Machine Learning Engineer Nanodegree - Capstone Project

Dimas Adinugroho

March 9, 2018

1 Introduction

1.1 Project Overview

Kaggle is a platform for data science competition, where many people trying to solve company's problem by building predictive modeling and produce the best model. This project is one of the kaggle competition that involve many Japanese restaurants and help them predicting how many customers to expect each day in the future. The forecasting won't be easy to make because many unexpected variables affect the visitor's judgment, for example, weather condition, preferences, date, popularity, etc. It will be more difficult when the restaurant only has little data.

Recruit Holdings Company has access to bigger data that could make this forecasting possible. To overcome this, they used data from Hot Pepper Gourmet(HPG) and AirREGI which are company they owned.

1.2 Problem Statement

By using the reservation and visitation history data from HPG and AirRegi, we will predict the total number of visitors to a restaurant for future dates. Our target variable is numerical that has time dependencies. Because we are dealing with time series related problem, the prediction not only depends on other external predictors but also past or historical data.

All of the analysis and steps (from data exploration to model implementation) will be put in one notebooks which has multiple sections for each steps. This project aims to build up a model that can correctly predict the visitors given features and past data using multiple approaches. The goals for this challanges is to forecast the number of visitors for each restaurants (**air_store_id**) from **2016-12-24** (24th December 2016) to **2017-04-22** (22th April 2017).

As stated before, there are several data that provided for this challange. The data comes from two separate companies and other sources:

- Hot Pepper Gourmet (hpg): Japan's gourmet site, here users can search restaurants and also make a reservation online on restaurant in japan.
  - hpg_reserve.csv: This file contains reservations made in the hpg system.
  - hpg_store_info.csv: This file contains information about hpg restaurants (4690 unique restaurant's id) like location, area and restaurant genre.
- AirREGI / Restaurant Board (air): A point of sales system specialized for Restaurant, can be used as reservation control and cash register system.
  - air_visit_data.csv: This file contains historical visit data for the air restaurants. There are about 829 unique restaurant's id and about 25108 rows. The target variables(visitors) that need to be forecast, located in this file.
  - air_reserve.csv: This file contains reservations made in the air system. Note that the reserve_datetime indicates the time when the reservation was created, whereas the visit_datetime is the time in the future where the visit will occur.
  - air_store_info.csv: This file contains information about air restaurants (829 unique restaurant's id) like location, area and restaurant genre.
- date_info.csv: This file gives basic information about the holiday dates in Japan.

- sample_submission.csv: This file shows a submission in the correct format, including the days for which we forecast.
- store_id_relation.csv: This file allows you to join select restaurants that have both the AirRegi and HPG systems.

I come to the following steps to forecast restaurant visitors:

1. Dataset Exploration, understanding the factors that affect customer behaviour (What features do we use to make better model), knowing the data characteristic and finding their relationship.
2. Identify patterns in historical data, the trends and seasonal variation of the time series.
3. Do neccesssary data preprocessing if it must, also train several different ML algorithm and get the baseline score.
4. Get the results and evaluate the models. Find more ways to improve it to make better forecasting.

1.3 Metrics

Every kaggle competition have different quality metrics that used by participants to evaluate their model performance. Restaurant Forecasting competition features Root Mean Squared Logarithmic Error (RMSLE). RMSLE and RMSE used to find out the difference between predicted values and the actual one. RMSLE usually used when we don't want to penalize huge differences in the predicted and actual values when both of them are huge numbers.

The RMSLE is calculated as:

$$\sqrt{\frac{1}{N}\sum_{i=1}^{n}(log(P_i+1)-log(\alpha_i+1))^2}$$

where:

- n is the total number of observations.
- P_i is your prediction of visitors.
- α_i is the actual number of visitors.
- $log(x)$ is the natural logarithm of x

2 Analysis

2.1 Data Exploration

The data comes from two separate sources: HPG and AirRegi. The restaurant's visitors that we need to forecast is only the visitor from AirRegi. However, about 4690 restaurants use HPG and only 829 use AirRegi so there are more information available in HPG. The information that we can extract for both of them are locations, genre and reservation data. The date for each restaurant also vary. We are not going to use unlabeled data for predictions (it is for Kaggle submissions).

