Rossmann Sales Forecasting

Abstract

Rossmann Stores Sales Forecasting is originated from one of Kaggle's Machine Learning competitions. The task is to predict 6 weeks of daily sales for 1,115 Rossmann stores that are located across whole Germany. In this paper, we will describe how to use existing data set and external dataset in training a forecasting machine learning model for this.

Project Overview

Sales forecasting or future prediction base on time series [1] dataset is common in many business cases. The better predictions can make business more efficient in early planning like stock preparation and human power relocation etc. This project is focusing on predicting a chain medicine beauty store called Rossmann daily sales for up to 6 weeks in advanced. Rossmann stores locate across multiple states in Germany. In this task, it proves 1,115 stores properties data and daily sales record from 2013 to 2015. It turns out that predicting numeric continuous data is a regression [2] task in machine learning area, and also this is a supervised learning [3]. Regression Analysis is to focus on the relationship between dependent variable and one or more independent variables. In machine learning area, there are many regression algorithms that we can use for this purpose, linear regression[4], SVM regression[5] and Decision Tree[6], all of which have their pros and cons.

Project Statement

In this project, predicting daily sales for up to 6 weeks in advanced is a supervised regression problem. By using machine learning models to predict the sales numbers from a set of vary dependent features. The targets that need to be predicted is in test.csv dataset. Ultimately, the task is about data analysis and data prediction.

A regression problem need to be justified in performance, we need some metric methods for evaluating our models performance. For general regression problems, we have several evaluation metrics for those. Like Mean Absolute Error [7], Mean Squared Error [8] and R2 squared [9] etc. In this Kaggle competition, we use Root Mean Square Percentage Error (RMSPE) [10]. In this project our goal is to minimize the RMSPE score as lower as possible. Bench mark score will be top 300 in this Kaggle competition.

I will follow data analysis process to apply in this project, which contains data exploratory, data cleaning, feature engineering, Algorithm selection, and model

fine tuning. In data exploratory stage, I will load the existing data into Python Pandas data structure , and apply some data exploratory function to take a look at the data first, and then I will clean the data as needed, since good predicting model comes from good data. After all these, I will try to use find out the relationship and coefficient relationship among the dataset, and apply feature engineering on this, also the most important step in this project. Then I will apply the selected features to candidate algorithm and compare their performance by metrics. The expectation for this project is the performance metrics as low as 0.117, of course the lower the better. Meaning your prediction is closer to the true value.

Metric

In this project, we will use Root Mean Square Percentage Error (RMSPE) for evaluating the model performance. RMSPE is suitable for evaluating regression problem. Basically, it will calculate how close the perdition value to actual value by putting punishment (scale) on large error. Our model will use SGD [11] method to decrease the value of RMSPE, the perfect situation would be 0 as no errors at all. Here with the formula for RMSPE:

$$\mathrm{RMSPE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{y_i}\right)^2},$$

Where y_i is the actual target value for a record, in this project, it will be the sales of a single store on a single day and y_hat is the predict target value.

Analysis

Data Exploration

In this project, it provides several datasets for competition purpose, and you are also allow to use external dataset as you can post the source where it comes from. I will first try to use the available dataset. There are three available datasets in the project: train.csv, store.csv and test.csv. Train.csv and store.csv are the training data for building the model to predict the targets in test.csv. Train.csv records 1,017,209 rows of different stores daily sales records from 2013/01/01 to 2015/07/31 including numbers of customers, store status and days property as state holidays or school holiday. We find that the data in train dataset is complete without missing data. Looking into the dataset, we find 9 features, amount them, Store, DayOfWeek, Open, Promo, StateHoliday and SchoolHoliday are category data, the rest I will treat them as numeric data. Sales is the one we are going to predict, so I will label the Sales as target for training in the later process.

```
RangeIndex: 1017209 entries, 0 to 1017208
Data columns (total 9 columns):
Store            1017209 non-null int64
DayOfWeek        1017209 non-null int64
Date             1017209 non-null datetime64[ns]
Sales            1017209 non-null int64
Customers        1017209 non-null int64
Open             1017209 non-null int64
Promo            1017209 non-null int64
StateHoliday     1017209 non-null object
SchoolHoliday    1017209 non-null int64
dtypes: datetime64[ns](1), int64(7), object(1)
```

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday
0	1	5	2015-07-31	5263	555	1	1	0	1
1	2	5	2015-07-31	6064	625	1	1	0	1
2	3	5	2015-07-31	8314	821	1	1	0	1
3	4	5	2015-07-31	13995	1498	1	1	0	1
4	5	5	2015-07-31	4822	559	1	1	0	1

In store.csv, it records 1,115 stores properties value which contains store ID, store type, assortment type, competition information from distance to time beginning, promotion activities information. We can find that there are some missing records in this dataset, for some of reasons like absent or non-applicable, which we will handle this later. Amount them, Store, StoreType, Assortment, CompetitionOpenSinceMonth, CompetitionOpenSinceYear, Promo2 Promo2SinceWeek, Promo2SinceYear, PromoInterval, are category date, other are numeric data.

```
Data columns (total 10 columns):
Store                        1115 non-null int64
StoreType                    1115 non-null object
Assortment                   1115 non-null object
CompetitionDistance          1112 non-null float64
CompetitionOpenSinceMonth    761 non-null float64
CompetitionOpenSinceYear     761 non-null float64
Promo2                       1115 non-null int64
Promo2SinceWeek              571 non-null float64
Promo2SinceYear              571 non-null float64
PromoInterval                571 non-null object
dtypes: float64(5), int64(2), object(3)
```

	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	Promo2	Promo2SinceWeek	Promo2SinceYear
0	1	c	a	1270.0	9.0	2008.0	0	NaN	NaN
1	2	a	a	570.0	11.0	2007.0	1	13.0	2010.0
2	3	a	a	14130.0	12.0	2006.0	1	14.0	2011.0
3	4	c	c	620.0	9.0	2009.0	0	NaN	NaN
4	5	a	a	29910.0	4.0	2015.0	0	NaN	NaN

For StoreType, Assorment and PromoInterval, they are represented by character. Value in character is difficult for machine learning model to recognize or even know the importance. So we need to do some preprocessing at later step.

In test.csv, it has 41,088 records that about 1,115 store daily record except sales and customers. There are 11 missing data, we will also need to fix this at later step.

