Capstone Project Report

Xuemei Gao

December 9, 2017

1 Definition

1.1 Project Overview

This project is a Kaggle challenge launched on Oct. 20, 2017. Here is the website of the competition: https://www.kaggle.com/c/favorita-grocery-sales-forecasting.

Sales forecasting estimates future sales, which helps to make business decisions, such as planning stocking amount, allocating resources, predicting sales revenue, managing cash flow, and so on. In retail industry, especially grocery retail industry, sales forecasting is critical, as they may get stuck with perishable goods if overstocked, or lose customers if understocked. Corporación Favorita, a large Ecuadorian-based grocery retailer, owns 54 stores in different cities/regions in Ecuador. It has this challenge on Kaggle in order to have a better prediction of product sales.

As a research topic, sales forecasting has been studied for a long time. The forecasts are often based on past sales data, industry-wide comparisons, and economic trends. (ref: https://trackmaven.com/marketing-dictionary/sales-forecasting/) Generally speaking, methods of forecasting can be classified into three categories: qualitative techniques, time series analysis and projection, and causal models (ref: https://hbr.org/1971/07/how-to-choose-the-right-forecasting-technique) The first depends on qualitative data, like expert opinion, where past information may or may not be taken into consideration. The second focuses me merely on historical data. It explores patterns and pattern changes to predict future sales. The third uses highly refined and specific information about relationships between system elements.

In this grocery sales forecasting problem, historical datasets are provided. From the perspective of machine learning, I tackled the problem with a supervised regression method, model product unit sales with a parametric method.

The Kaggle competition saves a testing data from public, where true target values are hidden. The trained model is applied to this testing data set to generate prediction of grocery sales, and results are then submitted to Kaggle for evaluation.

1.2 Problem Statement

The task **T** of this challenge is to forecast product sales of a specific item in a specific store for some given dates. In this project, experience **E**, historical datasets, are provided to tackle this problem: sale records from Jan.1 2013 to Aug.16 2017, along with some other historical data, such as oil price, transaction, etc. Evaluation metric **P** is Normalized Weighted Root Mean Squared Logarithmic Error (NERMSLE). Based on historical sales records and related factors **E**, a supervised machine learning method can be employed to perform task **T** -- predict product sales, measured with metric **P**.

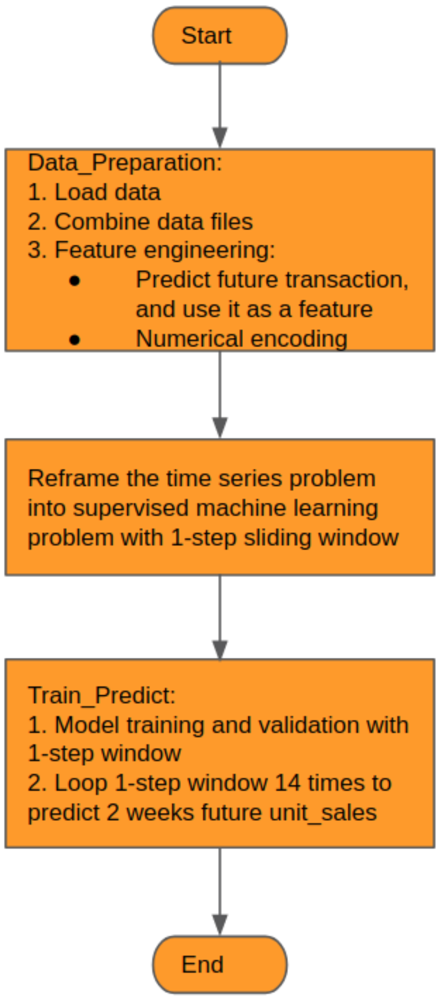
Start
Data_Preparation:
1. Load data
2. Combine data files
3. Feature engineering:
• Predict future transaction, and use it as a feature
• Numerical encoding
Reframe the time series problem into supervised machine learning problem with 1-step sliding window
Train_Predict:
1. Model training and validation with 1-step window
2. Loop 1-step window 14 times to predict 2 weeks future unit_sales
End

Figure 1: An overview of the pipeline

To solve this problem, I firstly fetch and pre-process raw data. There are several data files for this project. Related data is combined into a single DataFrame for feature engineering, such as numerical encoding of categorical features. To transform this time series problem into supervised machine learning problem, 1-step sliding window is applied on feature *unit_sales*. After data preparation, Extreme Gradient Boost regression model is trained and optimized with Randomized Search Cross-Validation. Finally, the optimized regression model is applied to forecast *unit_sales* for future dates. The entire machine learning pipeline is shown in Figure 1.

1.3 Metrics

Evaluation metric for the challenge is provided here https://www.kaggle.com/c/favorita-grocery-sales-forecasting#evaluation.

To be specific, the prediction is evaluated on the Normalized Weighted Mean Squared Loga-

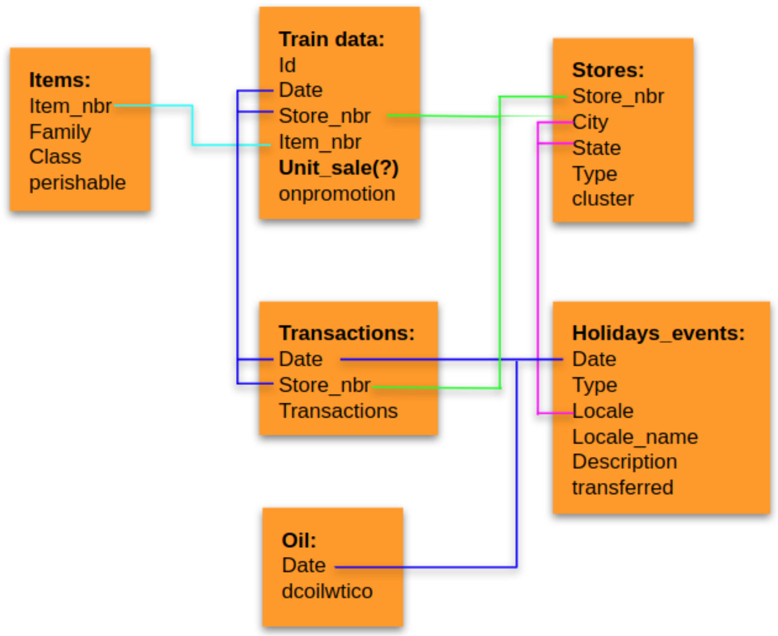

Figure 2: Relationship of all data files.

