Capstone Project

Machine Learning Engineer Nanodegree

Fred
June 29th, 2016

Definition

Project Overview

This project is originated from the Kaggle competition 'Rossmann Store Sales' in 2015, which can be found in https://www.kaggle.com/c/rossmann-store-sales. The following introduction is extracted from the competition home page:

"Rossmann operates over 3,000 drug stores in 7 European countries. Currently, Rossmann store managers are tasked with predicting their daily sales for up to six weeks in advance. Store sales are influenced by many factors, including promotions, competition, school and state holidays, seasonality, and locality. With thousands of individual managers predicting sales based on their unique circumstances, the accuracy of results can be quite varied.

In their first Kaggle competition, Rossmann is challenging you to predict 6 weeks of daily sales for 1,115 stores located across Germany. Reliable sales forecasts enable store managers to create effective staff schedules that increase productivity and motivation. By helping Rossmann create a robust prediction model, you will help store managers stay focused on what's most important to them: their customers and their teams!"

The project partially uses the dataset from the competition. The original competition has 3 sets of data: a training set with labels, and test set with only features, and a supplementary information set about the Rossmann stores. The test set is used to predict sales for the competition and contains feature-only data for the 6 weeks. The training set contains the historical sales data for 1115 Rossmann stores. The supplementary set contains information about the stores, such as their distance to competitors and promotion information. In this project, only the labeled training set and the supplementary information set are used. In contrast to forecast future sales as in the competition, this project aims to predict sales of some store for some day based on given features, such as the number of customers for the day, whether the day is a holiday, the day of week, the store type, the competitor distance, and promotion information, etc.

Problem Statement

We have the following problem statement based on the introduction in project overview.

Problem:

The problem we are trying to solve is: given a set of features about a list stores over a period of time, predict their sales.

Strategy:

- Data exploration and preprocessing: the data will be examined to determine if there are any missing values and select the useful features as input to modeling. Statistics about the data will also be obtained. The data will be split into training, validation, and testing subsets.

- Develop baseline performance benchmark: select dummy baseline method that the trained model should be benchmarked against.

- Model selection, training, optimization, and evaluation: select candidate models, use training subset to train the model, and use validation subset to tune and optimize the model, and finally evaluate the model using the testing subset.

Anticipated solution:

The problem is a regression problem. From the description above, the input data contains categorical, discrete and continuous features, and therefore we can either try feature engineering techniques like one hot encoding, or we can use models that can handle all three types of features. One promising candidate is tree-based algorithms, which can handle both categorical and continuous features.

Metrics

Predicting sales figure is a regression problem, and the metric used will be root mean square error (RMSE), which is the square root of MSE that has very nice mathematical properties. Besides, it is more sensitive to large errors than MAE, and is a desired property in that we prefer solutions that do not deviate significantly from ground truth.

Analysis

(approximately 2 - 4 pages)

Data Exploration

The data used in this project are train.csv, and store.csv. Description of the features can also be found from Kaggle competition home page.
The train.csv contains 1017209 historical sales data from Rossmann stores, and it has the following features:

Store: Id of the store, 1 to 1115.
DayOfWeek: 1 to 7
Date: as we are not interested in forecasting future sales, this field is not really useful.
Sales: the turnover for any given day. This is what we should predict.
For the entire dataset, the statistics are:

count	1.017209e+06
mean	5.773819e+03
std	3.849926e+03
min	0.000000e+00
25%	3.727000e+03
50%	5.744000e+03
75%	7.856000e+03
max	4.155100e+04

The min sales is 0 as some entries are recorded when the store is closed with no sales.
We are not really interested in these entries and they will be removed from the dataset.
Customers: number of customers on a given day.
For the entire dataset, the statistics are:

count	1.017209e+06
mean	6.331459e+02
std	4.644117e+02
min	0.000000e+00
25%	4.050000e+02
50%	6.090000e+02
75%	8.370000e+02
max	7.388000e+03

Again the 0 customers entries are due to stores not open.
Open: binary indicator on whether the store is open (=1) or not.
Promo: indicates whether a store is running a promotion on that day
StateHoliday: indicates a state holiday: a = public holiday, b = Easter holiday, c = Christmas
SchoolHoliday: indicates if the (Store, Date) was affected by the closure of public schools

Note that we are only going to use the entries when open = 1, as closed stores should always give 0 sales. Part of the train.csv will be reserved as test subset, and the remaining entries are used for training and validation/optimization.

