Machine Learning Engineer Nanodegree

Capstone Project

Takayuki Sato
November 8th, 2019

I. Definition

Project Overview

Sales forecasting is very important for the retail business. An accurate sales forecast helps companies to

- Understand company's performance / customer demand
- Change strategy for the company's growth
- Conduct beneficial Marketing promotion
- Conduct better management of Inventory
- Adjust price adequately

Many factors, such as, Past sales result / Seasonality / Marketing promotion effect / Consumer Behavior / Price / Competitors' situation / weather / Economic indicators may affect sales forecasting result. There are also many methodologies for time series forecast [1, 2]. Companies need to identify important factors and select an adequate methodology for an accurate sales forecasting.

In this project, I created machine learning models for time series sales forecast. I used the historical sales data for 45 Walmart stores located in different regions.

Problem Statement

Walmart Inc. is an American multinational retail corporation that operates a chain of hypermarkets, discount department stores, and grocery stores. It is the world's largest company by revenue and the largest private employer in the world.

In their kaggle competition(Walmart Store Sales Forecasting), they provide historical sales data for 45 Walmart stores located in different regions. Each store contains many departments. The objective of this competition is to project the sales for each department in each store. To add to the challenge, selected holiday markdown events are included in the dataset. These markdowns are known to affect sales, but it is challenging to predict which departments are affected and the extent of the impact.

This is a typical time series problem. General time series modeling framework(e.g. ETS, ARIMA), Gradient Boosting framework(e.g. xgboost), and Neural Network framework(e.g. LSTM) are expected to be potential solutions.

Metrics

I used the weighted mean absolute error (WMAE) as an evaluation metric.

$$WMAE = \frac{1}{\sum w_i} \sum_{i=1}^{n} w_i \left| y_i - \widehat{y}_i \right|$$

where

n is the number of rows ŷ i is the predicted sales yi is the actual sales wi are weights. w = 5 if the week is a holiday week, 1 otherwise

By using the WMAE metric, I can select an accurate model for holiday seasons' sales that is especially important for Walmart.

II. Analysis

Data Exploration

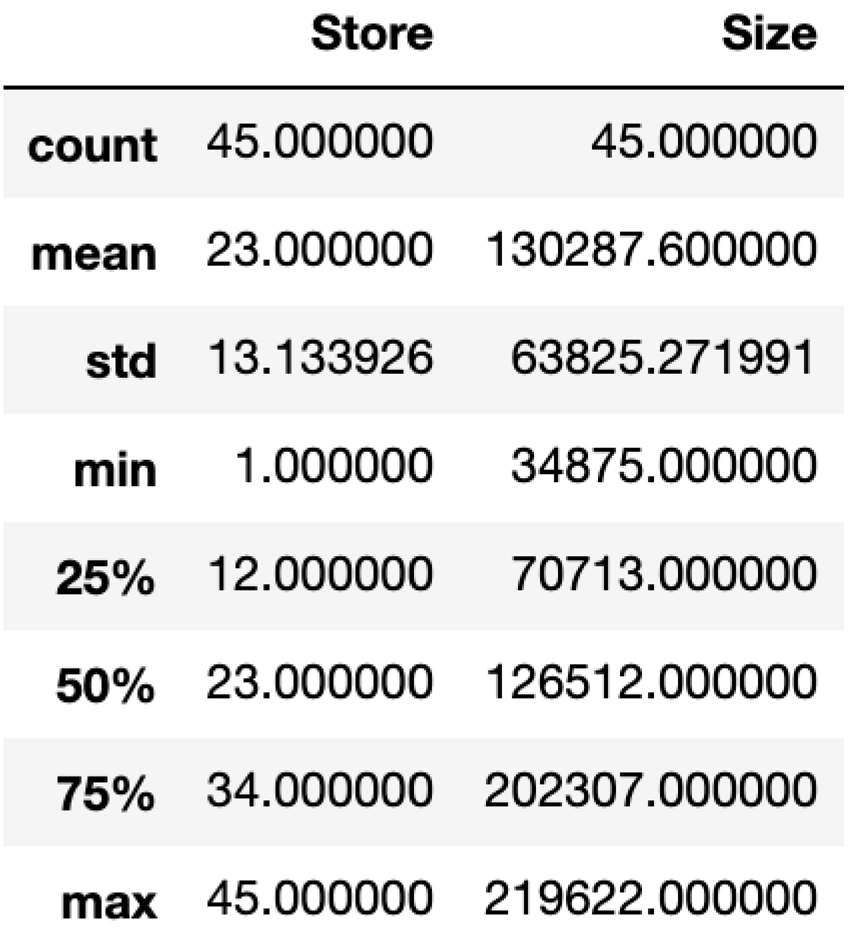
The dataset provided by the kaggle competition. (Walmart Store Sales Forecasting) includes historical sales data for 45 Walmart stores located in different regions.