rithmic Error (NERMSLE):

$$\sqrt{\frac{\sum_{i=1}^{n} w_i (ln(\hat{y}_i+1) - ln(y_i+1))^2}{\sum_{i=1}^{n} w_i}},$$

where for row i, $\hat{y}_i$ is the predicted unit_sales of an item and y_i is the actual unit_sales; n is the total number of rows in the test set. w_i, the weights is given based on if an item is perishable or not: Perishable items have a weight of 1.25 and all other items have weights of 1.00.

This metric is able to avoid penalizing large differences when both the predicted and true numbers are large and it works well to predict values across a large range of orders of magnitudes.

2 Analysis

2.1 Data Exploration

Datasets are provided by Corporación Favorita: https://www.kaggle.com/c/favorita-grocery-sales-forecasting/data .

2.1.1 Data overview

An overview of datasets and relationship is shown in Figure 2:

5 data files have date, but with different ranges.

data file	starting date	ending date

data file	starting date	ending date
train.csv	2013-01-02	2017-08-15
test.csv	2017-08-16	2017-08-31
transactions.csv	2013-01-01	2017-08-15
oil.csv	2013-01-01	2017-08-31
holidays_events.csv	2012-03-02	2017-12-26

The date of oil.csv covers the date of test.csv, which is an improper data leakage, as the oil price is hard to achieve for a future date.

Sales data is like weather data, the more the time of the data is close to the date to predict, the more valuable the data is. Since data provided in this challenge is huge for model training, I trimmed the data, and only uses the recent 3 month sales data. Data from August 1st, 2017 to August 15th, 2017 are separated and saved for testing.

2.1.2 Data information

1. train.csv

   - id: This is meaningless for model training, and will be dropped
   - date: From 2013-01-01 to 2017-08-15
   - store_nbr: continuous integer from 1 to 54
   - item_nbr: item id, un-continuous integers
   - unit_sale: continuous float number with min=-0.000153 max=89440
   - onpromotion: boolean 0 1, and missing entries

column	date	id	stor_nbr	item_nbr	unit_sales	onpromotion
datatype	int32	date	int16	int32	float32	float32

2. holidays_events.csv

   - date: 312 unique dates from 2012-03-02 to 2017-12-26
   - type: ['Holiday', 'Transfer', 'Additional', 'Bridge', 'Work Day', 'Event']
   - locale: ['Local', 'Regional', 'National']
   - locale_name: ['Manta', 'Cotopaxi', 'Cuenca', 'Libertad', 'Riobamba', 'Puyo','Guaranda', 'Imbabura', 'Latacunga', 'Machala', 'Santo Domingo','El Carmen', 'Cayambe', 'Esmeraldas', 'Ecuador', 'Ambato', 'Ibarra','Quevedo', 'Santo

Domingo de los Tsachilas', 'Santa Elena', 'Quito','Loja', 'Salinas', 'Guayaquil']

- description:103 entries (don't understand)

- transferred: [False, True]

column	date	type	locale	locale_name	description	transferred
datatype	date	string	string	string	string	bool

3. stores.csv

- store_nbr: integer from 1 to 54
- city: ['Quito', 'Santo Domingo', 'Cayambe', 'Latacunga', 'Riobamba', 'Ibarra', 'Guaranda', 'Puyo', 'Ambato', 'Guayaquil', 'Salinas', 'Daule', 'Babahoyo', 'Quevedo', 'Playas', 'Libertad', 'Cuenca', 'Loja', 'Machala', 'Esmeraldas', 'Manta', 'El Carmen']
- state: ['Pichincha', 'Santo Domingo de los Tsachilas', 'Cotopaxi', 'Chimborazo', 'Imbabura', 'Bolivar', 'Pastaza', 'Tungurahua', 'Guayas', 'Santa Elena', 'Los Rios', 'Azuay', 'Loja', 'El Oro', 'Esmeraldas', 'Manabi']
- type: ['D', 'B', 'C', 'E', 'A']
- cluster: [13, 8, 9, 4, 6, 15, 7, 3, 12, 16, 1, 10, 2, 5, 11, 14, 17]

column	store_nbr	city	state	type	cluster
datatype	int64	string	string	string	int64

4. oil.csv

- date: from 2013-01-01 to 2017-08-31

- dcoilwtico: continuous value from 26.19~110.62

column	date	dcoilwtico
datatype	date	float64

5. transactions.csv

- date: from 2013-01-01 to 2017-08-15 - store_nbr: 54 store numbers
- transactions: integers between 5 and 8358 Transactions data in testing set is not provided, so this dataset will not be used in training, although transactions are highly related to unit sales we want to predict.

6. items

- item_nbr: 4100 discrete values
- family: ['GROCERY I' 'CLEANING' 'BREAD/BAKERY' 'DELI' 'POULTRY' 'EGGS' 'PERSONAL CARE' 'LINGERIE' 'BEVERAGES' 'AUTOMOTIVE' 'DAIRY' 'GROCERY II' 'MEATS' 'FROZEN FOODS' 'HOME APPLIANCES' 'SEAFOOD' 'PREPARED FOODS' 'LIQUOR,WINE,BEER' 'BEAUTY' 'HARDWARE' 'LAWN AND GARDEN' 'PRODUCE' 'HOME AND KITCHEN II' 'HOME AND KITCHEN I' 'MAGAZINES' 'HOME CARE' 'PET SUPPLIES' 'BABY CARE' 'SCHOOL AND OFFICE SUPPLIES' 'PLAYERS AND ELECTRONICS' 'CELEBRATION' 'LADIESWEAR' 'BOOKS']
- class: 337 discrete values
- perishable: 0 and1

column	item_nbr	family	class	perishable
datatype	int64	string	int64	int64

2.2 Exploratory Visualization

2.2.1 Transactions

From the figure below we can see that the transactions data have is yearly periodic.It has the highest peak at the end of a year and some ups and downs in the middle of the year.