The store.csv contains 1115 entries, each has supplementary information about the corresponding stores, and has the following features:

Store: store Id
StoreType: 4 different store models: a, b, c, d
Assortment: assortment level: a = basic, b = extra, c = extended
CompetitionDistance: distance in meters to the nearest competitor store
CompetitionOpenSinceMonth: approximate month of when the competitor opened
CompetitionOpenSinceYear: approximate year when the competitor opened
Promo2: 0 = store is not participating, 1 = store is participating
Promo2SinceWeek: calendar week when the store started participating in Promo2
Promo2SinceYear: year when the store started participating in Promo2
PromoInterval: describes the consecutive intervals Promo2 is started

For each store, we also add two more features, namely the mean sales and median sales for the store. The mean and median are calculated only based on open stores.

Mean Sales

count	1115.000000
mean	6934.208449
std	2383.911051
min	2703.736573
25%	5322.299969
50%	6589.948470
75%	7964.200644
max	21757.483418

Median Sales

count	1115.000000
mean	6674.630942
std	2314.513962
min	2293.000000
25%	5106.250000
50%	6332.000000
75%	7622.750000
max	21872.000000

Accounting only for open stores, the sales statistics are:

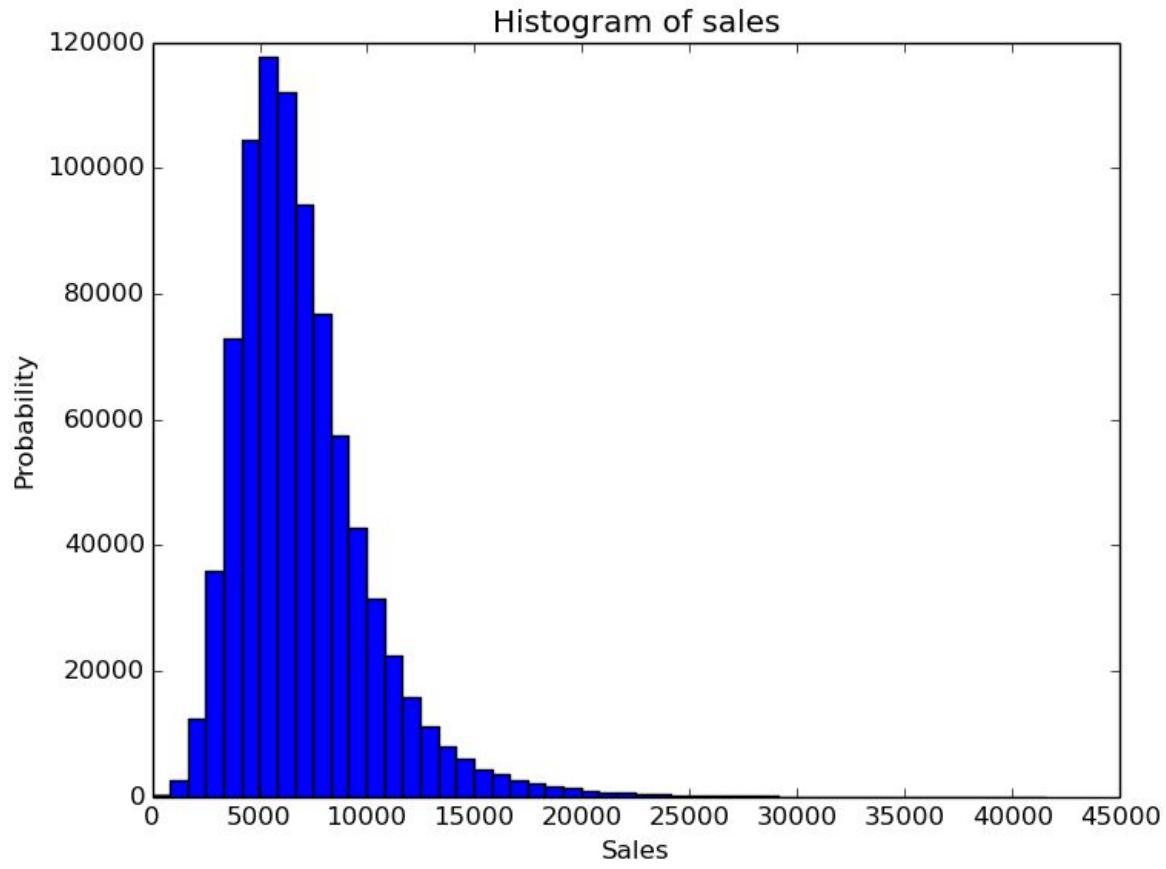
count	844392.000000
mean	6955.514291
std	3104.214680
min	0.000000
25%	4859.000000
50%	6369.000000

