

Machine Learning Engineer Nanodegree

Capstone Proposal

Naga Venkata Harisyam Manda

January 6th, 2018

Recruit Restaurant Visitor Forecasting using Stacking

1. Domain Background

Definition of Time Series: An ordered sequence of values of a variable at equally spaced time intervals [\[1\]](#).

Time series are frequently encountered when monitoring industrial data or the data from a process running in time. Time series analysis comprises methods for analyzing time series data to extract meaningful statistics and other characteristics of the data. Time series forecasting is the use of a model to predict future values based on previously observed values. [\[2,9\]](#)

Time series forecasting is being widely used in many fields:

- Sales Forecasting of a product in an Industry
- Budgetary Analysis of a Firm
- Stock Market Analysis
- Yield Projections in a Chemical Plant
- Process and Quality Control in Automated Manufacturing Plant
- Inventory Studies of a Grocery Store or a restaurant
- Workload Projections on employees
- Damage forecasting of a Vehicle fleet
- Forecasting the average price of gasoline in a city each day

Even though it is being widely used, the implementation of Machine learning techniques in time series forecasting is not very widely popular because of its complexity. The inherent time dependence of the features and the components (trend, seasonality etc.) of time series cannot be easily observed [\[3,4\]](#). But in the recent years, due to the development of deep learning models, enabled researchers to use different deep networks for performing time series forecasting. From the knowledge I gained from Machine learning nanodegree, I want to extend it a bit further to do time series forecasting. I also do time series forecasting for my thesis using the conventional ARIMA models and Holt winter methods, but the conventional models are slow for large datasets. Therefore, I want to use machine learning methods like LSTM's and other supervised learning techniques for doing time series forecasting and I would like to check how fast they could be when compared to traditional methods.

2. Problem Statement from Kaggle

I will be participating in the ongoing competition at Kaggle: <https://www.kaggle.com/c/recruit-restaurant-visitor-forecasting>. The problem statement is as follows:

Running a thriving local restaurant isn't always as charming as first impressions appear. There are often all sorts of unexpected troubles popping up that could hurt business.

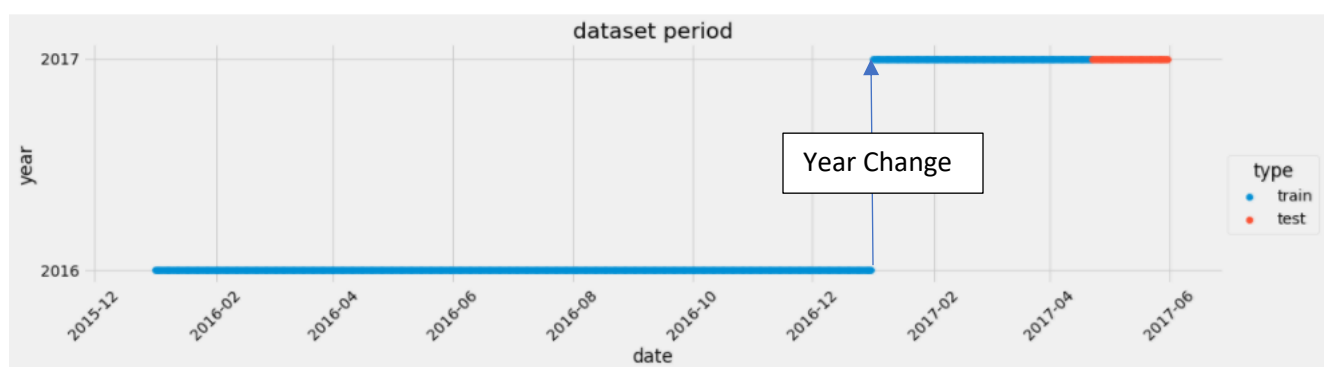
One common predicament is that restaurants need to know how many customers to expect each day to effectively purchase ingredients and schedule staff members. This forecast isn't easy to make because many unpredictable factors affect restaurant attendance, like weather and local competition. It's even harder for newer restaurants with little historical data.

Recruit Holdings has unique access to key datasets that could make automated future customer prediction possible. Specifically, Recruit Holdings owns Hot Pepper Gourmet (a restaurant review service), AirREGI (a restaurant point of sales service), and Restaurant Board (reservation log management software).

In this competition, we are challenged to use reservation and visitation data to predict the total number of visitors to a restaurant for future dates. This information will help restaurants be much more efficient and allow them to focus on creating an enjoyable dining experience for their customers. [10]

3. Datasets and Inputs

The datasets can be found here: <https://www.kaggle.com/c/recruit-restaurant-visitor-forecasting/data>. The target variable in the dataset is the number of visitors for different restaurants. The forecasting must be performed for all the restaurants in the submission.csv found in the link above. The training dataset has data from January-2016 to mid-April 2017, and the test set is from mid-April 2017 to June 2017.