Figure 4 shows the histogram of raw transactions, it ranges from 0 to 3000. Since the range of transactions in a store is wide, np.log1p is applied to transform the transaction data. The distribution after log-transformation is shown in Figure 5.

2.3 Algorithms and Techniques

2.3.1 Regression Model

In this project, I use Extreme Gradient Boosting (XGBoost) Regression, which is very popular to many Kaggle competition winners recently.
XGBoost is used for supervised learning problems. It is a tree-based regressor consisted of tree ensemble and tree boosting. With tree ensemble, several tree models are trained based on subsets of training data, and then combined together. Tree ensembling introduces bias, but is sufficient to reduces overfitting.

A bunch of comparasions show that XGBoost is much faster due to Gradient boosted trees.(https://jessesw.com/XG-Boost/) Gradient boosted trees are built in series so that a step of graident descent can be taken in order to minimize a loss function. In return, a much faster grid search is achieved for optimizing hyperparameters in model tuning.

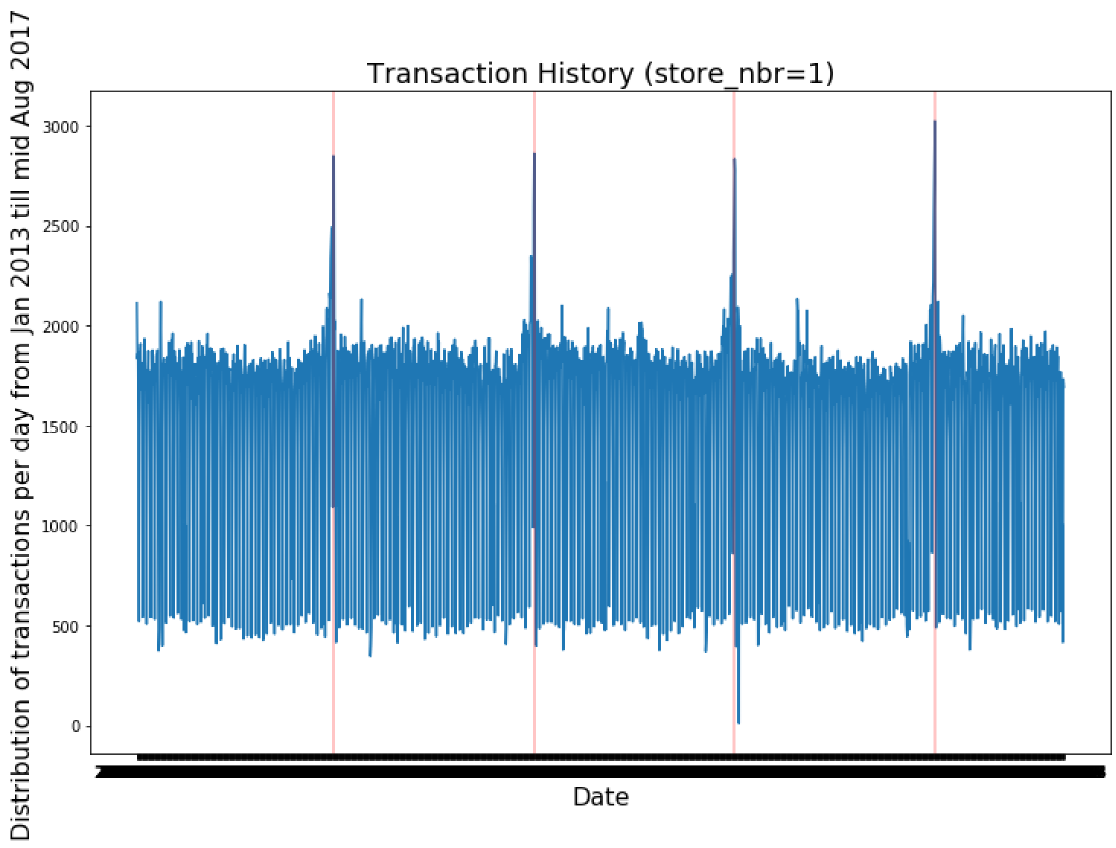

Figure 3: Transaction history of store 1

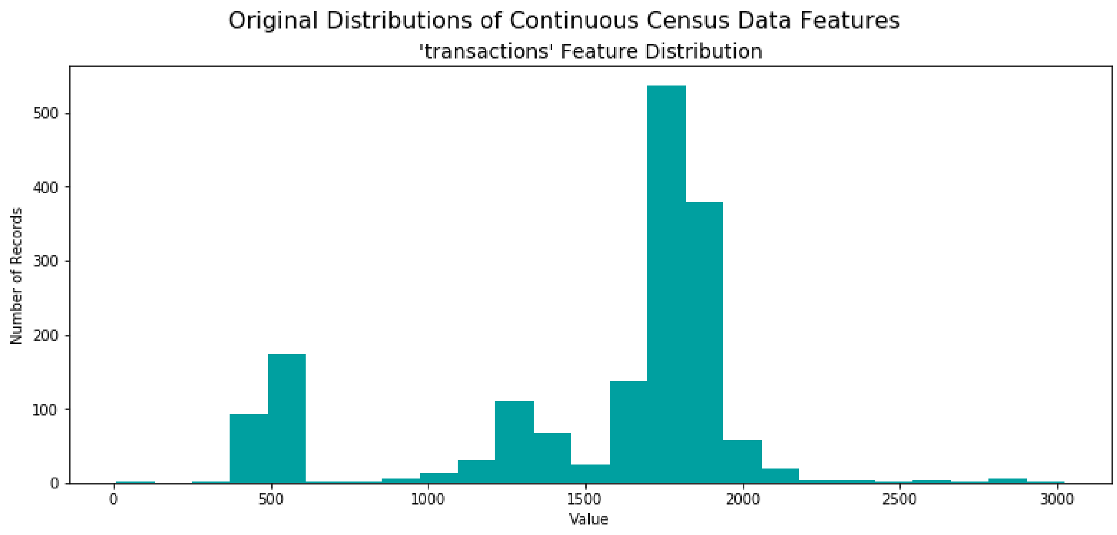

Figure 4: Raw transactions distribution

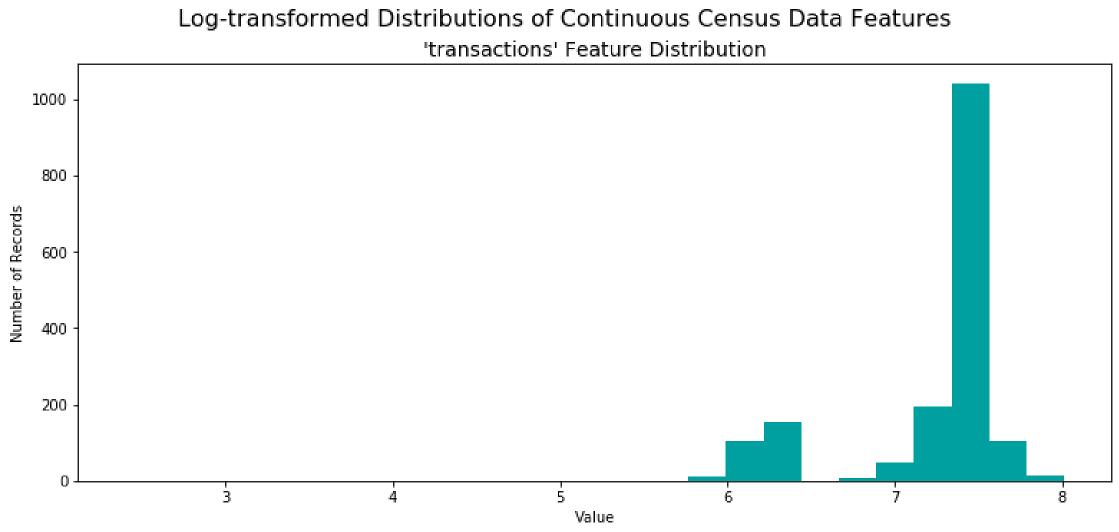

Figure 6: Log-transformed transactions distribution

2.3.2 Sliding window method

The grocery sales prediction problem is a time series problem. Time series problems can be phrased as machine learning problems with sliding window method. With sliding window method, the value at the previous time step can be used to predict the value at next time-step.