75% 8360.000000
max 41551.000000

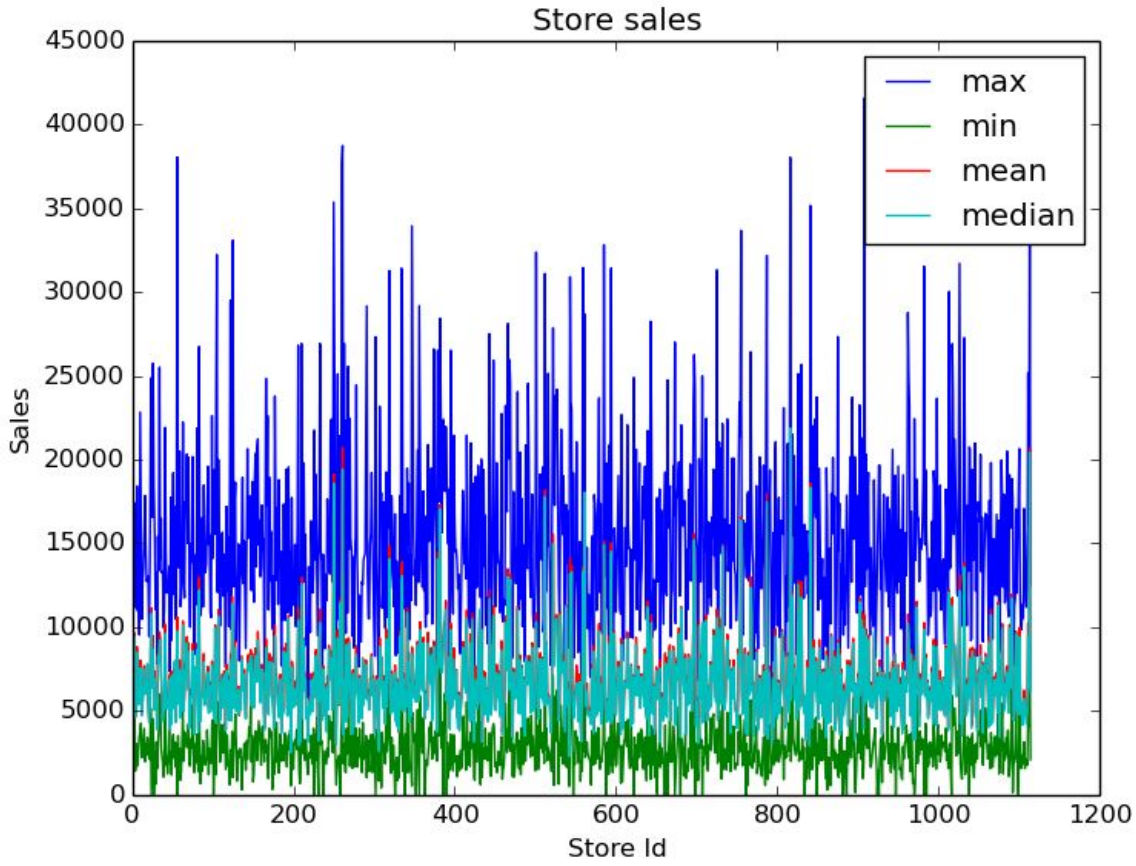
The above three statistics match well.

Exploratory Visualization

First we look at the distribution of sales.