From the training set: as we have time series data it is suggested to use the **Time Series split** [14] for creating the actual training and validation sets. The date range could be as:

Actual Training set: from January-2016 to December-2016 & Validation set: from January-2017 to mid-April-2017. But the proportions of actual train and validation set might vary for the k-fold

cross validation (mentioned in the subsequent section). The test set is already predefined in the competition. The actual training and validation sets are not created by the normal random split because if the random samples are picked as the validation set we might introduce look ahead bias because here the data is sequentially considered [\[13\]](#).

All the information in the dataset will be used to create new features in the feature engineering step. The brief overview of what the file contains, and their respective data formats are shown below:

File: air_reserve.csv

This file contains reservations made in the air system. 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_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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 92378 entries, 0 to 92377
Data columns (total 4 columns):
air_store_id      92378 non-null object
visit_datetime    92378 non-null object
reserve_datetime  92378 non-null object
reserve_visitors  92378 non-null int64
dtypes: int64(1), object(3)
memory usage: 2.8+ MB
```

File: air_visit_data.csv

This file contains historical visit data for the air restaurants.

- air_store_id
- visit_date - the date
- visitors - the number of visitors to the restaurant on the date

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 252108 entries, 0 to 252107
Data columns (total 3 columns):
air_store_id    252108 non-null object
visit_date     252108 non-null object
visitors       252108 non-null int64
dtypes: int64(1), object(2)
memory usage: 5.8+ MB
```

File: air_store_info.csv

This file contains information about select air restaurants.

- air_store_id
- air_genre_name: restaurant food theme
- air_area_name
- latitude
- longitude

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 829 entries, 0 to 828
Data columns (total 5 columns):
air_store_id    829 non-null object
air_genre_name  829 non-null object
air_area_name   829 non-null object
latitude        829 non-null float64
longitude       829 non-null float64
dtypes: float64(2), object(3)
memory usage: 32.5+ KB
```

File: hpg_store_info.csv

This file contains information about select hpg restaurants.

- hpg_store_id
- hpg_genre_name
- hpg_area_name
- latitude
- longitude

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4690 entries, 0 to 4689
Data columns (total 5 columns):
hpg_store_id      4690 non-null object
hpg_genre_name    4690 non-null object
hpg_area_name     4690 non-null object
latitude          4690 non-null float64
longitude         4690 non-null float64
dtypes: float64(2), object(3)
memory usage: 183.3+ KB
```

File: hpg_reserve.csv

This file contains reservations made in the hpg system.

- hpg_store_id - the restaurant's id in the hpg 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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000320 entries, 0 to 2000319
Data columns (total 4 columns):
hpg_store_id      2000320 non-null object
visit_datetime    2000320 non-null object
reserve_datetime  2000320 non-null object
reserve_visitors  2000320 non-null int64
dtypes: int64(1), object(3)
memory usage: 61.0+ MB
```

File: store_id_relation.csv

This file allows me to join select restaurants that have both the air and hpg system.

- hpg_store_id
- air_store_id

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 2 columns):
air_store_id    150 non-null object
hpg_store_id    150 non-null object
dtypes: object(2)
memory usage: 2.4+ KB
```

File: date_info.csv

This file gives basic information about the calendar dates in the dataset.