The key takeaways from the data exploration for each features:

- In air_visit_data, not all restaurant contains historical visit from 2016-01-01 (1st January 2016) to 2017-04-22 (22th April 2017), but most of them available start on July 2016. This is explain the sudden increase of total visitors that happen on July 2016. Also because we need one day period data, there are some missing value per-day in some restaurants. We may fill the missing values.
- During January-December there was a sharp increase in average of visitors and then after that it quickly fell down. Maybe this is due to people celebrate holiday(e.g christmas) or End of Month and then some restaurant closed because of end of month, caused the chart from went up to down quickly.

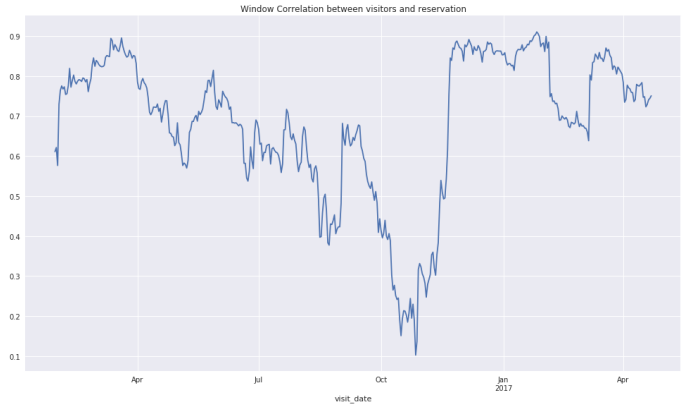

Figure 1: Rolling Correlation between mean visitors and mean reservation

- Interestingly, Tuesday is least busy day for restaurant not Monday. Usually it will increase each day until Sunday where the peak is reached. Monday is ranked at three for the busiest day, followed by Saturday and then Sunday.
- On average, December is the busiest month. For January, August, September and November they are the least busy months. May and June have the least on total visitors because the data is not available as stated before.
- Usually, there are some small peak when holiday comes. People tends to go to restaurant when holiday.
- Total restaurants that available by using HPG is 4690 and for AirRegi about 829 restaurants. By seeing store_id_relation data, only 150 restaurants that use both AirREGI and HPG. But when we merge it into air_store_info and hpg_store_info data, just 63 restaurants that can be explained (see the area, latlng and genre). Also, both HPG and AirRegi have a different way of categorizing the restaurant genre and the latlng is not exactly the same, so we may analyze it separately.
- Japanese style is the most common restaurant on Japan and usually have izakaya style. Asian restaurant (like Taiwanese, Thai, and Shanghai) is rarely seen in Japan but interestingly is the most popular restaurants followed by Karaoke/Party and Izakaya. So Izakaya is not only common but also popular.
- The correlation between mean reserve visitors and mean visitors is 0.59690. We can see from this stats, the reservations value tend to follow the visitors value. If we use rolling correlation rather than just correlation, the correlation between both data is much clearer. Correlation is higher and stable on December-January, it means people tends only to reserve when it comes to holiday.

2.2 Approach

There is some pattern found by using historical data, this means the visitor's data also depends on historical data (time dependent). Because of this properties, there are various steps involved to forecast the visitors. We can only use historical data without another feature (univariate) or even just external features without past values. In general, There are two approaches for forecasting visitors in all restaurants:

- Statistical approach: Forecast per-restaurant only by using trend, seasonal and cyclic pattern in historical data. We can use univariate ARIMA for this kind of approach.
- Machine Learning approach: Forecast by using supervised learning with features like date-related, time-related, location-related, weather, etc.