stores.csv

This file contains anonymized information about the 45 stores, indicating the type and size of the store.

Below are top 5 rows of stores.csv and descriptive statistics;

	Store	Type	Size
0	1	A	151315
1	2	A	202307
2	3	B	37392
3	4	A	205863
4	5	B	34875



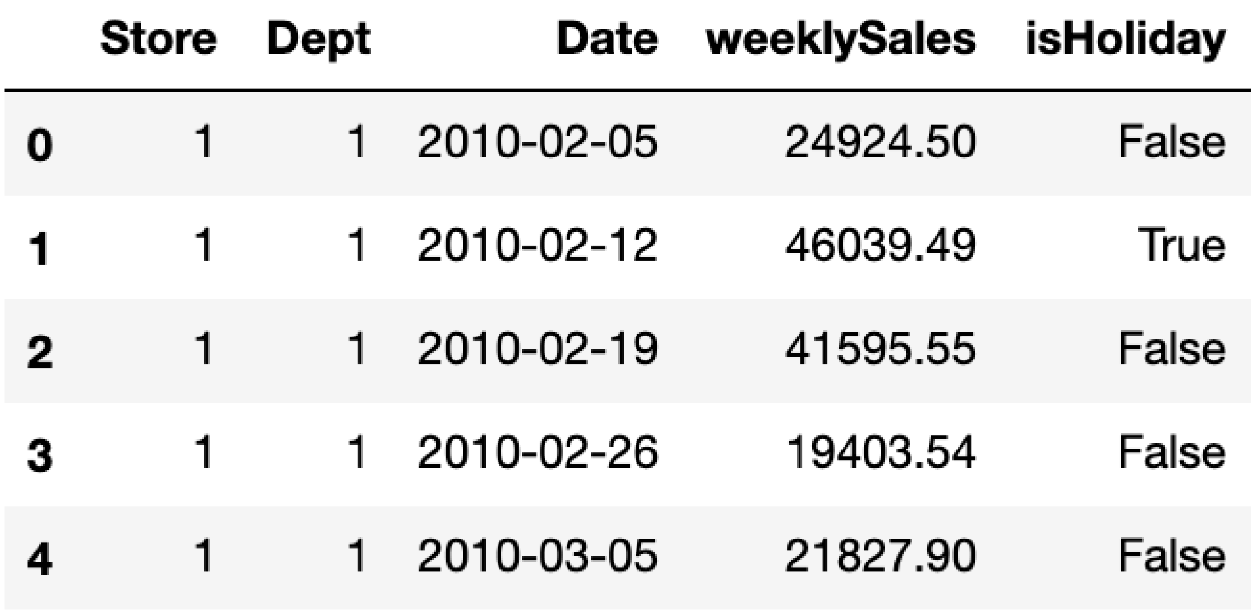
	Store	Size
count	45.000000	45.000000
mean	23.000000	130287.600000
std	13.133926	63825.271991
min	1.000000	34875.000000
25%	12.000000	70713.000000
50%	23.000000	126512.000000
75%	34.000000	202307.000000
max	45.000000	219622.000000

