Predict Future Sales based on Exploring Time-Series Dataset

Yufei Gao(gao,yu@husky.neu.edu)
Northeastern University

Abstract

In my final project, I participated in a competition held by Kaggle and utilized different machine learning models to predict the sales for a bunch of shops for November 2015 based on the historical data from January 2013 to October 2015.
The dataset basically includes daily historical data of the sales in the shops, it also includes some supplemental information of items, categories and shops. To preprocess the dataset, I just clean the dataset in the beginning. To make use of the documental data, I used tf-idf model and do the sentimental analysis and feedback the weight to the model. Then I applied linear regression, random forest, optimized distributed gradient boosting model(XGBoost) and lightGBM to train and predict the future sales.
As the result, I found that the XGBoost Model based on trend featured dataset has the highest accuracy, which has RMSE with 0.90684 and ranked TOP 16%.
In this paper, I will introduce the background of researching on time-series dataset at first. After basic introduction, I will show the mechanism and the implementation of my models. Finally, I will show the result of the project and draw a conclusion of my research.

1. Introduction

Machine Learning is the fastest growing field in the world which relative algorithms launched and developed every day. Thanks to the rapid development of supervised learning, I could tackle large bunch of data and do more complicated research on traditional dataset.
Most of datasets in the competition are kindly provided by 1C company. It contains 1 training set, 1 test set and 3 supplemental forms which provide documental supplement.

2.Problem Discussion

2.1 Trend time-series data

Since the dataset I use is time-series, one way to tackle this kind of data is to figure out the trend. I will compare the result of prediction based on the both dataset.

2.2 Lots of documental data

After research, I found that tf-idf could extract feature from documental data, and balance the weight. I will try to predict the result based on preprocessed dataset with some models.

3. Related works

This competition is a classical prediction problem on time-series dataset. Previously, people conducted many researches in this area concerning data preprocessing and predicting. I referred some of the effort and tried to improve the work:

3.1 Linear regression

Linear regression is commonly used in analysis and prediction problems. Initially, I use this model as simplest case and figure out linear relationship between the forecast variable and a single predictor variable.

3.2 Random forest

Random Forest is an improvement of Bagging ensemble learning method. It uses a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features.

3.3 XGBoost

XGBoost initially started as a research project by Tianqi Chen as part of the Distributed (Deep) Machine Learning Community (DMLC) group, it is an implementation of gradient boosted decision trees designed for speed and performance that is dominative competitive machine learning.

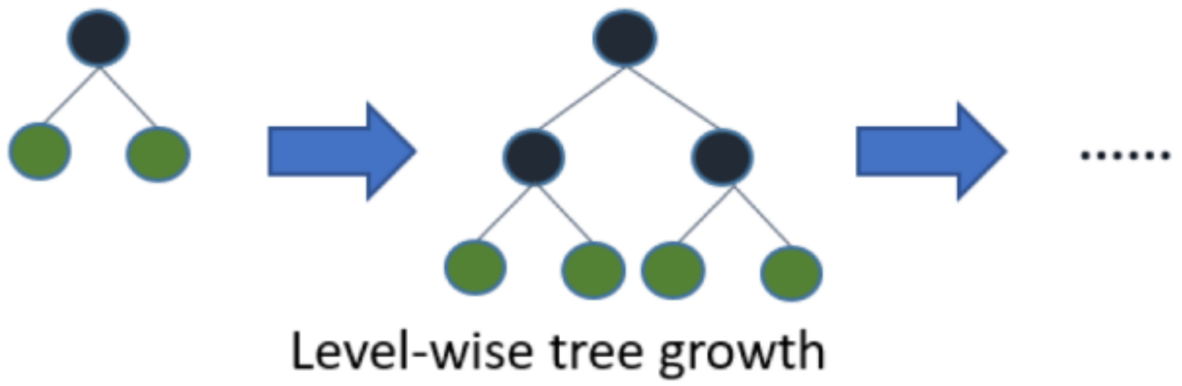

Figure1: leaf-wise growth strategy used in lightGBM

XGBoost implemented level-wise growth strategy. It can split leaves in the same level simultaneously and optimize the model in multi-threads. However, take different leaves in the same weight would effected by noises and add the cost of training.

3.4 lightGBM

Light GBM is an open source and high-performance framework based on decision tree algorithm, which was introduced in LightGBM: A Highly Efficient Gradient Boosting Decision Tree by Guolin Ke. Initially it was designed for Gradient Boosted Decision Tree, it also supports other models such as Goss and random forest.