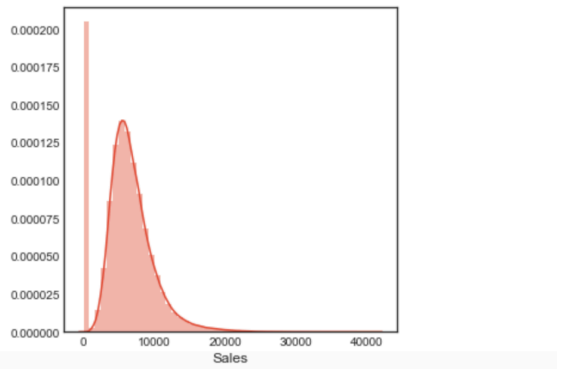
```
RangeIndex: 41088 entries, 0 to 41087
Data columns (total 8 columns):
Id               41088 non-null int64
Store            41088 non-null int64
DayOfWeek        41088 non-null int64
Date             41088 non-null datetime64[ns]
Open             41077 non-null float64
Promo            41088 non-null int64
StateHoliday     41088 non-null object
SchoolHoliday    41088 non-null int64
dtypes: datetime64[ns](1), float64(1), int64(5), object(1)
memory usage: 2.5+ MB
```

	Id	Store	DayOfWeek	Date	Open	Promo	StateHoliday	SchoolHoliday
0	1	1	4	2015-09-17	1.0	1	0	0
1	2	3	4	2015-09-17	1.0	1	0	0
2	3	7	4	2015-09-17	1.0	1	0	0
3	4	8	4	2015-09-17	1.0	1	0	0
4	5	9	4	2015-09-17	1.0	1	0	0

Exploratory Visualization

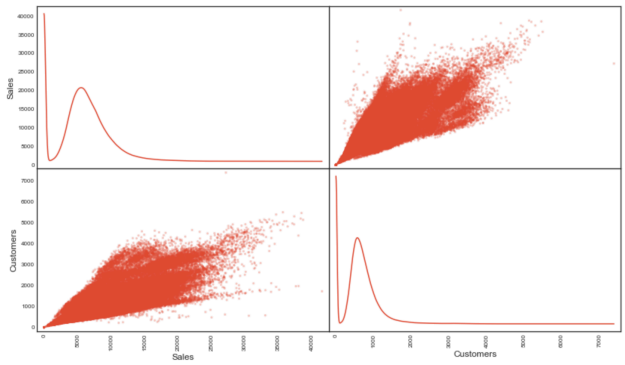
First, I will take a look at the target distribution.

```
<matplotlib.axes._subplots.AxesSubplot at 0x111c6ef50>
```

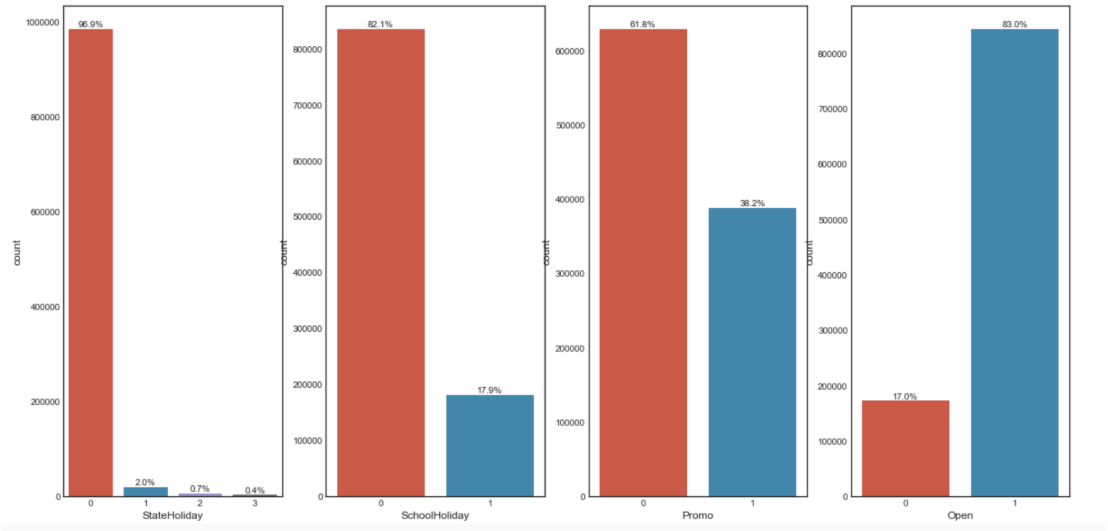


Without data preprocessing, the Sales data has a high skew, high percentage in 0 Sales data, which we need to verify later.

Next, I will check the correlation within dataset. From the diagram below we can see that, Sales have strong relationship with customers, which is true in fact, the more customers the more sales could be.

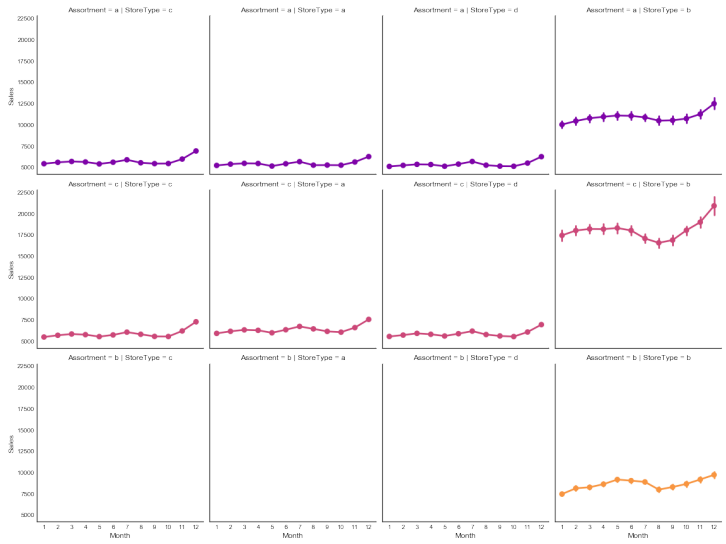


Next, I will take a look at the distribution of different category data. From the diagram, we can interpret that there are few holidays a year, most of the store do promotion, and most of them are open.