Before modeling the data, usually, we split the data into training and testing dataset to evaluate models we built. Evaluating time series data is different than what we usually do, this is because

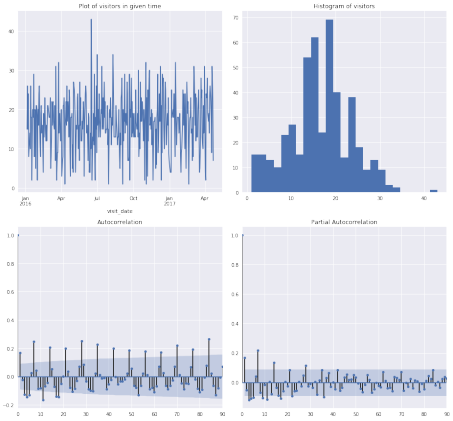

Figure 2: ACF and PACF of random restaurant with no differencing.

we assume that there is no relationship between observation (independent). This is not true for time series because time-series data depends on past data we cannot split data randomly. In order to fix this, we split data and respect the temporal order which the data observed. In time-series, evaluate models based on historical data called **backtesting**.

2.2.1 Statistical Approach

ARIMA models is one the common statistical method that used historical pattern to predict future values. ARIMA stands for **A**utoregRessive **I**ntegrated **M**oving **A**verage. ARIMA models aim to describe the autocorrelations in the data.

ARIMA models in denoted with notation ARIMA(p, d, q), it consist of three component and used to parameterize the models. that is:

- Autoregressive or AR(p), assumes that the present value of a time series can be explained as a function of its past values, the function is constructed using a linear combination of past values of the variable with lag p or dependent on p past values. The term autoregression indicates that it is a regression of the variable against itself.
- Integrated or I(d), compute the differences between consecutive observations. This is known as **differencing**. Differencing can help stabilize the mean of a time series by removing changes in the level of a time series, and so eliminating trend and seasonality.
- Moving Average or MA(q), use past errors variables for forecasting in a regression-like model (like weighted moving average of the past few error). It is dependent on q past error terms or lag q on past error.

The combination of differencing with autoregression and moving average model [HA12], this is called non-seasonal ARIMA model. The full model can be written as:

$$y'_t = c + e_t + \theta_1 e_{t-1} + ... + \theta_p e_{t-q} + \Phi_1 y'_{t-1} + ... + \Phi_p y'_{t-p}$$

When a time series data with seasonal effects, we use another model called seasonal ARIMA or SARIMA, which is denoted as ARIMA(p, d, q)(P, D, Q)s where (P, D, Q) is seasonal parameters and s is the periodicity of the time series.

We sampled random restaurants to get the parameters for ARIMA model. To determine the parameters p, d, q, we need to check ACF and PACF of the series with no differencing.

The first step in fitting ARIMA model is the determination of the order of differencing(d) needed to stationarize the series. for the nonseasonal part, we see that the ACF plot (Figure 1) already showing no correlation between lag. Also there is a seasonal component at the lag 7 (spikes at lag 7). It means that this series contains seasonal component, which is we have to search seasonal parameters also.

If the lag-1 autocorrelation is zero or even negative, then the series does not need further differencing. This means, that the data already stationary and we use 0 for differencing parameter.

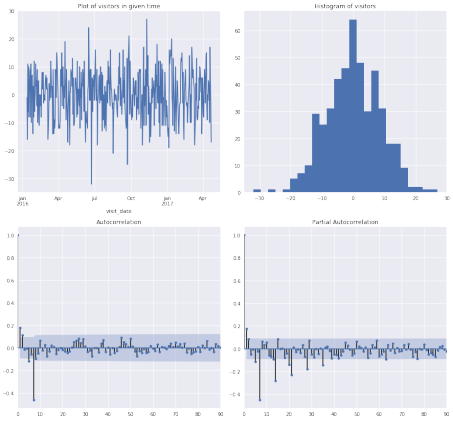

Figure 3: ACF and PACF of random restaurant with seasonal differencing

After a time series has been stationarized, the second step is to determine MA(q) and AR(p) values by looking at the ACF and PACF plot of the differenced series (Figure 2). Because PACF is exponentially decaysing and there is a significant spike at lag-1 in ACF, it means that we don't need to add AR component only MA component. So for nonseasonal part we use (0, 0, 1) parameters.

As stated before, the series has a seasonal patterns, so we need to search seasonal parameters. Because the series has a strong seasonal patterns(Figure 3), differencing values in seasonal parameters should be 1. Never use more than one order of seasonal differencing or more than 2 orderd of total differencing(seasonal + nonseasonal). the ACF plot on seasonal differencing plot has negative value at lag-7, considering adding MA term to the model. So for seasonal part we use (0, 1, 1, 7) parameters.