train.csv / test.csv

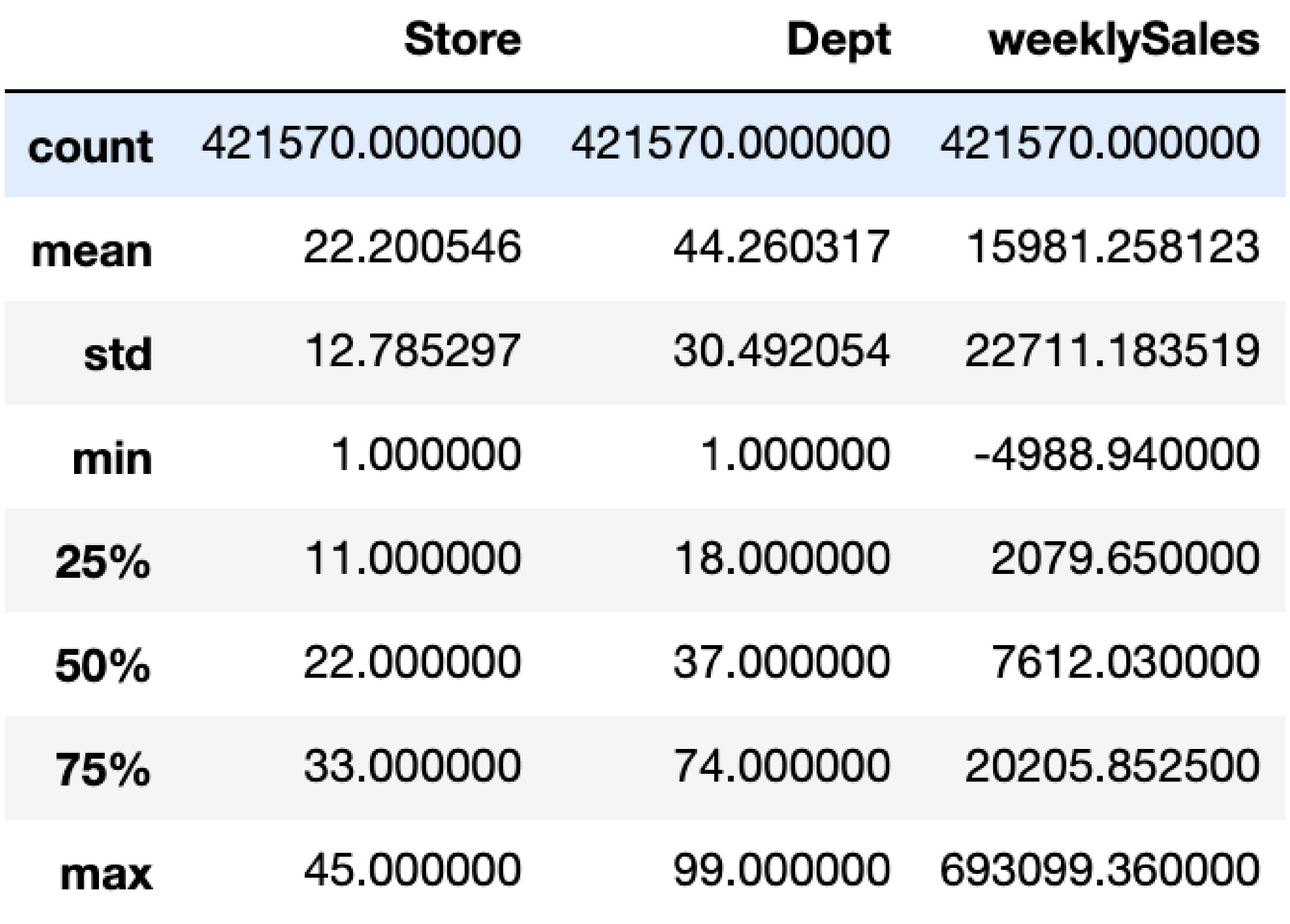
Train data is the historical weekly data, which covers to 2010-02-05 to 2012-11-01. Within this file you will find the following fields:

- Store - the store number
- Dept - the department number
- Date - the week
- Weekly_Sales - sales for the given department in the given store
- IsHoliday - whether the week is a special holiday week