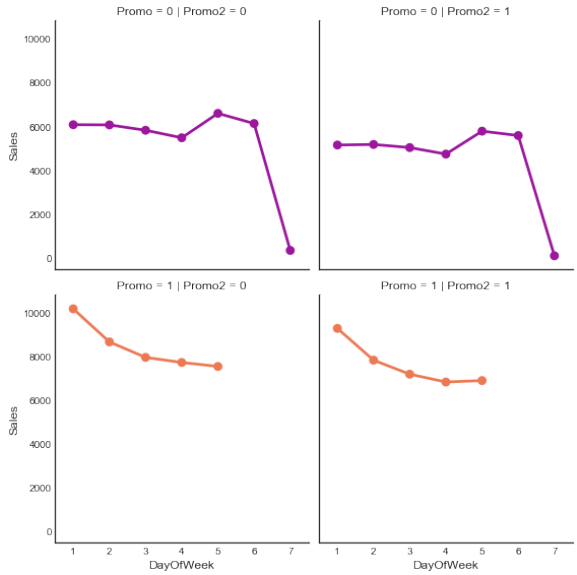
And then I will combine train and store dataset to reveal some relationship between sales and store properties. There are missing value in Competition properties and Promo2 properties due unavailable or purely missing, which we will find out and handle later.

```
Data columns (total 18 columns):
Store                        1017207 non-null int64
DayOfWeek                    1017207 non-null int64
Date                         1017207 non-null datetime64[ns]
Sales                        1017207 non-null int64
Customers                    1017207 non-null int64
Open                         1017207 non-null int64
Promo                        1017207 non-null int64
StateHoliday                 1017207 non-null int8
SchoolHoliday                1017207 non-null int64
StoreType                    1017207 non-null object
Assortment                   1017207 non-null object
CompetitionDistance          1014565 non-null float64
CompetitionOpenSinceMonth    693861 non-null float64
CompetitionOpenSinceYear     693861 non-null float64
Promo2                       1017207 non-null int64
Promo2SinceWeek              509177 non-null float64
Promo2SinceYear              509177 non-null float64
PromoInterval                509177 non-null object
dtypes: datetime64[ns](1), float64(5), int64(8), int8(1), object(3)
memory usage: 140.7+ MB
```



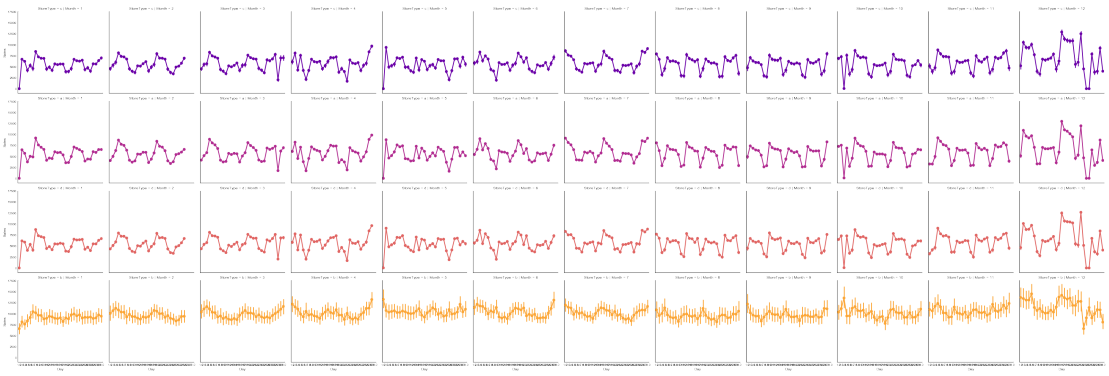
The diagram above shows the relationship amount Sales, StoreType and Assortment, we can find that StoreType b is big different from the others. StoreType b and Assortment c have more Sales than others. The sales trends are similar from those different types of store, which are strong related to datetime.

Next, I will check the relationship about Promo and Promo2, to see how importance they are for predicting the sales.



From the diagram above we can find that, Promo is more important then Promo2, and the fact that Saturday and Sunday will have no Promo. And continuous promo (Promo2) make sales lower.

Let's take a look at the time related diagram. The below diagram shows us the sales in 12 months from different type of stores. We can find out that Store Type B is still higher sales then others. The other types of Store have similar sales pattern. And Sales are varying from different months and days. Except Type B store, other stores doesn't open on Sunday, so they have similar seasoning pattern in sales.



Conclusion:

- There are 4 types of stores and 3 types of assortments; Store of Type B and assortment c is the most selling store and open on Sunday. The other type of stores will close at Sunday.
- Daily Sales have strong coefficient with Daily Customers.
- Promo2 (Continue promotion) has no benefit to increase Sales
- Store type in a,b,c share the same pattern in sales, varying from months to months, days by days.

Algorithms and Techniques

In this project, I apply XGBoost, a scalable end-to-end tree boosting system to train the prediction model. Since XGBoost has been proven in many machine learning challenge to achieve state-of-art result. Even for this project, training sample size is more than 100k, scalable system can have higher capability in learning from a large dataset.

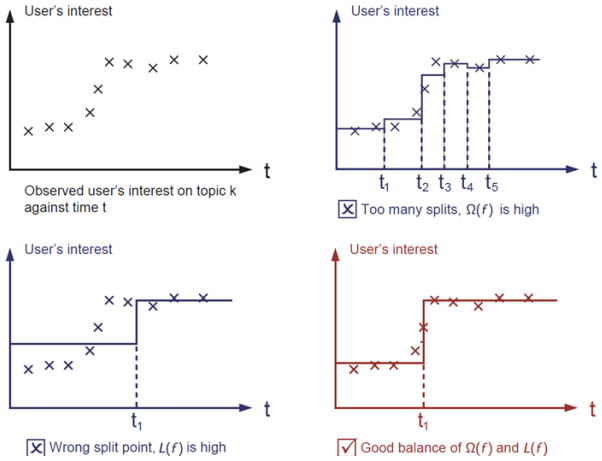
XGBoost is advanced from Tree Ensemble methodology, by assembling multiple simple decision trees model to generate a better result. The tree ensemble model is a set of classification and regression trees (CART) [12]

The advantage of decision tree is that it's adaptable to different kind of data type from numerical to categorical data. Tree process and algorithm can be easily interpreted, and also suitable for nonlinear regression problem. The disadvantage of tree is that it's easily be overfitting, and complexity can will take a long time for training.