2.3.3 FBProphet

As we can see from Sec.2.1.1, date of testing data is from 2018-08-16 to 2018-08-31, but transactions data provided does not cover this range. From Figure 3, we can see that transactions signal is periodic with a peak at the end of each year. I use FaceBook Prophet(FBprophet) package to predict transactions of testing dates. FBprophet is able to integrate holidays and events and capture trends and seasonality. Transactions data is significant to predict *unit_sales*, so the predicted transactions, together with historical transactions provided, is used as a feature.

2.3.4 Numerical Encoding vs. One-Hot Encoding plus PCA

After one-hot-encoding, each sample has more than 4000 features. To reduce dimensionality, PCA is performed. Based on the cumulated explained_variance_ratio shown in Figure 6, if the number of PCA components is trimmed at 1000, about 80% of explained_variance_ratio is captured. The feature size is still very large.

Numerical encode simply represents categories with numerical number. One article (https://medium.com/data-design/visiting-categorical-features-and-encoding-in-decision-trees-53400fa65931) shows for tree-based model, Numerical-encoding performs better than One-Hot Encoding in both accuracy and speed. Here I'm going to use Extreme Gradient Boost Regression, which is indeed a tree-based regression model. So I'll use numerical encoding, instead of One-Hot encoding plus PCA.