Below are top 5 rows of train.csv and descriptive statistics;



	Store	Dept	Date	weeklySales	isHoliday
0	1	1	2010-02-05	24924.50	False
1	1	1	2010-02-12	46039.49	True
2	1	1	2010-02-19	41595.55	False
3	1	1	2010-02-26	19403.54	False
4	1	1	2010-03-05	21827.90	False



	Store	Dept	weeklySales
count	421570.000000	421570.000000	421570.000000
mean	22.200546	44.260317	15981.258123
std	12.785297	30.492054	22711.183519
min	1.000000	1.000000	-4988.940000
25%	11.000000	18.000000	2079.650000
50%	22.000000	37.000000	7612.030000
75%	33.000000	74.000000	20205.852500
max	45.000000	99.000000	693099.360000

Test data is the historical weekly data, which covers to 2012-11-02 to 2013-08-02, with the same fields except for sales.

features.csv

This file contains additional data related to the store, department, and regional activity for the given dates. It contains the following fields:

- Store - the store number
- Date - the week

- Temperature - average temperature in the region
- Fuel_Price - cost of fuel in the region
- MarkDown1-5 - anonymized data related to promotional markdowns that Walmart is running. MarkDown data is only available after Nov 2011, and is not available for all stores all the time. Any missing value is marked with an NA.
- CPI - the consumer price index
- Unemployment - the unemployment rate
- IsHoliday - whether the week is a special holiday week

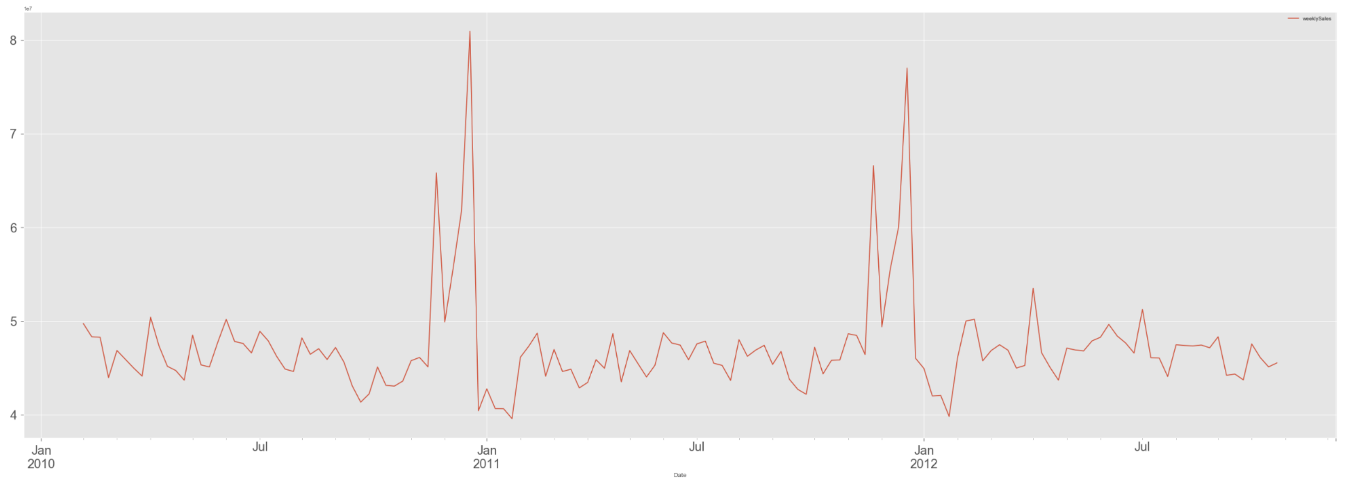
Below are top 5 rows of features.csv and descriptive statistics;

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	isHoliday
0	1	2010-02-05	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	8.106	False
1	1	2010-02-12	38.51	2.548	NaN	NaN	NaN	NaN	NaN	211.242170	8.106	True
2	1	2010-02-19	39.93	2.514	NaN	NaN	NaN	NaN	NaN	211.289143	8.106	False
3	1	2010-02-26	46.63	2.561	NaN	NaN	NaN	NaN	NaN	211.319643	8.106	False
4	1	2010-03-05	46.50	2.625	NaN	NaN	NaN	NaN	NaN	211.350143	8.106	False