XGBoost refine algorithms, and find a balance between variance and bias. Below is the objective function from XGBoost. $L(\theta)$ is the loss function for judging how close the prediction is to the actual value. For regression problem, MSE is normally used in loss function. Actually, XGBoost is suitable for custom loss function, so it's more flexible in model training.

$$\text{obj}(\theta) = L(\theta) + \Omega(\theta)$$

Considering the regularization terms $\Omega(\theta)$ will control the complexity of the model, which will lower the chance of overfitting.



Finally, we need to find out the smallest obj(θ) for the best model.

XGBoost has three types of parameters for above purpose: General Parameters, Booster Parameters and Learning Task Parameters [13]. In this task, we will try to find out the features that best suit to the tree algorithm and tuning the algorithm parameters to make the good prediction model.

In features selection, I will use xgboost build-in get_fscore function for features importance scoring. And since xgboost has evaluation build-in, I will start with few trees to testify importance features, then get into more depth and vast tree models. The whole process is to balance between complexity and generalization.

Benchmark

In the Rossmann competition from Kaggle, the top 330 competitors can reach a score as low as 0.117 in leader board.

So I expect my result will be lower than 0.117 in terms of two points:

- Bias: the ideal model should have public score as low as under 0.117
- Variance: the ideal model should have private score as low as under 0.117 and have close score to public score, meaning that the model is less overffing to the traning dataset.

Methodology

Data Preprocessing

Data cleaning and feature engineering takes up most of time in the whole project. The importance of data preprocessing is out weight than other process, since good data generate good model.

1. Data Cleaning

As I mentioned before that, we found some missing data in data exploration section.

```
Data columns (total 18 columns):
Store                        1017207 non-null int64
DayOfWeek                    1017207 non-null int64
Date                         1017207 non-null datetime64[ns]
Sales                        1017207 non-null int64
Customers                    1017207 non-null int64
Open                         1017207 non-null int64
Promo                        1017207 non-null int64
StateHoliday                 1017207 non-null int8
SchoolHoliday                1017207 non-null int64
StoreType                    1017207 non-null object
Assortment                   1017207 non-null object
CompetitionDistance          1014565 non-null float64
CompetitionOpenSinceMonth    693861 non-null float64
CompetitionOpenSinceYear     693861 non-null float64
Promo2                       1017207 non-null int64
Promo2SinceWeek              509177 non-null float64
Promo2SinceYear              509177 non-null float64
PromoInterval                509177 non-null object
dtypes: datetime64[ns](1), float64(5), int64(8), int8(1), object(3)
memory usage: 140.7+ MB
```

CompetitionDistance and other two Competition related data have some Nan value, and Promo2 related properties also have some missing value. I will fill this nan value with 0 and ‘n’ first as symbol. Since XGBoost is also good at handling missing value.

Most of the data type are good to go with, so next I will do the feature engineering.

2. Feature Engineering

In this part, I will try to transform some features for training.

a) Data Transformation

I pick some feature with large scale in date range like competition distance, sales, customers to transform to lower scale by using logarithm.

Then I transform categorical data to numeric data by Label Encoding, since XGBoost is not capable of dealing with categorical data. They are StateHoliday, StoreType, Assortment .

Date	Store	StoreType	Assortment	StateHoliday
2013-01-01	1	3.0	1.0	1.0
2013-01-02	1	3.0	1.0	0.0
2013-01-03	1	3.0	1.0	0.0
2013-01-04	1	3.0	1.0	0.0

I split date information into several aspects to let the model learn time series data. They are DayOfWeek, DayOfMonth, WeekOfYear, DayOfYear, Month, Year and Season.

Date	Store	DayOfWeek	Day	WeekOfYear	DayOfYear	Month	Year	Season
2013-01-01	1	2	1.0	1.0	1.0	1.0	2013.0	1.0
2013-01-02	1	3	2.0	1.0	2.0	1.0	2013.0	1.0
2013-01-03	1	4	3.0	1.0	3.0	1.0	2013.0	1.0
2013-01-04	1	5	4.0	1.0	4.0	1.0	2013.0	1.0
2013-01-05	1	6	5.0	1.0	5.0	1.0	2013.0	1.0
2013-01-06	1	7	6.0	1.0	6.0	1.0	2013.0	1.0
2013-01-07	1	1	7.0	2.0	7.0	1.0	2013.0	1.0
2013-01-08	1	2	8.0	2.0	8.0	1.0	2013.0	1.0

Then I start two new features about competition during and promotion duration from Competition start date and Promo2 start date.

Date	Store	CompetitionMonthDuration	Promo2WeekDuration
2013-01-01	1	53.0	1.0
2013-01-02	1	53.0	1.0
2013-01-03	1	53.0	1.0
2013-01-04	1	53.0	1.0
2013-01-05	1	53.0	1.0

For training data and evaluation data splitting, I will split out the training, test set in a way of time series. I will use last 2 weeks for testing data for evaluating the model as it's good at practice.

Implementation

By finishing data preparation, I start to build up the XGBoost model for training the dataset.

Starting by setting the model as a little bit high learning rate and less trees for

verification. Since from those parameters setting, I can find out which features are useful in descending the evaluation loss rate. I will use this pre training to find out the useful features for deep study.

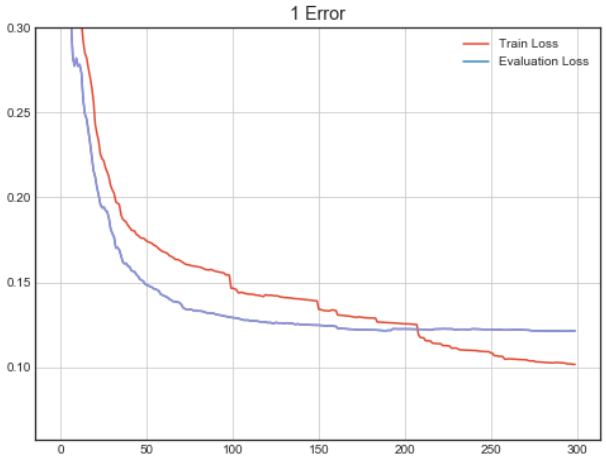
The initial parameters I set is like this: eta: 0.4, max_depth: 8, subsample: 0.8, colsample_bytree: 0.9, min_child_wight: 8 and seed: 31.

I will add the watchlist inside the XGBoost trainer to monitor the evaluation progress, and add early stopping as 100 to stop training if no more better descending. At first I used 300 estimators for test training.