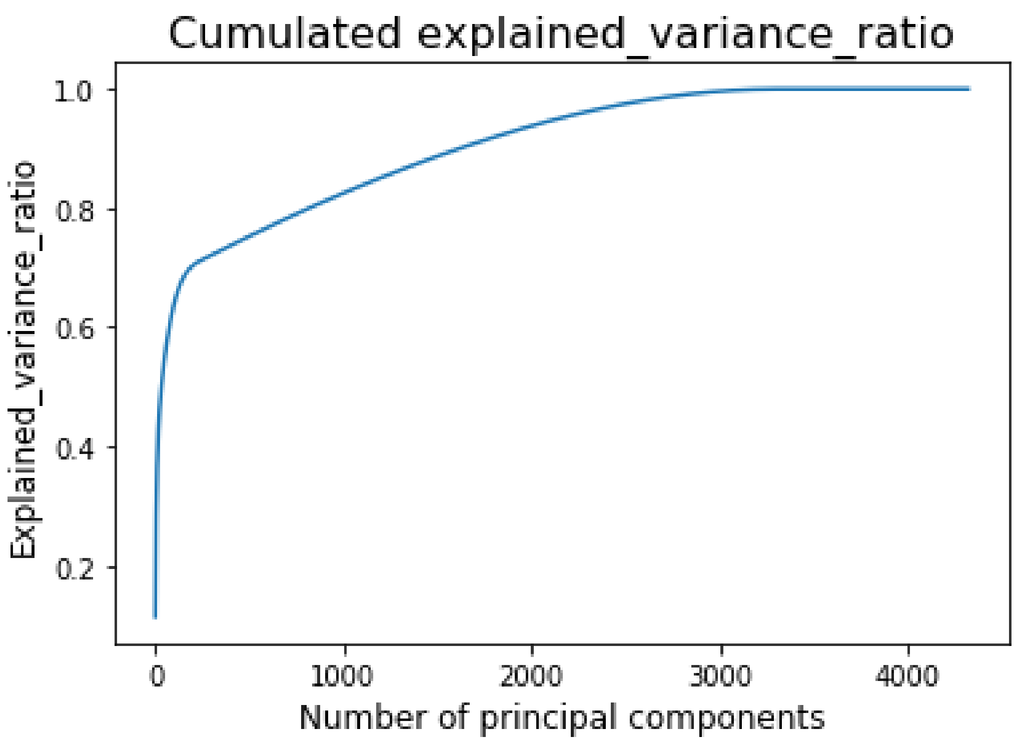

Figure 6: Accumulated explained variance ratio vs number of components of PCA

2.3.5 Data Separation and Trimming

The training dataset is large: 125,497,040 items across 4 years and 8 months. By setting proper datatypes of columns in Pandas function read_cvs, original 6 GB training data can be reduced to 4GB, which is still very large.
Since the sales of stores are independent, I firstly split data for different stores, and train models for each store. **All following experiments and analysis are based on Store_nbr = 1**

2.4 Benchmark

A naive model is to predict unit_sale based on historical values. This non-parametric model is an unweighted average of the historical values of the same item in the same store from the same date of previous 4 years. Unit_sales of items that can not be found in history record are filled with 0.
Evaluation from Kaggle website shows prediction from this method scors 0.908 (https://www.kaggle.com/c/favorita-grocery-sales-forecasting/submissions?sortBy=date&group=all&page=1)

3 Methodology

3.1 Data Preprocessing

Competitions on Kaggle provide a testing dataset with target values hidden from public. Sometimes, some features appears only in training data, while some only in testing data. To avoid extra

troubles caused by features only belongs to either training data or testing data, I combine training data and testing data together to perform data preprocessing, and split them when data preprocessing is completed.
Working with a complete data containing training data and testing data, items shown only in testing data are filled with 0 in training data, which is reasonable to train a model with situations never seen.

The preprocessing of data includes these steps:

1. Use FBProphet to predict transactions of future date in test.csv. Holiday_events, and payoff dates are taken into considerations, as they are important factors affecting transactions. Since some holidays and events are regional or local, location of a store is used to filter holidays and events for that specific store.

2. Data merge: training data with item.csv, also merge training and testing data with transactions for each date of every store.

3. Numerical encoding: My experiments shows numerical encoding competes with One-hot encoding at the prediction results, but much faster.

4. Moving average is computed and used as a feature. This is able to smoothen time-series signals.

3.2 Implementation

The implementation can be split into two stages: 1. The prediction of transactions with FBProphet 2. Regression model tuning

3.2.1 Transaction prediction

With FaceBook Prophet, future transactions can be predicted. Since holidays are location related, the store is firstly located. Holidays within the related locations are filtered and applied in Prophet. Together with historical transactions data, transactions in dates of testing data are predicted.
The transactions is well-predicted as shown in Figure 7. The end-of-year peak and some ups and downs in the middle of the year are captured by Prophet. Figure 8 shows the trend of all the transaction, effects of holidays and events, and yearly and weekly pattern. The yearly pattern shows the large peak at the end of a year, and weekly patten shows high peak at the end of weekend and lower peak at the middle of the week, which is consistent with our living experience.

Using FBProphet on transactions from different stores, we can see that periodic pattern changes. This is the main reason that I separate training and testing data for each store, and train models independently.

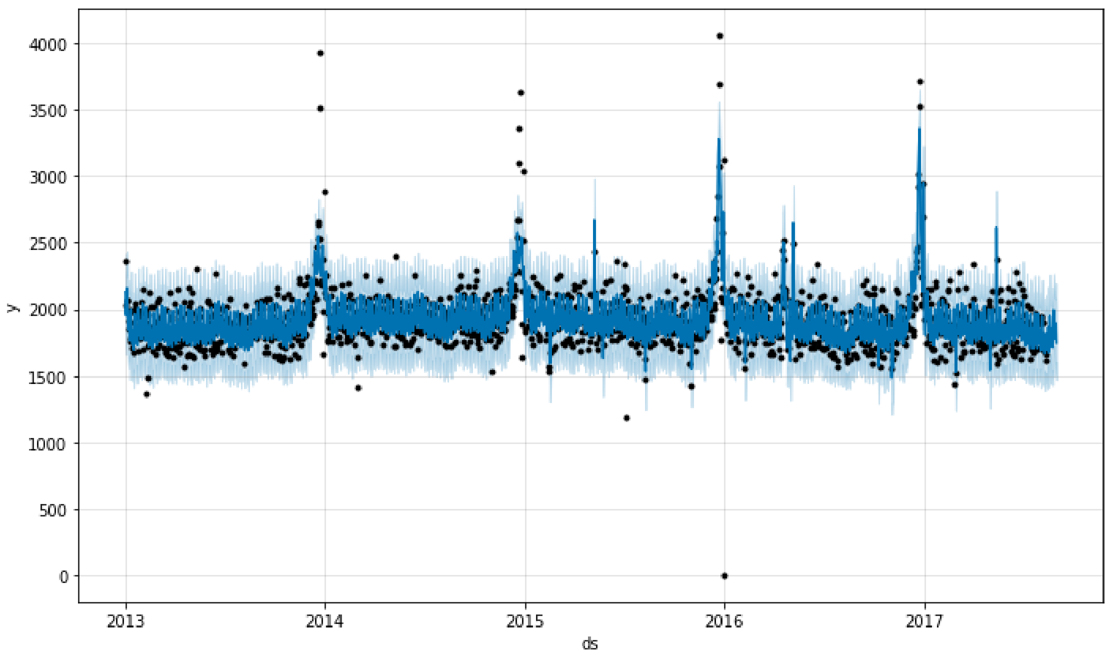

Figure 7: Predictions of transactions vs actual transactions

3.2.2 Metrics

In implementation, the target is log-transformed for modeling, then the metric defined in Sec 1.3. *Sample_weights* is simplified to the Normalized Weighted Mean Squared Error. Also, the squared root operation is gotten rid off without any effect on the usage of original metric.
Besides, perishable items and non-perishable items weigh differently. *Sample_weight* is introduced to deal with weighted *mean_square_error*.