ARIMA model including **eXogenous** covariates, ARIMAX extends the ARIMAX. The model will including the linear effects of one or more exogenous series. We also can use this with seasonal ARIMA model, become SARIMAX. The *holidays* and *mean_reserve_visitors* columns can be assed to the X of SARIMAX as eXogenous covariates. Because we know that *mean_reserve_visitors* only correlated when holiday, the effect won't be significant. So we only includes *holidays* column as eXogenous variables.

By using only its historical values, the effect of other independent features is abandoned. It fails to capture the effect of factors which are non-seasonal and non-trend in nature like locations and genre. Still, it worth to try.

2.2.2 Machine Learning Approach

Machine Learning can be used for forecasting a time series model. If we want to consider the features that have non-seasonal and non-trend, a statistical approach is harder to use. What makes time series different from normal regression is that they are time-dependent. Time series data have some non-linear pattern in it, for example, seasonal and trends. To make it into regression problem, we need to capture the patterns and include them in the feature.

To include seasonal factors, we can add time-related features to the models by creating dummy variables. Because we know that the series have weekly, monthly and holidays effects, we might add these factors as dummy variables. By using this approach we are assuming the seasonal patterns is deterministic, means it does not change over time.

We don't include the reservation data to the model. Because we are only using mean in reservation data, this won't capture visitation in each restaurant very well. Also, as i stated before, the reservation data only correlated when it near holidays.

Most ML Algorithms do not work with categorical variable, they require all input and output to be numeric. So in order to create dummy variables that can be used in ML Algorithm, we need to encode the data to turn a categorical variable into a numerical variable. There are two standard methods for encoding the data that is Label Encoding and One-Hot Encoding. There us much debates on when we prefer to use one method over another. We use Label Encoding in Prefecture, City and Restaurant genre and One-Hot Encoding in Week, Month, and Holiday. There are several points that we consider to decides the encoding method:

- **Label Encoding**: Give numerical values to different classes. Usually used when data have ordinal relationship or the data have to be ordered because there are relations between them.
- **One-Hot Encoding**: When the data has no ordinal relationship, label encoding can't be used because it can be lead to poor performance so One-Hot Encoding is better choice. One-Hot Encoding will create sparse matrix where one categoric value represented as one column. When they are only two categoric value, Label Encoding is preferable. If they have many categoric values, One-Hot Encoding can create Curse of Dimensionality so we need to be careful when using it.

XGBoost can be used to solve the problem with regression method. It is a popular algorithm and usually used for kaggle competition to achieve state-of-art results [CG16]. XGBoost is implementation of gradient boosting algorithm which is boosting method that use gradient descent to optimize cost function. Like other boosting methods, gradient boosting combines weak 'learners' into a single 'strong learner' in an iterative fashion. [wik]. What makes XGBoost popular is that it extremely fast and efficient.

Usually, many practitioners use XGBoost as black box and run as pre-built libraries thus we do not use all of the advantages of it. In order to reach better accuracy and get the optimal performance, the algorithm needs to be well-configured. The parameters that we should understand are:

- **n_estimators**: Number of boosted trees to fit.
- **early_stopping_rounds**: Find a way to automatically find the ideal value by stopping iterate when the validation score stops improving (deteriorating) after n straight rounds. The better ways to use it, is by setting higher value of n_estimators and use early_stopping_rounds to stop iteration.
- **learning_rate or eta**: multiply each updates of predictions by n value before adding them in. In general, small learning rate will yield accurate models even though it does create more iteration.
- **n_jobs**: When use larger dataset, runtime of creating model is crucial. This will create parallelism to build our model faster.
- There are several parameters that can used to control model complexity:
  - **max_depth**: Maximum depth of a tree, increasing this value will make a model more complex and prone to overfitting.
  - **min_child_weight**: Regularization parameter, minimum sum of instance of weight needed in a child.
- To add randomness to make training robust to noise:
  - **subsample**: subsample ratio of training examples per tree
  - **colsample_bytree**: subsample ratio of columns when constructing each tree.