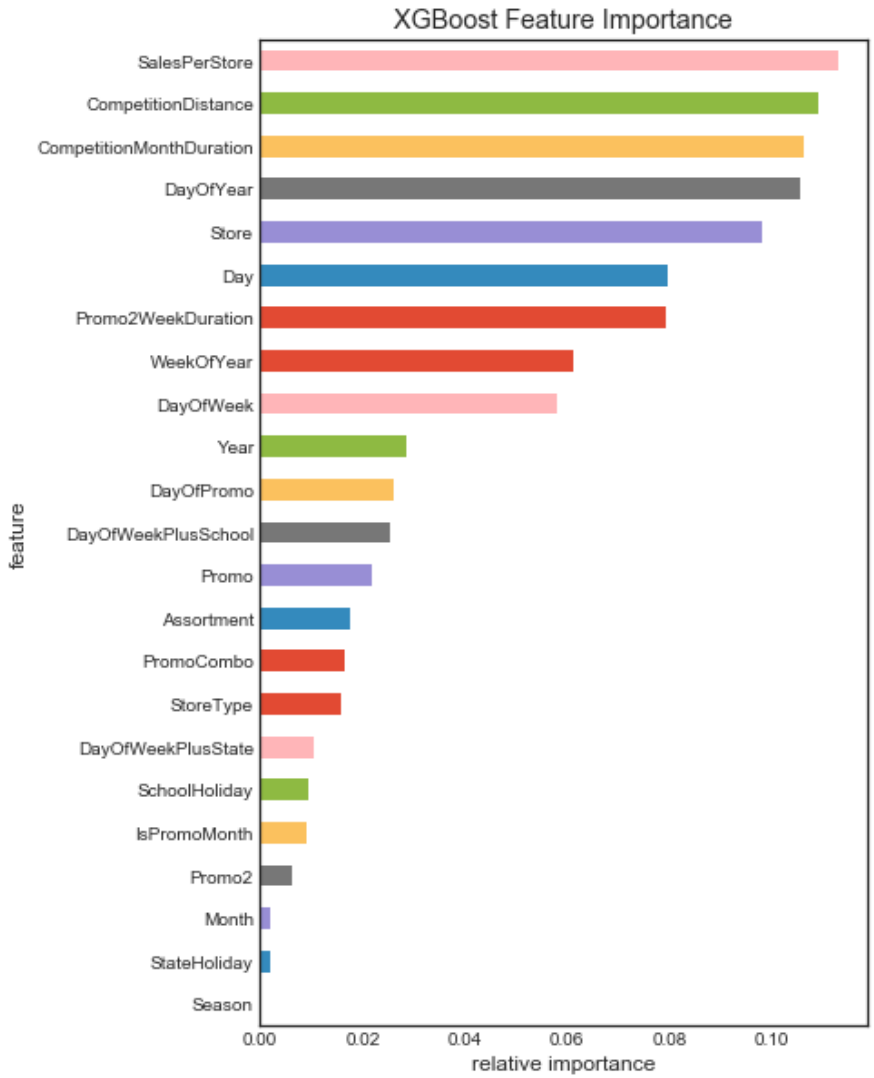
The evaluation score is 0.1215 at training dataset. After uploading to the Kaggle Data Board, the public score is 0.12674 and private score is 0.14387, either of them are lower than 0.117. Meaning that the train, test dataset are actually good to simulate the online test dataset. However, the model is overffiting to the dataset, making the private score is worse.

Private Score	Public Score
0.14387	0.12674

From the training model, we also generated two diagrams for reviewing as below.



The above diagram shows that the evaluation loss has not been dramatically decrease since 150 estimators, and even worse, it's going up, meaning the model is overfitting.



The above features importance diagram shows the importance of each feature during the model building.

From the above diagram, I interpret that XGBoost is better when the features are categorical spread. Some of the features are close, we need to combine them for efficiency.

Refinement

Several refinements strategies I used:

1. Feature Engineering:

I drop some similar features and remain features as CompetitionDistance, DayOfWeek, Open, Promo, Store, WeekOfYear, DayOfYear, Day, Month, Year, CompetitionMonthDuration, Promo2WeekDuration. Some new features I added:

SalesPerStore : History Sales data divide by History Customer data, average for each store.
State [14]: New Feature from external resource, the state each store belongs to.
AssortStore: Combine Assortment and StoreType, generate a new feature with Label coding
DayOfWeekPlusState: Combine DayOfWeek and StateHoliday to generate a new feature
DayOfWeekPlusSchool: Combine DayOfWeek and SchoolHoliday to generate a new feature
DayOfPromo: Combine Day and Promo to generate a new feature
PromoCombo: Combine Promo and Promo2 to generate a new feature

2. Training Sample:

I divide the training sample into server part, and train each part separately, last ensemble the outcome to generate a more robust results.

I divide the training sample into 10 sub-samples, based on Store ID, and 5 sub-samples separately.

I only use the dataset with Sales bigger than 0, since we can find out that when the Store is Close (Open==0), the Sales is 0, so there is no need to train the 0 Sales data.

Small sample can increase training speed and ensemble methods can balance different model results.

3. XGBoost Parameters Tuning.

By confirming the useful features, we can start tuning the XGBoost Parameters for better performance in result.

The changes are: eta->0.01, max_depth->10, subsample->0.3, colsample_bytree->0.8, min_child_weight->8, reg_alpha->1e-04, eastimators->30000, early_stop -> 350.

Lower learning rate can make learner learn deeper but easily overfit, so I adjust subsample and colsample_bytree to constraint the overfit.