3.2.3 1-Step Sliding Window

1-step sliding window is only able to predict *unit_sales* for 1 future date, while testing data in this project ranges 14 days. My solution is to predict *unit_sales* for next day, and use this prediction as history data to make predictions for the day after next day, so on and so forth.

3.3 Refinement

RandomizedSearchCV is employed to search for an optimal parameter set. The length of my training data is 5 times the length of testing data, so I use 5 fold cross-validation to make the length of validation data equal to the length of testing data.
Parameters optimized in RandomizedSearchCV include:

- "n_estimators": with too many estimators, XGBregression tends to overfit training data.

- "max_depth": with too many estimators, XGBregression tends to overfit training data.

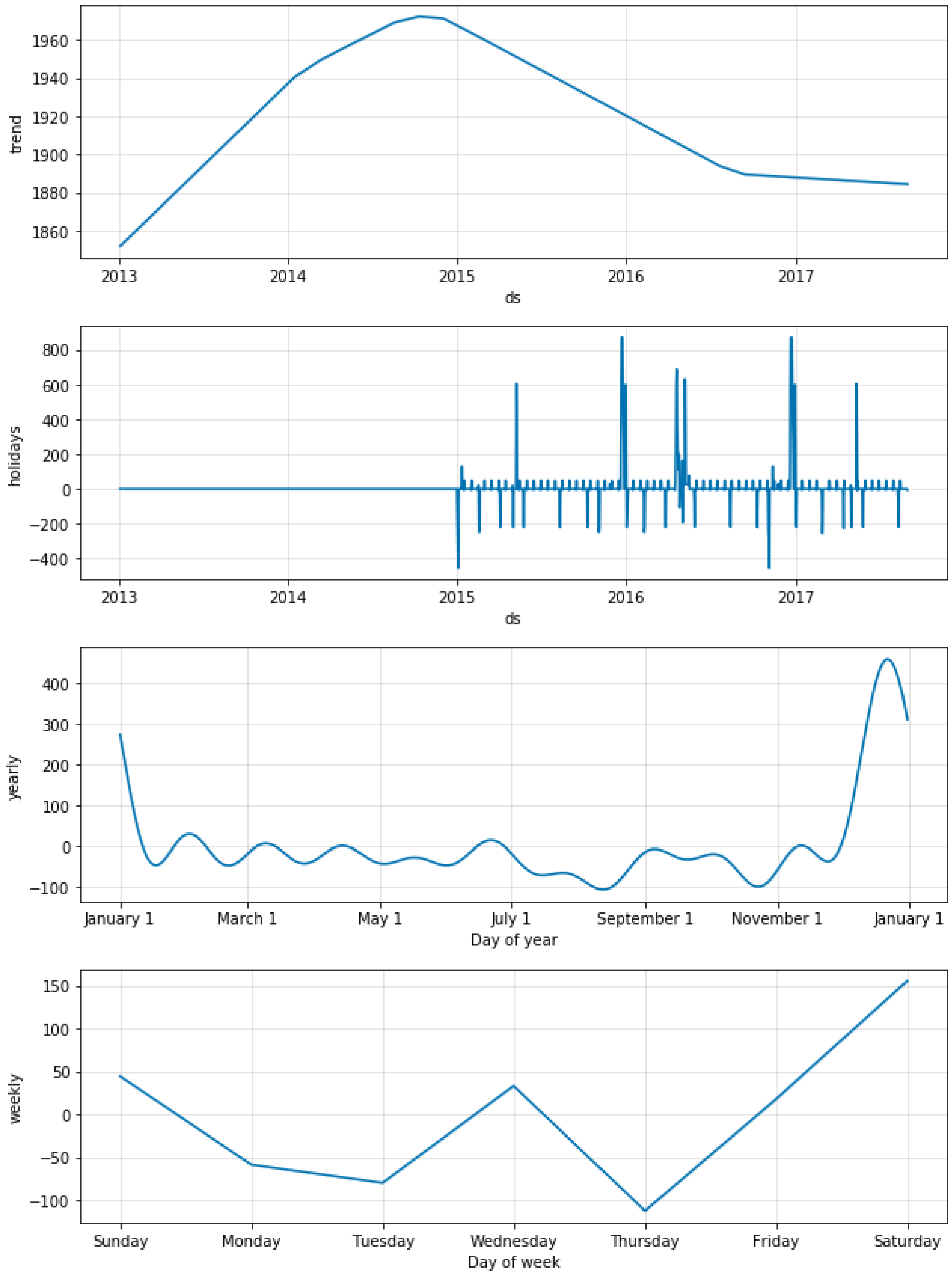

Figure 8: Predictions of transactions vs actual transactions

Trial number	Initial Trial	Trial 1	Trial 3	Trial-last
n_estimators	randint(5, 40)	randint(40, 80)	randint(80, 120)	randint(40, 80)
max_depth	randint(10, 30)	randint(10, 40)	randint(10, 40)	randint(40, 80)
learning_rate	uniform(0.05, 0.3)	uniform(0.05, 0.3)	uniform(0.02, 0.3)	uniform(0.1, 0.4)
gamma	uniform(0, 10)	uniform(0, 10)	uniform(0, 10)	uniform(0, 10)
reg_alpha	expon(0, 50)	expon(0, 50)	expon(0, 50)	expon(0, 50)
min_child_weight	expon(0, 50)	expon(0, 50)	expon(0, 50)	expon(0, 50)
Test Score	0.327	0.307	0.308	0.304
Training time(min)	4	12	14	13

Parameter range of parameters with randomized search cross-validation

- "learning_rate": Smaller learning_rate takes longer time to reach an optimum, while larger learning_rate learns faster, but may cause oscillation, and unable to reach an optimum.