There are two form of XGBoost:

- **xgb**: XGBoost direct core library, use DMatrix as internal data structure which optimized for both memory efficiency and training speed.
- **XGBRegressor**: Sklearn wrapper for XGBoost library. This allow us to use syntax as sklearn and sklearn's Grid Search method with parallel processing in the same way as xgb did.

I choose relatively small learning_rate, 0.1. Generally a learning rate that can works is between 0.05 and 0.3, so we choose number between these. We can use high n_estimator, the cross-validation and early_stopping_rounds will taken care of this problems. There are many tree-specific that we can tuned, we will use Grid Search method to get the best hyper-parameters. We also can tuned the regularization parameters like lambda and alpha to reduce the model complexity and enhance performance.

3 Result

3.1 Evaluation Metrics

The benchmark we are going to use is untuned Random Forest model, that is RMSLE= '0.787'. This score is easy to beat because untuned Random Forest tends to overfit the training data.

In Table 1, we can clearly see the result for both approaches. By using only historical and holidays, ARIMA can capture better for forecasting visitors rather than Machine Learning approaches. The XGBoost model is quite close to the benchmark model. So even we tuned the hyper-parameters, the score only increase slightly. Thus we need more features that can explained the visitation data rather than tune the model.

Approach	Algorithm	RMSLE
Benchmark	Random Forest	0.787
Statistical	SARIMAX	0.598
ML	XGBoost	0.7412

Table 1: The result of RMSLE

It is not a standard to train ARIMA model over multiple time series with same parameters. Different restaurant generated different time series thus will have different parameters. By using grid search in ARIMA model, the model tends to overfits the training data. So to get more clear picture of the series, manual search is preferable.

4 Conclusion

4.1 Reflection

This project compares different approaches to solving restaurant forecasting. Analyzing the pros and cons each method and which is the best for this kind of problem. However, it was very difficult to decide because there are many other methods that can be used. Also, there are some variables that can be explored further and improved the method accuracy. This suggests the need for more advanced approaches and exploring deeper dependent variables to make a better model that can understand customer behavior and make a better forecast.

4.2 Challenges

To treat time-series into regression problem can be painful. Because it is time-dependent and also have a form of seasonality and trends. It is not enough to capture it by using weekly, monthly and holiday pattern. There are many features that do not include in the models, like weathers, seasons, etc. Independent variables that have same affect for all restaurants is holidays while reservation data is not. So, we still haven't captured the correlation between reservation data and visitation. This will improve the score if we can use it.

To make better prediction by using regression, we can use Grid Search method. Unfortunately this requires substantial amount of resources. On my own PC, it takes about one to two hours to calculate one Grid Search. Because there are many hyper-parameters that can be tuned, it takes patience to get the best hyper-parameters.

There are many another approach that can be used for this challenge, for example LSTM, Prophet and UCM. One of the method that many people recommend is LSTM (Long Short-Term Memory). LSTM Network is a RNN (Recurrent Neural Networks) that is trained using back propagation Through Time and overcomes the vanishing gradient problem. It can be used in Univariate and Multivariate Time Series forecasting problems. Having a few data, this will be another challenge to make the make a better model.

4.3 Improvement

There is a lot of improvement in the future. I already have some in mind for this project:

- For data exploration, we need to explores the features that affect the visitor's behavior furtherer. We can use other data that available outside the challenges like weather and seasons data.
- To get the best of XGBoost model, hyperparameter tuning is important to get maximum performance. 'GridSearchCV' in scikit-learn is the popular option for this. But outside scikit-learn, we can use 'hyperopt' packages that eventually designed for model optimization.
- Most of kaggle competition winners used well-tuned of many models to get a better prediction. We can implement ensemble technique to improve the predictive performance. Ensemble technique uses a new model that combined the predictions from two or more models that already been trained on train dataset. There are many types of ensembles, we can use averaging of predictions or even weighted averaging from multiple models to get better accuracy.

References

[CG16] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *In 22nd SIGKDD Conference on Knowledge Discovery and Data Mining*, 2016.

[HA12] Rob J Hyndman and George Athanasopoulos. Forecasting: principles and practice. 2, 2012.

[wik] wikipedia. Gradient boosting.