Results

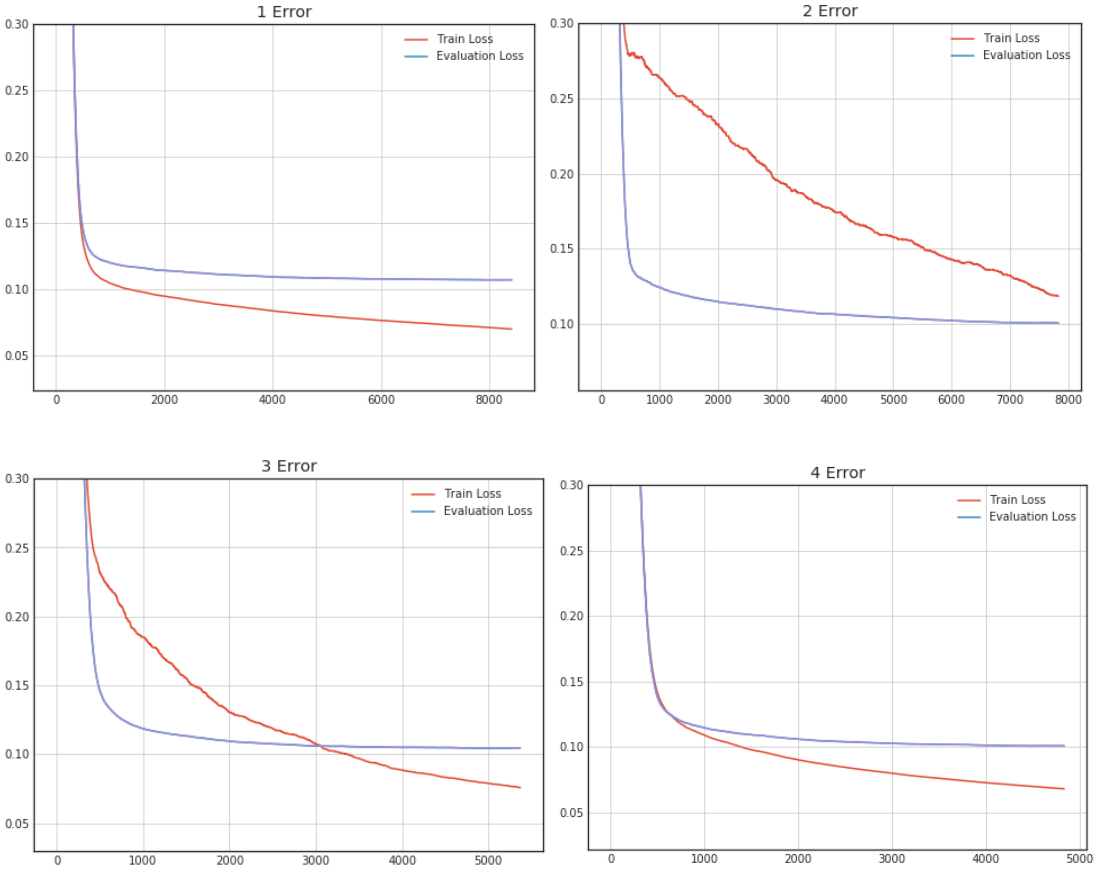
Model Evaluation and Validation

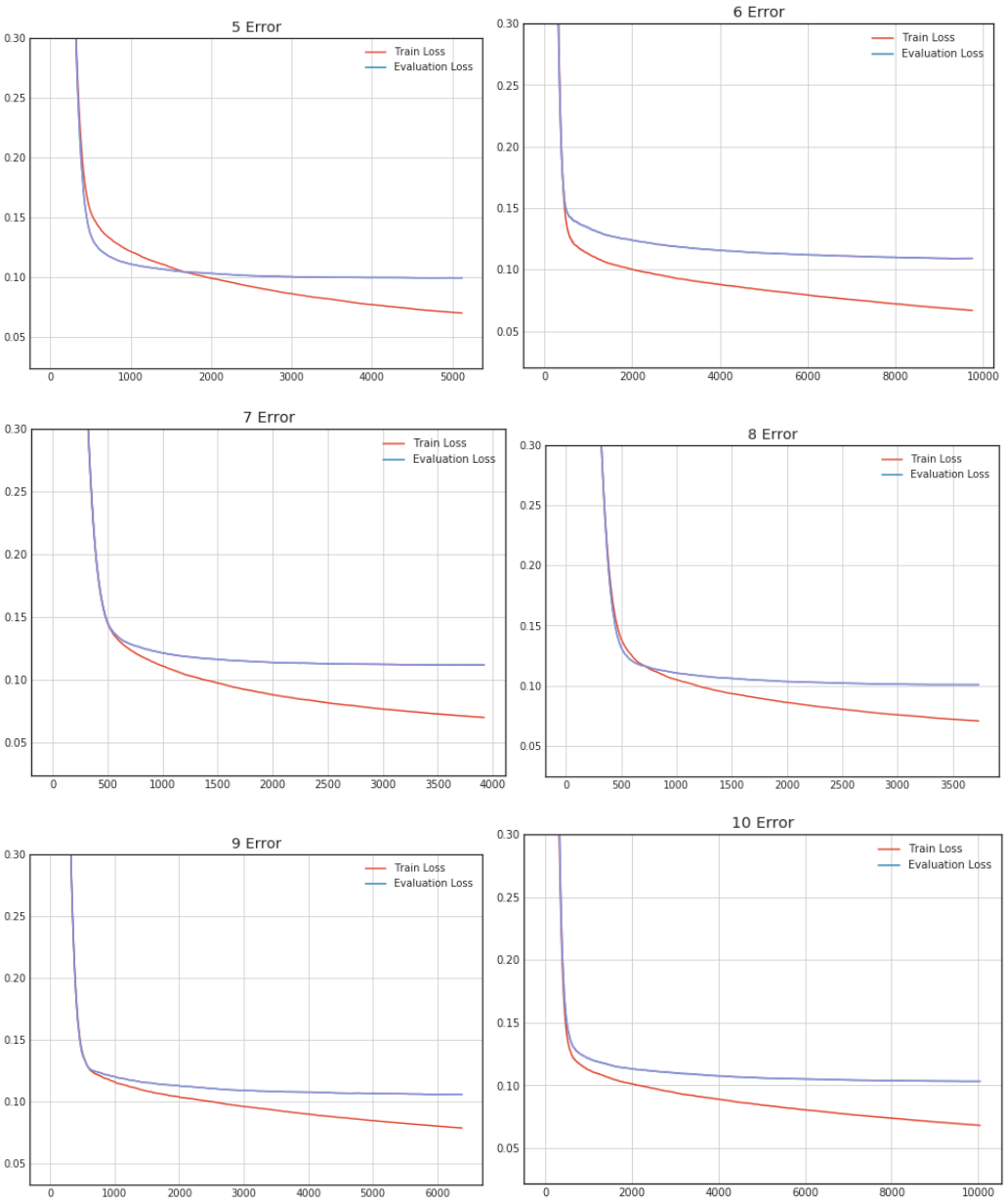
After refinement, I used 10 subsamples and 5 subsamples separately

to train the model and assign trained models in a model list for predicting correspond store daily sales in test set. The result from 10 subsamples is better than full sample. As below:

Private Score	Public Score
0.11728	0.11079

And here with ten different models training performance.