- "colsample_bytree": Use leave-out-one to leave out a portion of features to avoid overfitting.

- "subsample": Use leave-out-one to leave out a portion of samples to avoid overfitting.

- "gamma": Minimum loss reduction required to make a future partition on a leaf node of the tree. The larger, the more conservative the algorithm will be.

- "reg_alpha": L1 regulation term on weights, increase this value will make model more conservative

- "min_child_weight:In linear regression mode, this corresponds to minimum number of instances needed to be in each node. The larger, the more conservative the algorithm is."

I tried different parameter ranges in in randomized search cross validation, Table 7 shows a part of parameter combinations tried. As we can see from this table, from randint(5, 40) (Initial trial) to randint(40, 80) (Trial 1), more estimators improve testing score. However, from randint(40,80) (Trial 1) to randint(80, 120) (Trail 2), there is no obvious change of test score. A bunch of parameter combinations are tried, and the final parameters used in randomized search cross-validation is listed on Trial-last.

4 Results

4.1 Model Evaluation and Validation

The model trained is chosen based on the result of RandomizedSearchCV. Optimized Hyperparameters are listed below:
{ 'gamma': 2.3252557176959785,
'learning_rate': 0.29619504198266383,
'max_depth': 46,

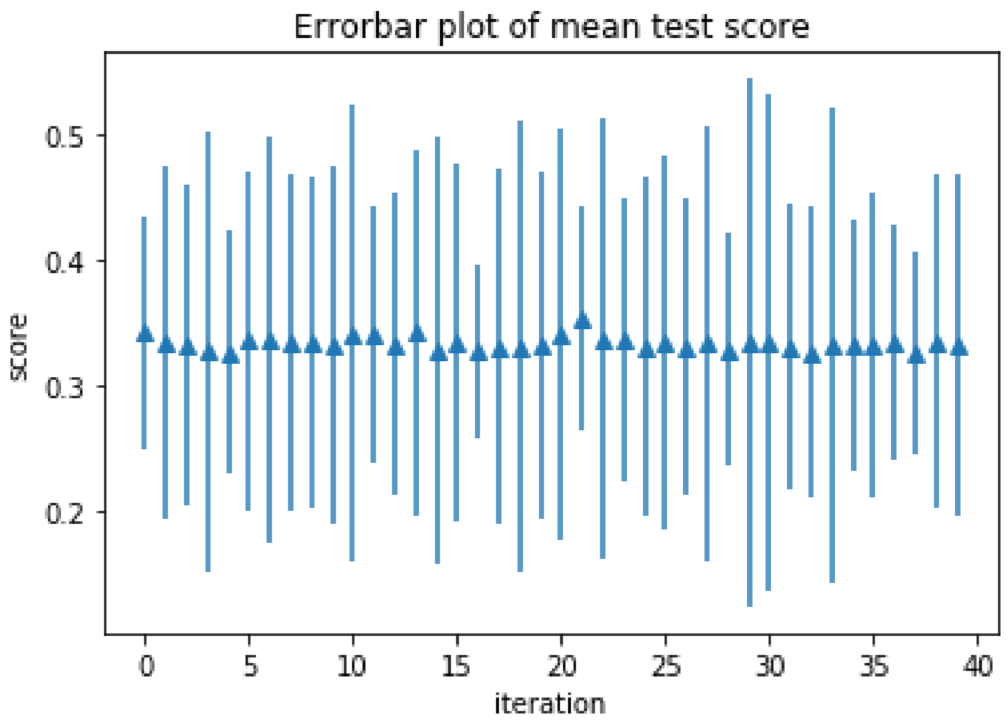

Figure 9: Error bar of test score in randomized search cross-validation.

'min_child_weight': 24.523669903588427,
'n_estimators': 61,
'reg_alpha': 26.861352394253334,}

Figure 9 shows the mean and standard deviation of testing score. The model performs consistently from the perspective of the mean of testing scores. However, the variance is still a little bit large.

4.2 Justification

The model is applied on Kaggle public testing data, where targets are unavailable. The final score of the model I tuned is 0.670. The benchmark I mentioned in the proposal is the average of historical unit_sales at the same day, which scores 0.908 on public testing data from Kaggle. Compared with the benchmark, xgboost greatly improved the predictions.
Currently, the best score on Kaggle Leaderboard is 0.503. So there is a lot space to improve the model. Still, there is a big gap between the test score with training data and the score from Kaggle.

5 Conclusion

Grocery prediction is critical for grocery retailers. This is a time-series related problem. Time series problems have some special characteristics compared with other machine learning problems. For example, it has temporal structure like trends and seasonality to be handled. FB Prophet is a good package to analyze and extract trends and seasonality. Also, historical value is important, where moving average is applied to catch historical information and reduce noise.

