additional features for each of its customer restaurants. More detailed genre descriptions or tagging based on user reviews is one way that Recruit Holdings could expand upon the genre column in the current data. Additionally, information regarding the physical size of each restaurant may help predict the number of daily visitors. In the current data, only information on whether each day is a Japanese national holiday is provided. The team suspects that more informative labels that differentiate holidays from one another would add to the models' predictive power. Finally, the true opening date of each restaurant would be highly valuable. A limitation of the implementation detailed is that the number of days open treats all restaurants who were open on the first day of the training set as opening that day. Despite this, it was observed that the number of days the restaurant was open had significant importance when LGBR made predictions. Thus, a more accurate measure could only improve the model's accuracy.

Impact

Using the model detailed above for prediction on the 39-day test period, the team achieved a final Root Mean Squared Log Error of 0.567. Based on the provided information, Recruit Holdings' Dining Services group did not have a prior capability for predicting visitors to their restaurants algorithmically. Due to its relatively simple implementation process, the team used linear regression as the baseline of comparison while keeping in mind it is unlikely that many customers of Recruit Holdings were using any predictive modeling techniques to forecast their daily visitors. Compared to the linear regression model, the Light Gradient Boosting Regression Tree is more accurate on average by 2 visitors for each restaurant each day. This may seem like an insignificant improvement from the baseline at first glance, but when scaled up to the 829 restaurants, compounded by the need for predictions to be made on an ongoing basis, this incremental improvement provides tremendous value. Also, keep in mind, the baseline regression model makes use of the features the team has engineered. Specifically, the average number of visitors for each store 6 - 9 weeks ago that was provided by the "look-back" function was shown to capture more variation in the target variable than any other feature previously captured by Recruit Holdings.

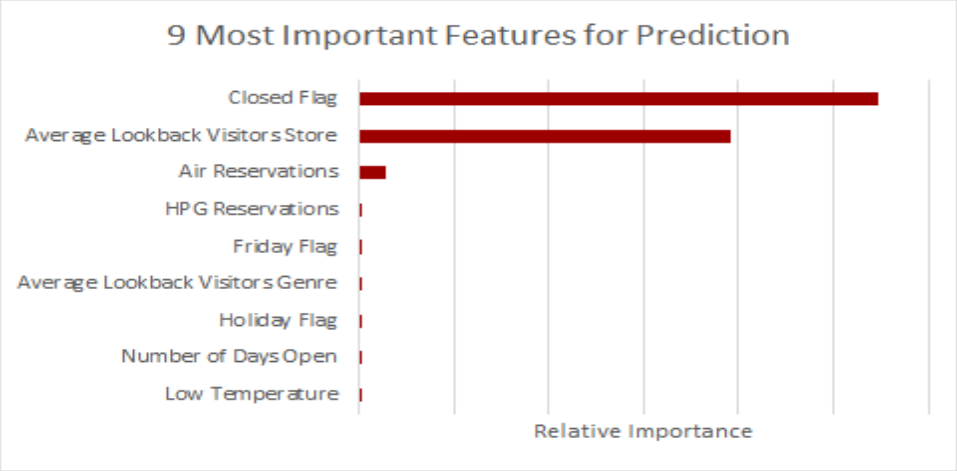

Figure 8: Feature Importance

Recruit holdings can bring these predictions to current and prospective clients to establish strong credibility of their forecasting abilities. Restaurants that utilize this capability will be able to utilize more accurate visitor forecasts to optimize staffing and increase their supply chain efficiency.

Risks

Risks of using this model will naturally increase with time if the data with which the models are trained on is not updated. As time passes, the data generation process for restaurant visitors will inherently change. For our models to capture this change and continue predicting visitors at a comparable accuracy, the training data must be refreshed regularly.

Appendix

Code

1. **data_munging.ipynb** (Data Cleaning)
2. **holiday_&_basic_feature_fcns.ipynb** (Data Cleaning)
3. **lookback_fcns.ipynb** (Feature Engineering)
4. **weather.py** (Data Cleaning)
5. **weather_imputations.ipynb** (Feature Engineering)
6. **static_modeling_&_prep.ipynb** (Modeling)
7. **light_gbm.ipynb** (Modeling)
8. **seq2seq.ipynb** (Modeling)

Link to github repository: https://github.umn.edu/engel746/MSBA6420_finalproj

Data

1. **air_reserve.csv** (contains reservations made in the air system)
2. **hpg_reserve.csv** (contains reservations made in the hpg system)
3. **air_store_info.csv** (contains information about selected air restaurants)
4. **hpg_store_info.csv** (contains information about selected air restaurants)
5. **store_id_relation.csv** (contains selected restaurants that are on both air/hpg systems)
6. **air_visit_data.csv** (contains historical visit data for the air restaurants)
7. **date_info.csv** (holidays in Japan for the period relevant to dataset)
8. **air_store_info_with_nearest_active_station.csv** (matches each air store ID with the nearest weather station in Japan [from Hunter McGushion's weather data])
9. **1-1-16_5-31-17_Weather** (directory containing files with daily weather data from 1663 weather stations in Japan [from Hunter McGushion's weather data])

Resources

https://www.kaggle.com/c/recruit-restaurant-visitor-forecasting

https://www.kaggle.com/huntermcgushion/rrv-weather-data

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

https://github.com/microsoft/LightGBM

https://explained.ai/gradient-boosting/index.html

Kaggle Submission

Submission and Description	Private Score	Public Score	Use for Final Score
Submission_File_Fazel_closed.csv just now by Tabassum Fazel Recruit Restaurant LGB	0.58991	0.56789	☐