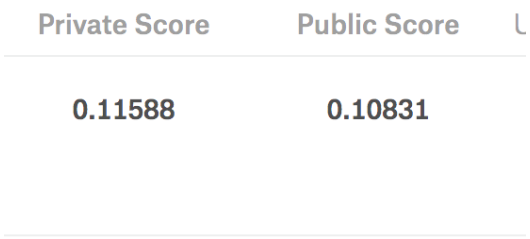
The ten different training subsamples provide stable results, around 0.105. Each subsample contains 111 stores daily transaction records. And I also use the same predictor later for the correspond store prediction, the result is very stable as practice.

Next, I divide the sample into 5 parts, each subsample contains 222 store daily transaction records. And the result as below:

Private Score	Public Score
0.11767	0.11019

It's also very stable for the public and private Score.

By merging these two results, finally get the scores that both are lower than 0.117



The models set are trained for specific set of stores, it's more sensitive to set of stores performance, that will lower the variance for whole set of sample.

Justification

As analyze above, XGBoost performs well in time series regression problems. Without complicated features engineering, it can easily reach score as low as 0.126. With parameters tuning in depth, it can produce more generalize model.

By using subsample and ensemble method in this tasks, diverse performance of prediction has been smoothed.

At the very beginning, I did try tuning the XGBoost algorithm base on the whole dataset, the result had not been lower than 0.12.

Adding model set concept to this problem, different trained model for different set of test dataset is out perform for single model performance.

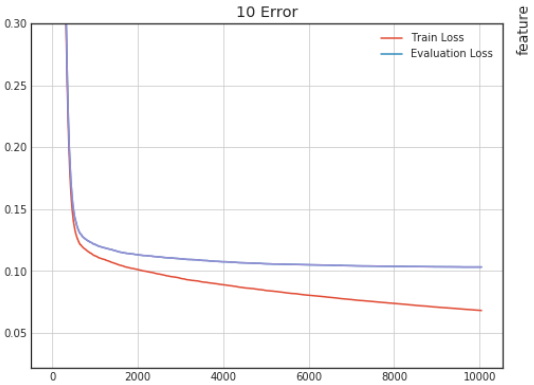
The shortage of this method is difficult to find the balance in splitting the dataset. More splitting take more time in training and ensemble.

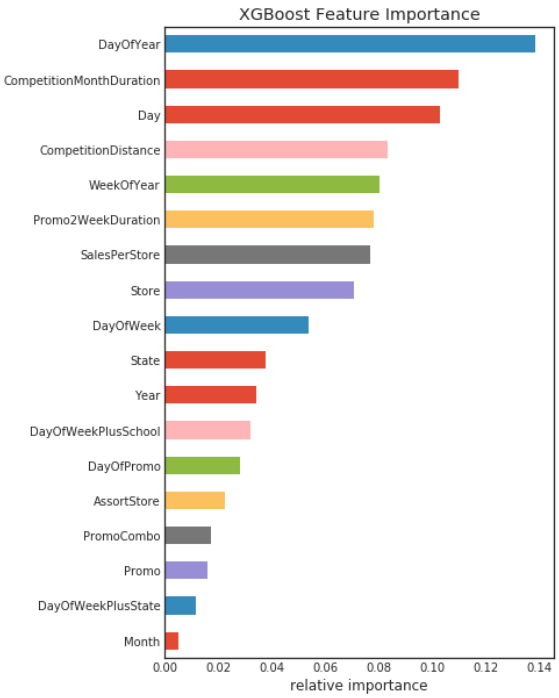
The final score is lower than the 0.117 benchmark in both private board and public board, they are close, so the model is stable and well trained.

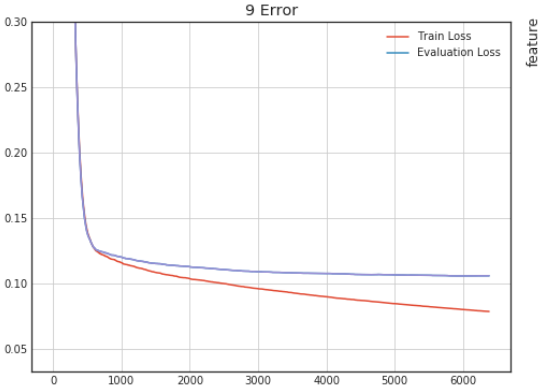
Conclusion

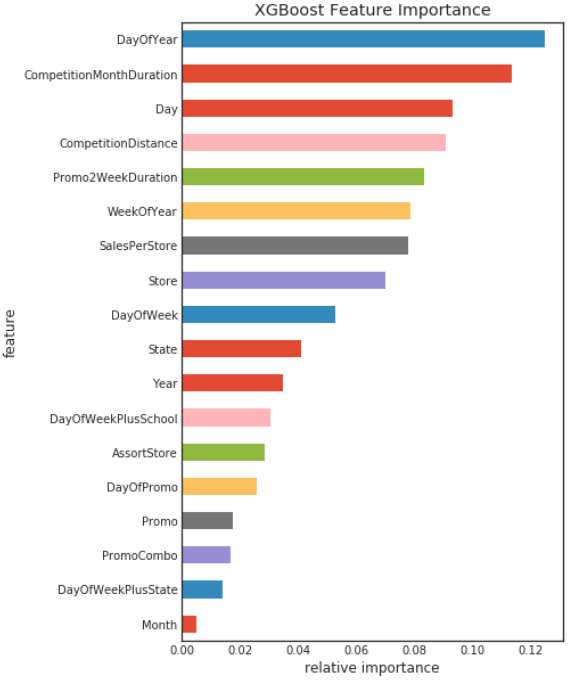
Free-form Visualization

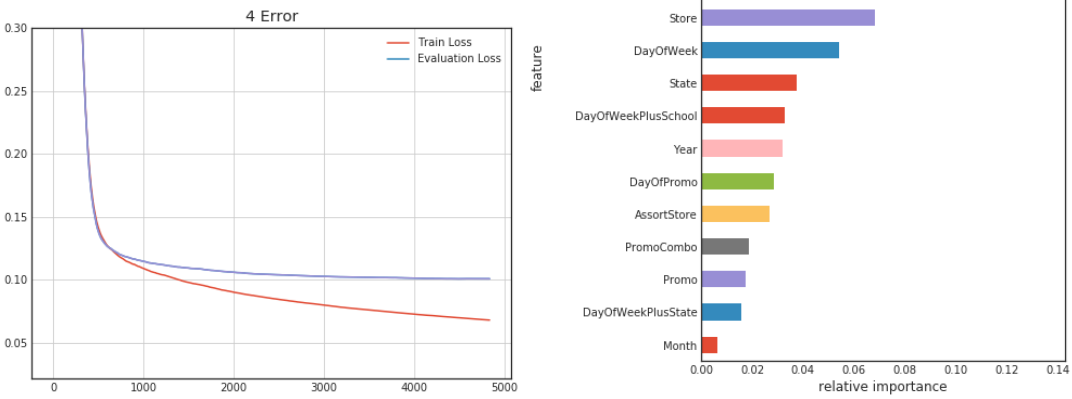
I selected some of the subsample evaluation diagram and features importance showing different models' strength.