- calendar_date
- day_of_week
- holiday_flg - is the day a holiday in Japan

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 517 entries, 0 to 516
Data columns (total 3 columns):
visit_date      517 non-null object
day_of_week     517 non-null object
holiday_flg     517 non-null int64
dtypes: int64(1), object(2)
memory usage: 12.2+ KB
```

File: sample_submission.csv

This file shows a submission in the correct format, including the days for which you must forecast.

- id - the id is formed by concatenating the air_store_id and visit_date with an underscore
- visitors- the number of visitors forecasted for the store and date combination

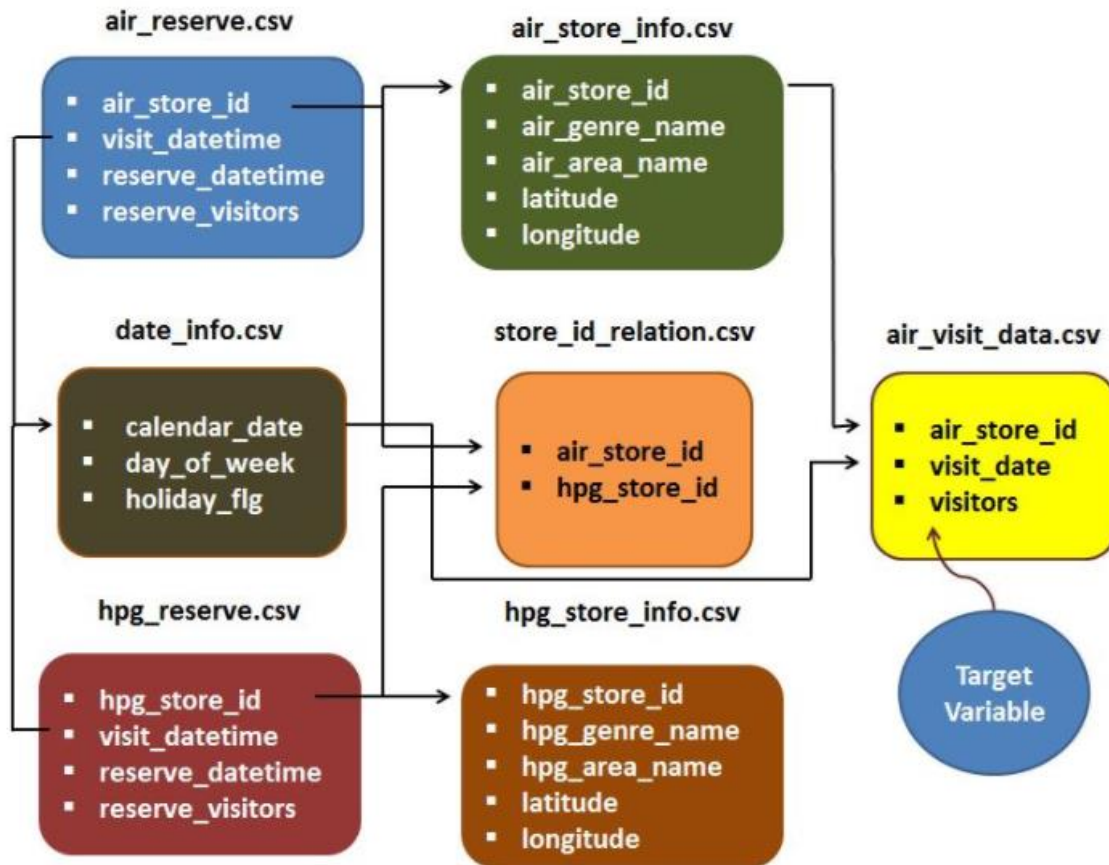






Figure 1: File relations [11]

4. Solution Statement

As mentioned earlier, I would like to use supervised regression models (linear models, SVM's, KNN, ensemble methods) and LSTM's [5] (and MLP) for the forecasting of number of visitors for the restaurants. I will also build up ARIMA models for comparison. At last i would like to stack up all the machine learning algorithms to provide a better result. I would also investigate different possible stacking procedures mentioned in [7,8]. My goal is to reach as high as possible in the leaderboard of the competition.

Leaderboard Ranking: <https://www.kaggle.com/c/recruit-restaurant-visitor-forecasting/leaderboard>

My rank at present is **509/1243** with using lightGBM and minimal feature engineering:

506	▼ 118	Liu Wangsheng		0.492	10	1mo
507	▼ 118	SN		0.492	13	1mo
508	▼ 118	blue0620		0.492	7	10d
509	▼ 117	Venkatesh Sathya Harisyam		0.492	2	3h

5. Benchmark Model

My benchmark model could be an out of box random forests algorithm. I will train it with default parameters present in sklearn, this would be my benchmark and I will try to beat the RMSLE value of it when I generate different variants.

6. Evaluation Metrics

The evaluation metrics used in the competition is **RMSLE**.

Definition of RMSLE is: **Root Mean Squared Logarithmic Error**, which is calculated as

$$\sqrt{\frac{1}{n} * \sum_{i=1}^n (\log(p_i + 1) - \log(a_i + 1))^2}$$

Where:

n is the total number of observations

p_i is the prediction of visitors

a_i is the actual number of visitors

$\log(x)$ is the natural logarithm of x

Lower the RMSLE value, better the forecast is determined by the model

7. Project Design

For this project I will be using:

- Python 3.6.3
- scikit-learn
- Keras with Tensorflow backend
- statsmodels
- H2O AutoML
- Folium and Seaborn

The first step is to obtain data from the competition and extracting them out of the *.rar files.

I will then start with Exploratory Data Analysis (EDA) by using different visualization techniques with the help of seaborn and folium libraries. Based on the conclusions from my EDA, I will create features if necessary (feature engineering). I will check for missing values and impute them if necessary. I will standardize the dataset and split it into training and validation set. The test set is already given as a separate file which will also be standardized as per the training and validation set.