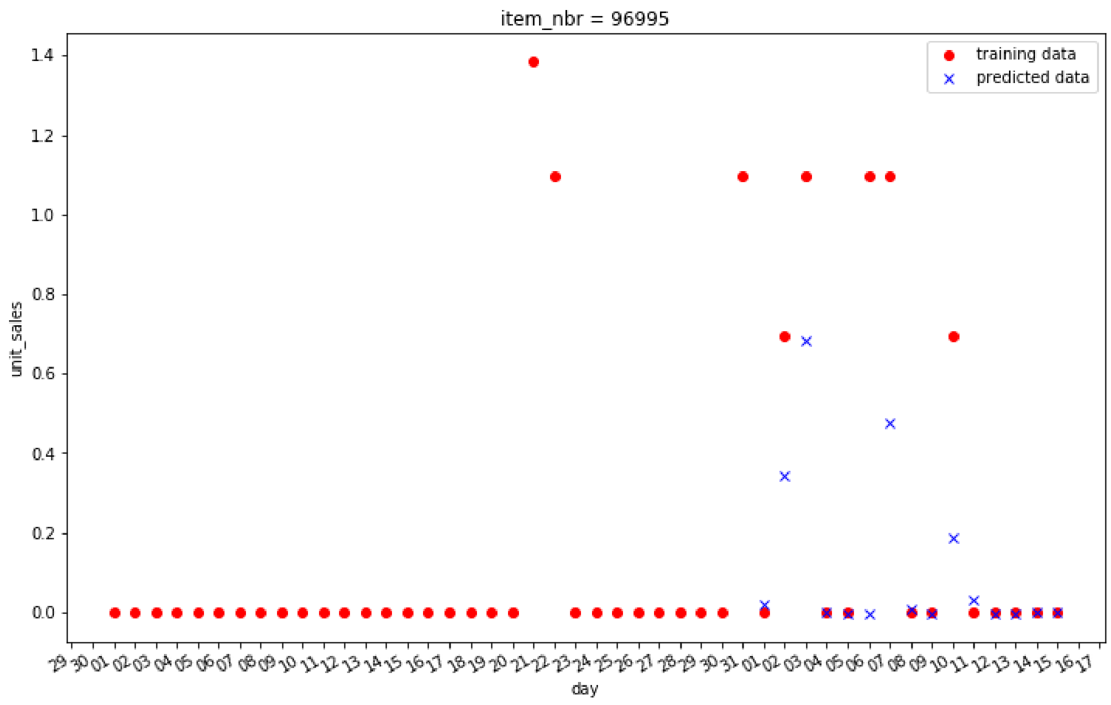

Figure 10:Predictions and history of item 96995 at store 1

The trained model is able to predict *unit_sales* approximately. Figure 10 and Figure 11 below show the true value and predicted value of 2 items from store 1. As we can see from both plots, the model predicts well on more than half of the days, and not very well on other days. Generally speaking, it's able to predict unit_sales for most days.

This trained model is applied on data from 2018-08-01 to 2018-08-15 to predict 15 days *unit_sales*. The testing score is 0.310.

5.1 Reflection

In addition to classic machine learning methods learned from MLND program, some time series specific approaches, such as FBProphet, moving average as feature engineering are used to tackle the problem. To be specific, time series problems are distinct from traditional machine learning problems as history data has effect on data at next time stamp. More techniques of time series problems need to be explored.

As the previous reviewer pointed out that classic cross-validation techniques such as K-Fold deals with samples that independent and identically distributed. However, in time series problems, an observation at next time stamp is related to that of last time stamp. That's why sliding window technique is used to reconstruct the time serious problem to machine learning problem. The only limitation is that with this technique, only predictions of the next day can be made directly, while predictions of one day later should be made based on the prediction of the next day. That's not the cross-validation technique can handle. In a word, the cross-validation in this project

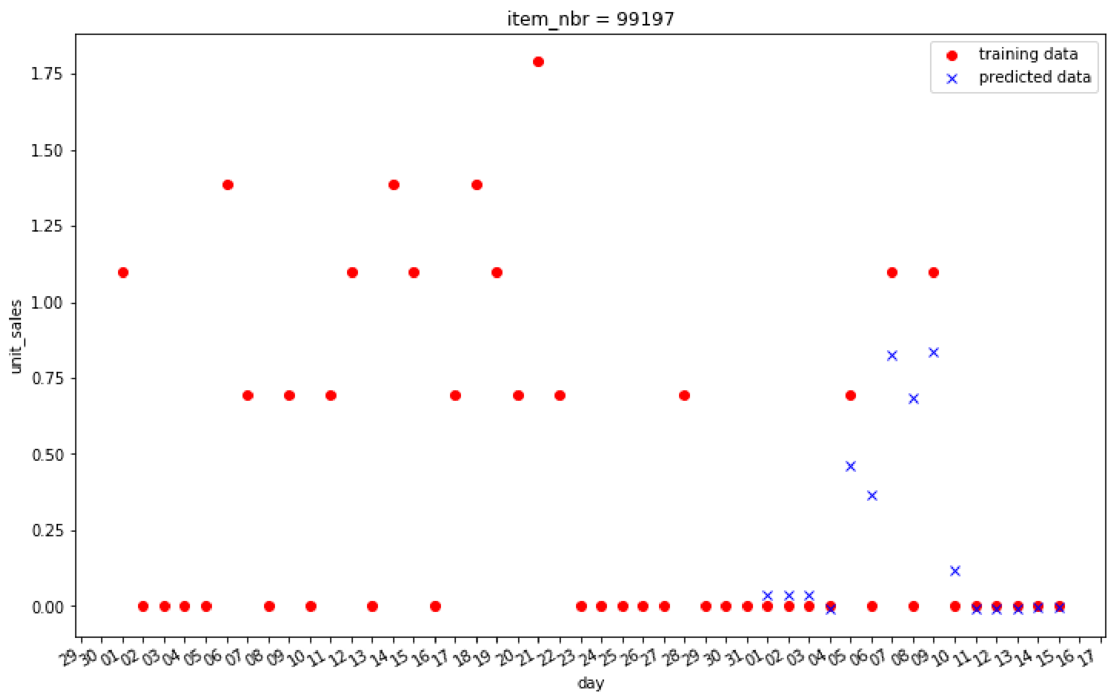

Figure 12: Predictions and history of item 99197 at store 1

only bases on the evaluation of predictions of the next day.

5.2 Improvement

Parallel computing can be introduced to save training time. In this project, model of each store is independent from each other, which is a good case for parallel computation.