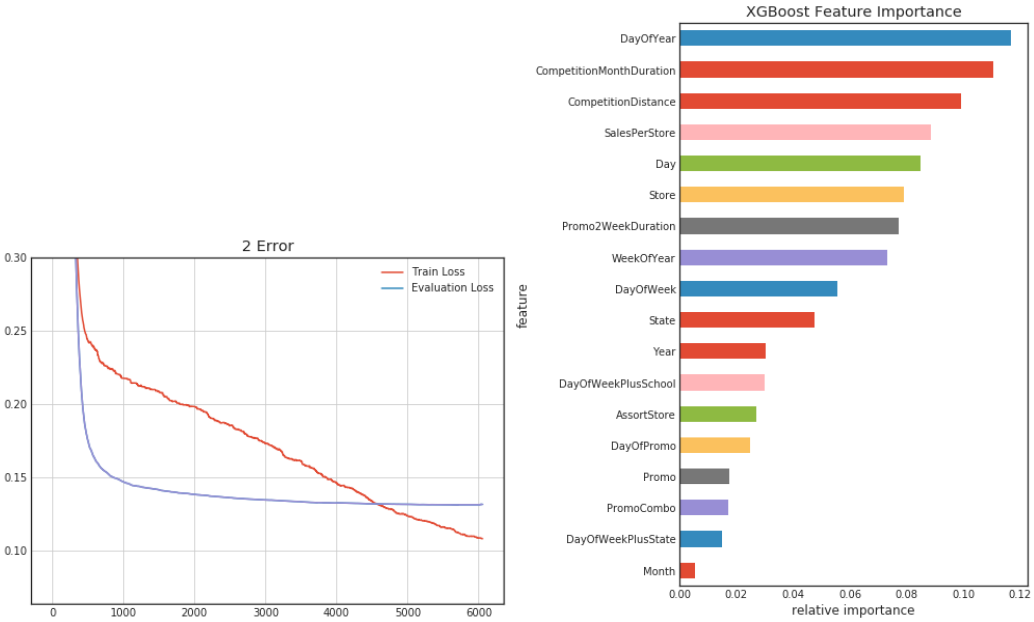

10 Error
Train Loss
Evaluation Loss
0.30
0.25
0.20
0.15
0.10
0.05
0
2000
4000
6000
8000
10000


XGBoost Feature Importance
feature
DayOfYear
CompetitionMonthDuration
Day
CompetitionDistance
WeekOfYear
Promo2WeekDuration
SalesPerStore
Store
DayOfWeek
State
Year
DayOfWeekPlusSchool
DayOfPromo
AssortStore
PromoCombo
Promo
DayOfWeekPlusState
Month
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
relative importance


9 Error
Train Loss
Evaluation Loss
0.30
0.25
0.20
0.15
0.10
0.05
0
1000
2000
3000
4000
5000
6000


XGBoost Feature Importance
feature
DayOfYear
CompetitionMonthDuration
Day
CompetitionDistance
Promo2WeekDuration
WeekOfYear
SalesPerStore
Store
DayOfWeek
State
Year
DayOfWeekPlusSchool
AssortStore
DayOfPromo
Promo
PromoCombo
DayOfWeekPlusState
Month
0.00
0.02
0.04
0.06
0.08
0.10
0.12
relative importance





From the above diagram we can see that different subsamples train a different models that focus on different aspects of features.

Reflection

This project is to predict time series data, other than solely datetime feature, the project dataset also provides other data from domain knowledge like promotion , competition and customers etc, which makes XGBoost suitable.

Although XGBoost is a good candidate for regression model, variance and bias

always need tuning for balancing. All lump-sum samples training is not as good as slicing dataset training in terms of accuracy and speed.

XGBoost is an boosting decision tree as well as tree forest, internal ensemble makes it out perform than single tree performance, I apply the same logic in terms of training dataset in outer space for better application.

Improvement

In this project, I did not consider further feature engineering after reaching benchmark and not even consider further slicing in training set. If I could further slicing the dataset and train model for each store by different aspects of features, the outcome would be better. But the time will take longer, that's the tradeoff.

Also, from some competitors, a technique called entity embedding was applied on this project to achieve better features selection, which I didn't use. XGBoost algorithm is good at categorical data in terms of numerical way, if I could find out more specific feature like special holidays or trends in those years would help improve the whole performance.

Reference

1. Time series https://en.wikipedia.org/wiki/Time_series
2. Regression Analysis https://en.wikipedia.org/wiki/Regression_analysis
3. Supervised Learning https://en.wikipedia.org/wiki/Regression_analysis
4. Linear regression http://scikit-learn.org/stable/modules/linear_model.html
5. SVM http://scikit-learn.org/stable/modules/svm.html
6. Decision Tree http://scikit-learn.org/stable/modules/tree.html
7. Mean absolute error https://en.wikipedia.org/wiki/Mean_absolute_error
8. Mean squared error https://en.wikipedia.org/wiki/Mean_squared_error
9. Coefficient of determination https://en.wikipedia.org/wiki/Coefficient_of_determination
10. Root Mean Square Percentage Error https://www.kaggle.com/c/rossmann-store-sales#evaluation
11. Stochastic gradient descent https://en.wikipedia.org/wiki/Stochastic_gradient_descent
12. XGBoost Tutorial https://xgboost.readthedocs.io/en/latest/model.html
13. Complete Guide to Parameter Tuning XGBoost https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/
14. State csv https://github.com/entron/entity-embedding-rossmann