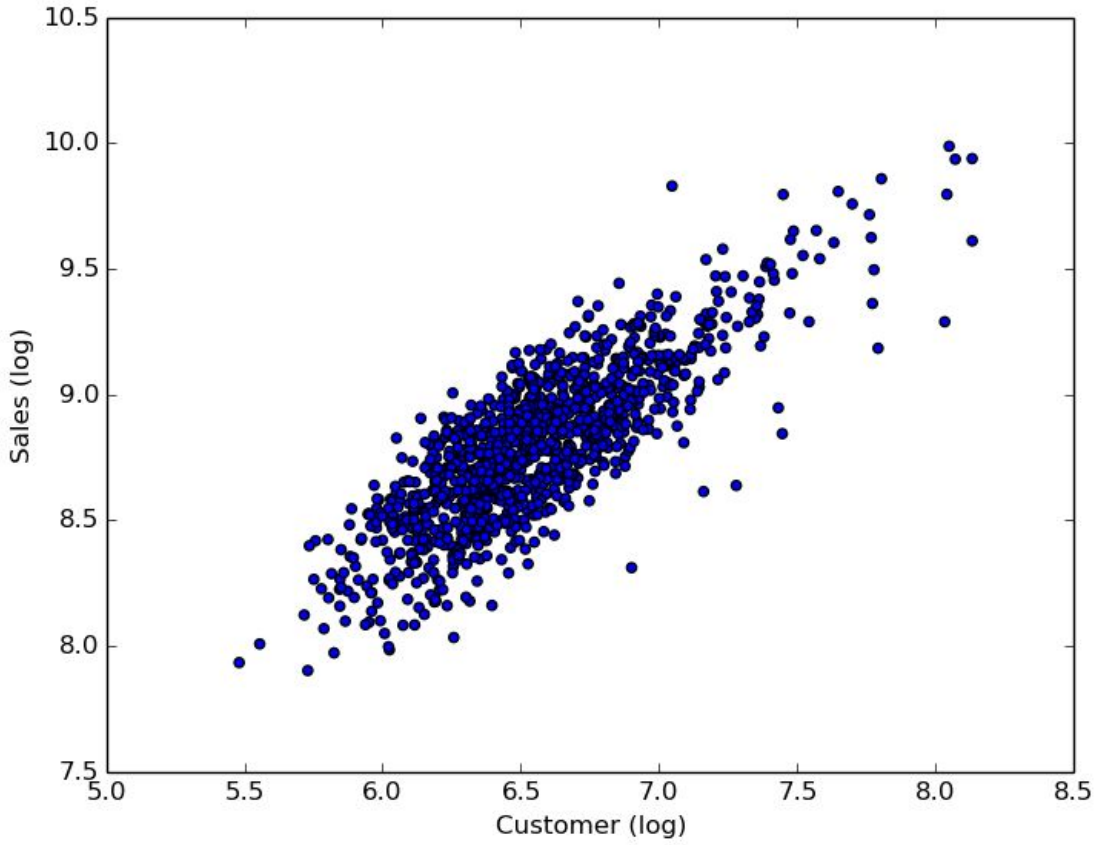


The histogram plot above shows that the distribution of sales is almost balanced without skew, except a slightly heavy tail to the right. This can also be evidenced from the per store sales.



For each store, the minimum, maximum, mean, and median of sales are plotted above, where the maximum of sales are spread further from the centre (mean and median).

It is expected that some features are correlated with the sales. As an example, the average sales per store is plotted against the average number of customers for the store.



As expected, the mean sales increase with increasing number of customers, and there is a clear linear relationship between the log of number customers and the log of mean sales.

Algorithms and Techniques

The decision tree regressor is selected to be the algorithm used. From data exploration, the input data contains categorical, discrete, and continuous types, and therefore tree-based algorithms are suitable. Some of the default parameters that are of interest for the decision tree regressor are:

max_depth=None,
min_samples_split=2,
min_samples_leaf=1,

Benchmark

The per-store mean and median sales are used as benchmark result. The can be obtained from the original data by first grouping all entries for a store and then calculating the mean and median.

Methodology

(approximately 3 - 5 pages)

Data Preprocessing

The preprocessing step is handled by preprocess.py.

The two csv files are merged using store Id as key. Then the merged data is filtered such that closed stores are removed. Initially the merged data has 20 columns (without Sales label column) and 844392 rows, summarized as follows:

```
Store                     844392 non-null int64
DayOfWeek                 844392 non-null int64
Date                      844392 non-null object
Sales                     844392 non-null int64
Customers                 844392 non-null int64
Open                      844392 non-null int64
Promo                     844392 non-null int64
StateHoliday              844392 non-null object
SchoolHoliday             844392 non-null int64
MeanSales                 844392 non-null float64
MedianSales               844392 non-null float64
StoreType                 844392 non-null object
Assortment                844392 non-null object
CompetitionDistance       842206 non-null float64
CompetitionOpenSinceMonth 575773 non-null float64
CompetitionOpenSinceYear  575773 non-null float64
Promo2                    844392 non-null int64
Promo2SinceWeek           421085 non-null float64
Promo2SinceYear           421085 non-null float64
PromoInterval             421085 non-null object
```

Note that some features ('CompetitionOpenSinceMonth', 'CompetitionOpenSinceYear', 'Promo2SinceWeek', 'Promo2SinceYear', 'PromoInterval') has more than 30% of missing values, and in this project these features are dropped, as they are not critical features like number of customers, and can be removed to simplify the model.