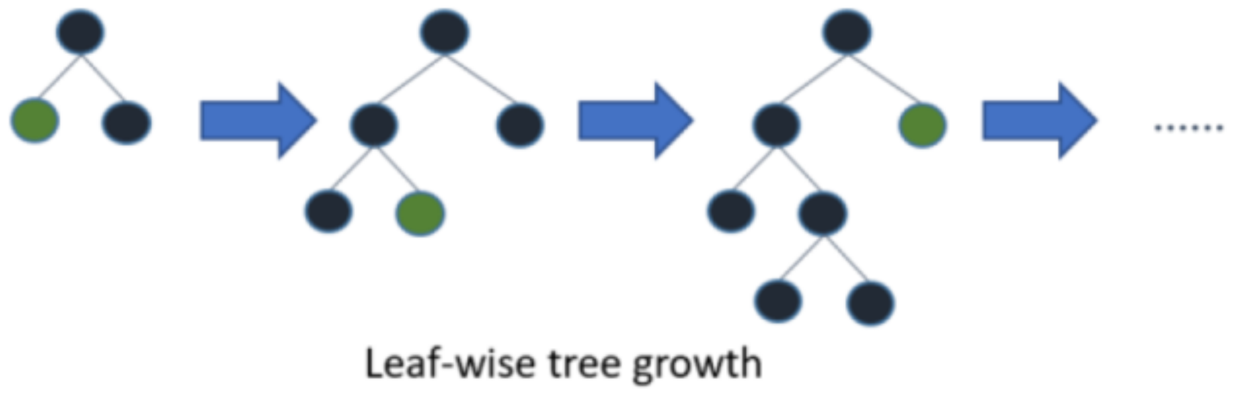

Figure2: leaf-wise growth strategy used in lightGBM

Compare with the level-wise strategy by XGBoost, the LightGBM use leaf-wise strategy. It will split the most valuable leaf and fit the dataset. In this way, the model could save spends and fit the dataset faster. However, it will also lose some information in other leafs. As the decision tree goes deeper, the overfitting issue will happen.

3.5 tf-idf model

As a short for the word "term frequency–inverse document frequency", this method could bring out a numerical analysis which reflect how important a word to the dataset in specific collection. For instance, the word "as" appears more than the word "phoenix" in the corpus, which means in most documents the word "phoenix" could reflect more features than the word "as". To adjust the unbalanced fact, the model will give more weight to the less appeared words, figure out the feature and reflect to the dataset.

3.6 trend features

Another way for feature engineering is to figure out the tendency of numerical variables in the dataset. Since I am exploring a time-series dataset, item price and the sold number changes constantly, so figure out the tendency should make sense for such dataset.

4. Implementation

4.1 Preprocessing – aggregate

After cleaning the dataset, I start aggregate the dataset since the data is daily and I need monthly data for analysis. I aggregated the data based on the requirement of the model and regroup them for convenience.

4.2 Preprocessing – tf-idf

For those models predict based on tf-idf dataset, before data aggregation, I will implement tf-idf model before data aggregation.
I trained three documental files separately with TfidfVectorizer in sklearn package. For each training section, I set 25 features, count the length of the data and the word count, and use tf-idf model to return the weight for the shop, item and item category. After preprocessing, I just joint the dataframes and use different models to predict the future sales with weighted dataset.

4.3 Preprocessing – trend feature engineering

Inspired by one public kernel, for those models predict based on trend featured dataset, I will draw the tendency of the dataset after aggregation. Since both preprocessing methods could generate huge matrix, I will not implement two methods simultaneously. In some cases, I will also only consider the most important variables to make the prediction.

To figure the tendency, I will combine shop and item as a special id, and find out the tendency about the sales of the combination. I will figure out the tendency of item sold, item in each shop sold, the type, price and revenue etc., for some important variable I will have the tendency in different periods, including 1 - 3 months, 6 months and 1 year. I also add the date the item and the item in specific shop first and last sold.

After that, I will use different models to predict the sales in the incoming month based on the preprocessed dataset.