Once the training and validation sets are ready, I will build the supervised regression models. I will optimize the hyperparameters of the regression models by using K-fold cross validation and randomized grid search. As mentioned in section 3, the actual training and validation sets are created by a time series split. For time series the K-fold split of the training and validation sets are generated like the figure below:

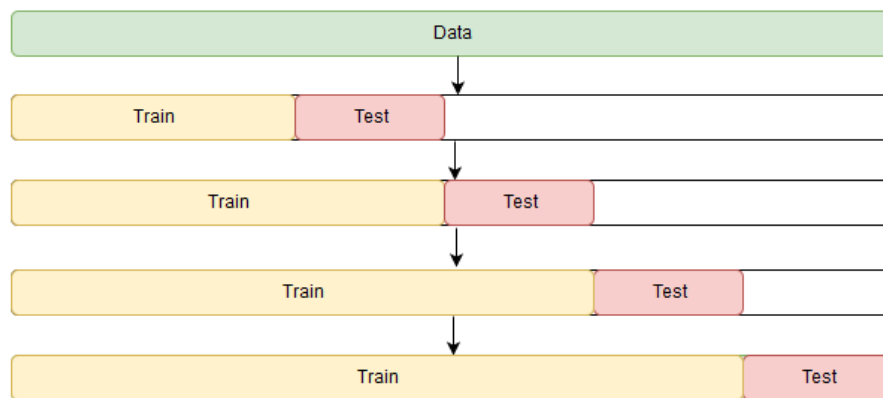


Figure 2: Time series split for k-fold CV [\[12\]](#)

For example, if a 4-fold cv is used then:

1st fold: Actual training: 01/2016 to 12/2016 & Validation: 01/2017

2nd fold: Actual training: 01/2016 to 01/2017 & Validation: 02/2017

3rd fold: Actual training: 01/2016 to 02/2017 & Validation: 03/2017

4th fold: Actual training: 01/2016 to 03/2017 & Validation: 04/2017

The use of randomized grid search for the ML models will help in creating very diverse models which will later help in stacking the models. I will also build up ARIMA model for comparison with the base ML regression models. If the ML models (mainly I will check ensemble methods) are performing poorly when compared to ARIMA model for the test set predictions, then I will revert to feature engineering and data analysis step and will try to get better features.

After finishing the basic ML regression models, I will start building up LSTM with keras and check for their convergence and performance in comparison with ARIMA model. After finishing up with

NN models, I will start stacking the models by using different combinations of meta learners. All the models will be evaluated with the evaluation metric mentioned in section 6. The stacked ensemble will be compared with H2O's AutoML model. For every result from the base learners and the stacked models the leaderboard ranking from the competition is obtained. Based on the RMSLE value (for the test set predictions) from the leaderboard and the running time, the models are ranked accordingly. Finally, the best ranked stacked model is reported out as the final model.

8. References

- [1] <http://www.itl.nist.gov/div898/handbook/pmc/section4/pmc41.htm>
- [2] Imdadullah. "Time Series Analysis". Basic Statistics - <http://itfeature.com/time-series-analysis-and-forecasting/time-series-analysis-forecasting>
- [3] Manisha Gahirwal, Vijayalakshmi M. - Inter Time Series Sales Forecasting,
<https://arxiv.org/ftp/arxiv/papers/1303/1303.0117.pdf>
- [4] <https://machinelearningmastery.com/time-series-forecasting/>
- [5] <https://machinelearningmastery.com/time-series-forecasting-long-short-term-memory-network-python/>
- [6] Christoph Bergmeira, Rob J Hyndmanb, Bonsoo Koob: A Note on the Validity of Cross-Validation for Evaluating Autoregressive Time Series Prediction
<https://robjhyndman.com/papers/cv-wp.pdf>
- [7] <https://github.com/h2oai/h2o-tutorials/tree/master/h2o-world-2017>
- [8] <https://mlwave.com/kaggle-ensembling-guide/>
- [9] <http://home.ubalt.edu/ntsbarsh/Business-stat/stat-data/Forecast.htm#rgintroduction>
- [10] <https://www.kaggle.com/c/recruit-restaurant-visitor-forecasting>
- [11] <https://www.kaggle.com/jeru666/rrv-forecasting>
- [12] <https://stats.stackexchange.com/questions/14099/using-k-fold-cross-validation-for-time-series-model-selection>
- [13] <https://robjhyndman.com/hyndsight/tscv/>
- [14] <https://www.datasciencecentral.com/profiles/blogs/avoiding-look-ahead-bias-in-time-series-modelling-1>