In addition, the ‘Open’ feature is not useful any more, as all the filtered data has Opne = 1. The ‘Date’ and ‘Store’ features are also removed.

The textual features ('StoreType', 'Assortment', and 'StateHoliday') are also converted to numerical values, representing categorical index.

The missing values in the remaining dataset are filled with value of -99.

After the above processing, the data is summarized as follows:

```
DayOfWeek            844392 non-null int64
Customers            844392 non-null int64
Promo                844392 non-null int64
StateHoliday         844392 non-null int64
SchoolHoliday        844392 non-null int64
MeanSales            844392 non-null float64
MedianSales          844392 non-null float64
StoreType            844392 non-null int64
Assortment           844392 non-null int64
CompetitionDistance  844392 non-null float64
Promo2               844392 non-null int64
```

Implementation

The modeling part is handled by model.py.

The RMSE is chosen as the performance metric, and it is implemented in performance_metric.Basically, it leverages on the mean_squared_error of sklearn by taking the square root of the MSE.

The input data is split into training and test subset.

The main algorithm evaluates the performance of benchmark using mean sales, the benchmark using median sales, and the performance of decision trees.

A fixed random seed is set globally.

Refinement

Initially the decision tree algorithm uses the default setting, and the following result was obtained.

	Baseline-Mean	Baseline-Median	Decision Tree (untuned)
RMSE	1958.16052196	1983.75902324	710.179313057

The RMSE for the basic decision tree is much smaller than baseline methods.

Grid search is used to fine tune the maximum depth of decision trees. The maximum depth for the optimum model is 20, and the RMSE is 663.15948936, which is 6.6% reduction from untuned model.

Results

(approximately 2 - 3 pages)

Model Evaluation and Validation

Six entries were selected from the original dataset. The model is then trained by 10 different random splits, and prediction are made based on the 10 trained models over the 6 test data point.

The ground truth result is:

Data index	Sales
1	5263
2	5020
3	4782
4	5011
5	6102
6	4364

For the 10 random splits, the following predicted sales are obtained:

Split no. \index	1	2	3	4	5	6
1	4994	4623	4944	4492	5627	4177
2	4610	4610	4868	4908	5609	4118
3	4663	4663	4256	4492	5509	4106
4	4937	4665	4665	5565	5548	3911
5	4634	4634	4813	4492	5442	4078

6	4822	4681	4516	5047	5512	4129
7	4758	4581	4581	4492	5664	3994
8	4678	4678	4434	4492	5482	4069
9	4685	4685	4478	4492	5806	4169
10	4333	4579	4946	5475	5604	4119

The corresponding RSME for the 10 random splits are:

Split no.	RMSE
1	664.4338157651182
2	671.7307360455959
3	662.86766904224783
4	666.28385335214466
5	663.29476853737606
6	663.16899717610283
7	667.94856201625748
8	666.62653289216337
9	659.52082644325674
10	662.3065591744496

From the above result, the RMSE across the 10 random splits is almost consistent, which shows that the model is not sensitive to input data.

Justification

Initially the decision tree algorithm uses the default setting, and the following result was obtained.

	Baseline-Mean	Baseline-Median	Decision Tree (untuned)	Decision Tree (tuned)
RMSE	1958.16052196	1983.75902324	710.179313057	663.15948936