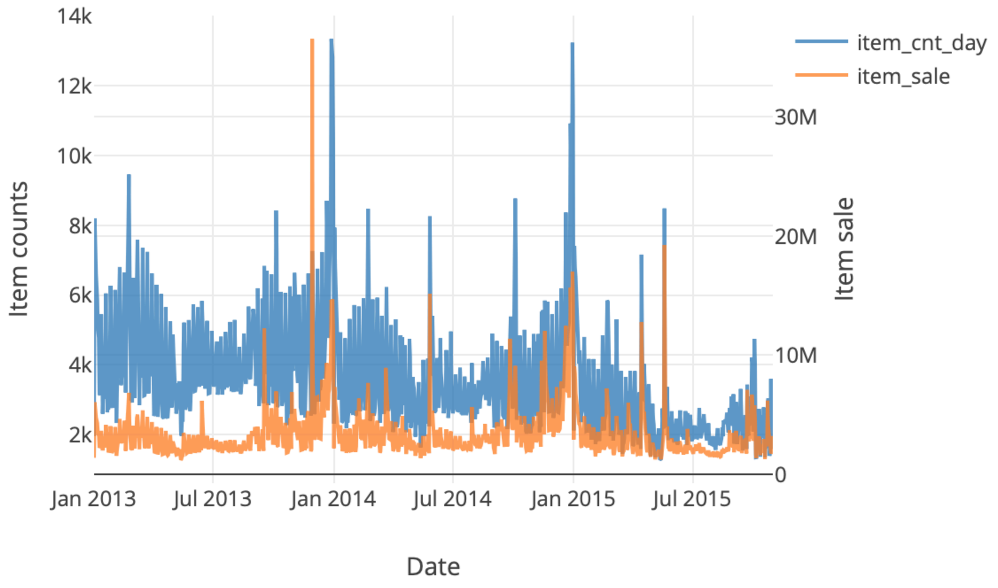

Figure3: tendency of the item sales

4.4 Models

In this project, I implemented Linear Regression Model, Random Forest Model, XGBoost and LightGBM in different situations. Since I have limited computing resources and time, I just set the parameters based on the regular value and try to edit them by crossing validation in some cases. Also, since the limited computing resource, I set the parameters relatively easy to run and those could increase as I get better devices.

4.4.1 Linear Regression

I implemented LR model from sklearn package directly based on original dataset and the tf-idf weighted dataset, and predict the result separately.

4.4.2 Random Forest

Similar to LR model, I just implement the Random Forest Model from sklearn Package directly and predict the result based on two kinds of dataset. I also tried to ignore some unimportant variables and predict the result, see if I could get better result in this way.

4.4.3 LightGBM

I used LGBMRegressor from lightgbm package and I set the regression model relatively shallow. Its max depth is 8 and have 200 estimators, I also set the learning rate slight high as the model needs to fit the data in limited round.

4.4.4 XGBoost

I used XGBRegressor from xgboost package. Similar to LightGBM, I set the model shallower than regular since the limitation of computing resource. I also set a early stopping mechanism to stop earlier than design if the model doesn't improve in some rounds.

5. Result

I submitted various results to Kaggle and the result are listed below.

Method	RMSE(the lower, the better)
Linear Regression	1.16
Linear Regression with tf-idf	9.23
Random Forest	1.20
Random Forest with tf-idf	1.23
Random Forest with most important variables	1.21
Random Forest tf-idf, important variables	1.22
LightGBM	1.06
LightGBM with tf-idf	1.20
LightGBM with trend features	0.96
XGBoost	1.13
XGBoost with tf-idf	1.19
XGBoost with trend features	0.90

6. Conclusion

6.1 tf-idf function for this dataset

According to the result, it seems the tf-idf have negative effect on the result. It not only occupied many computing resources but also make the prediction worse – especially for the linear regression, the result is ridiculous. It happens probably because such function generated a lot of noise and I put it into the model and effect the result.

6.2 figure out the tendency

Apparently, this method make sense for the time-series dataset. In this way I could get rid of the noises generate by data long before, and predict those kinds of dataset more preciously. However, the possibility of overfitting also increases because the training pool goes shallower and more easily influence by parameters.

6.3 compare between different models

In conclusion, for such time-series dataset, linear regression also make sense to some extends. Random forest could handle much more different variables but it performs not good because of unweighted leaves. LightGBM fits the training set faster than XGBoost and occupied less computing resource, but as valuable variables increase and the difference of value between leaves decrease, LightGBM will also lose some accuracy as expense.

7. Reference

[1] https://petolau.github.io/Ensemble-of-trees-for-forecasting-time-series
[2] https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc
[3] https://www.kaggle.com/dlarionov/feature-engineering-xgboost