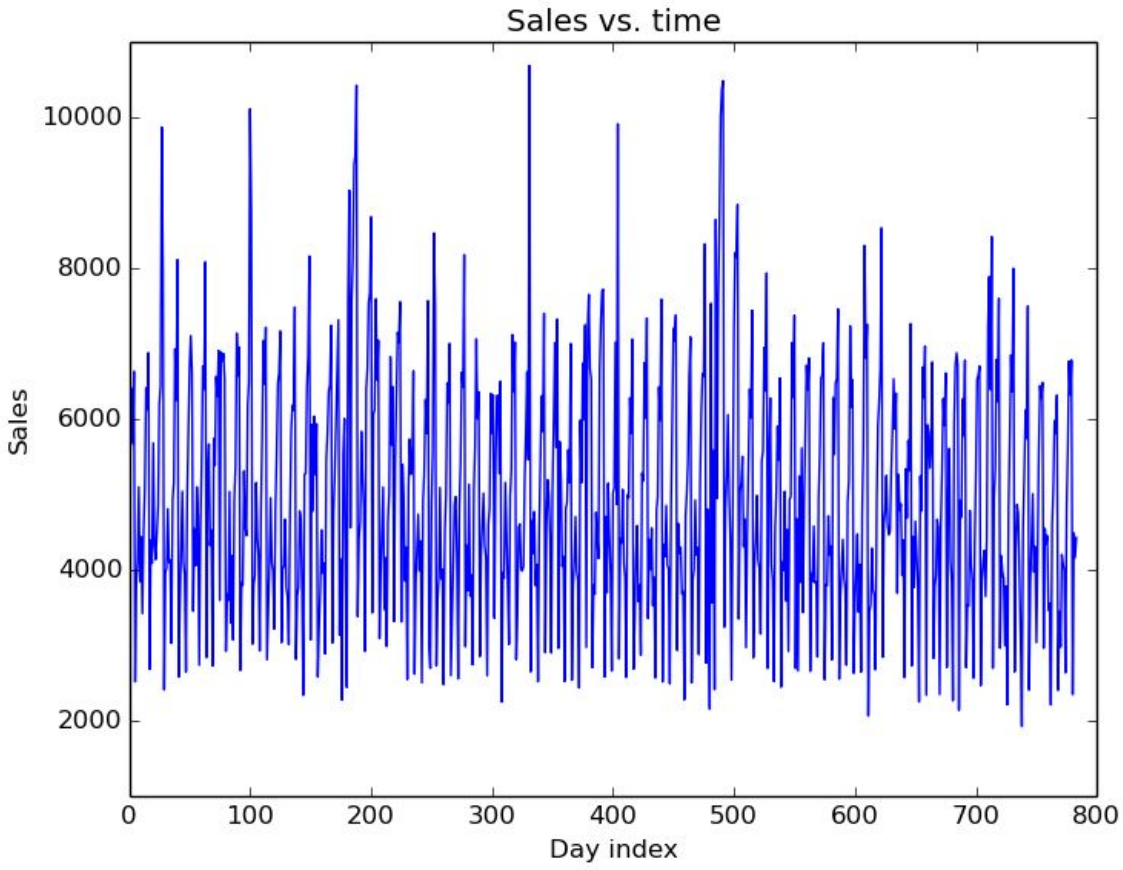
The RMSE for the basic decision tree is much smaller than baseline methods. The tuned model's RMSE is further reduced by 6.6% from the basic untuned model.

Conclusion

(approximately 1 - 2 pages)

Free-Form Visualization

An interesting observation can be made when sale is plotted against the day index: there is a periodic pattern in sales, as shown in the figure below. For some days the sales are significantly higher than the sales on others. This pattern may be explained by the regular shopping behavior that people buy more on weekends for the entire week's supply.



Reflection

Feature engineering may be the most challenging part of the project. The features come from different sources and needs to be combined. There are also missing data in the

original features. The approach adopted is to drop features that have significant portion of missing values. For features with only a few missing values, the missing values are replaced by a dummy flag value.

Improvement

Grid search over larger parameter space:
After tuning, the optimal model has the following parameters:
max_depth = 24,
min_samples_leaf =2,
min_sample_split = 20,
The RMSE for tuned model is 609.079577224, which reduces 14% from the untuned model.

The decision tree easily overfits.
Ensemble methods may be used to enhance tree-based algorithm performance.
Due to resource limitation, ensemble methods such as random forest is not implemented.

Furthermore, the periodic sales pattern discussed above suggests time series based techniques to extract the seasonal factors for sales prediction.

Overall accuracy of the regressor can be improved by techniques such as stacking, which is commonly used in Kaggle competitions, where predictions from multiple regressors are combined (e.g. by linear regression) to give the final prediction. From the result in sensitivity analysis, the predicted price seems to be lower biased. This bias term may also be compensated in the stacking stage.