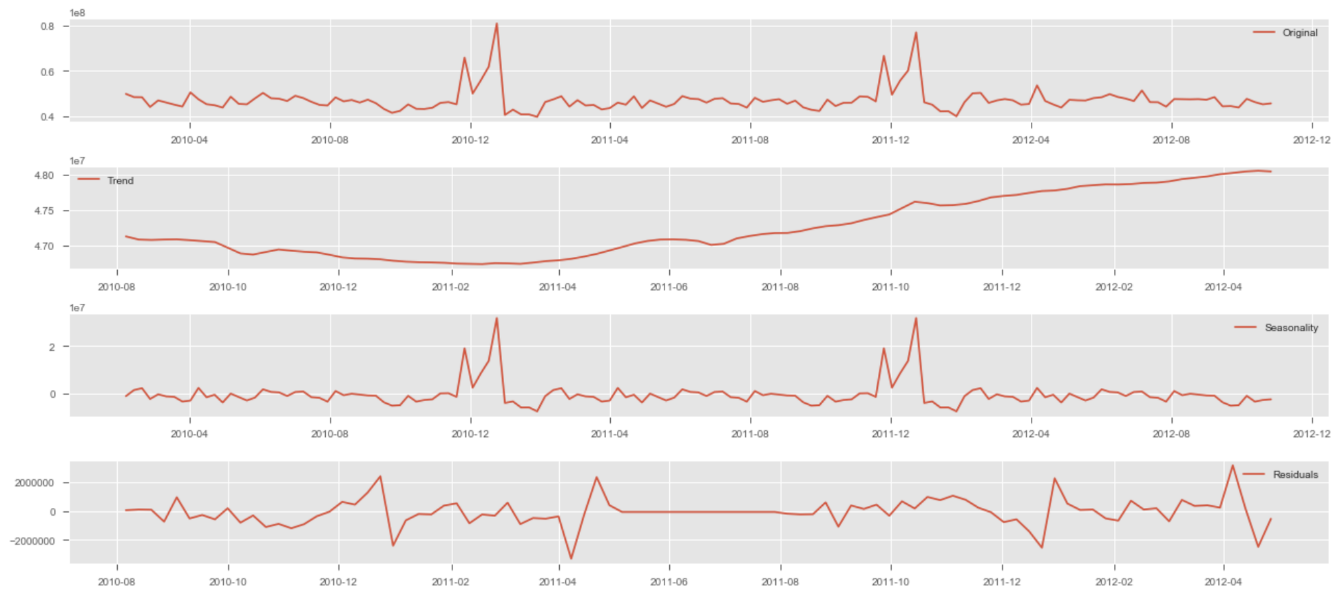
	Store	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment
count	8190.000000	8190.000000	8190.000000	4032.000000	2921.000000	3613.000000	3464.000000	4050.000000	7605.000000	7605.000000
mean	23.000000	59.356198	3.405992	7032.371786	3384.176594	1760.100180	3292.935886	4132.216422	172.460809	7.826821
std	12.987966	18.678607	0.431337	9262.747448	8793.583016	11276.462208	6792.329861	13086.690278	39.738346	1.877259
min	1.000000	-7.290000	2.472000	-2781.450000	-265.760000	-179.260000	0.220000	-185.170000	126.064000	3.684000
25%	12.000000	45.902500	3.041000	1577.532500	68.880000	6.600000	304.687500	1440.827500	132.364839	6.634000
50%	23.000000	60.710000	3.513000	4743.580000	364.570000	36.260000	1176.425000	2727.135000	182.764003	7.806000
75%	34.000000	73.880000	3.743000	8923.310000	2153.350000	163.150000	3310.007500	4832.555000	213.932412	8.567000
max	45.000000	101.950000	4.468000	103184.980000	104519.540000	149483.310000	67474.850000	771448.100000	228.976456	14.313000

Exploratory Visualization

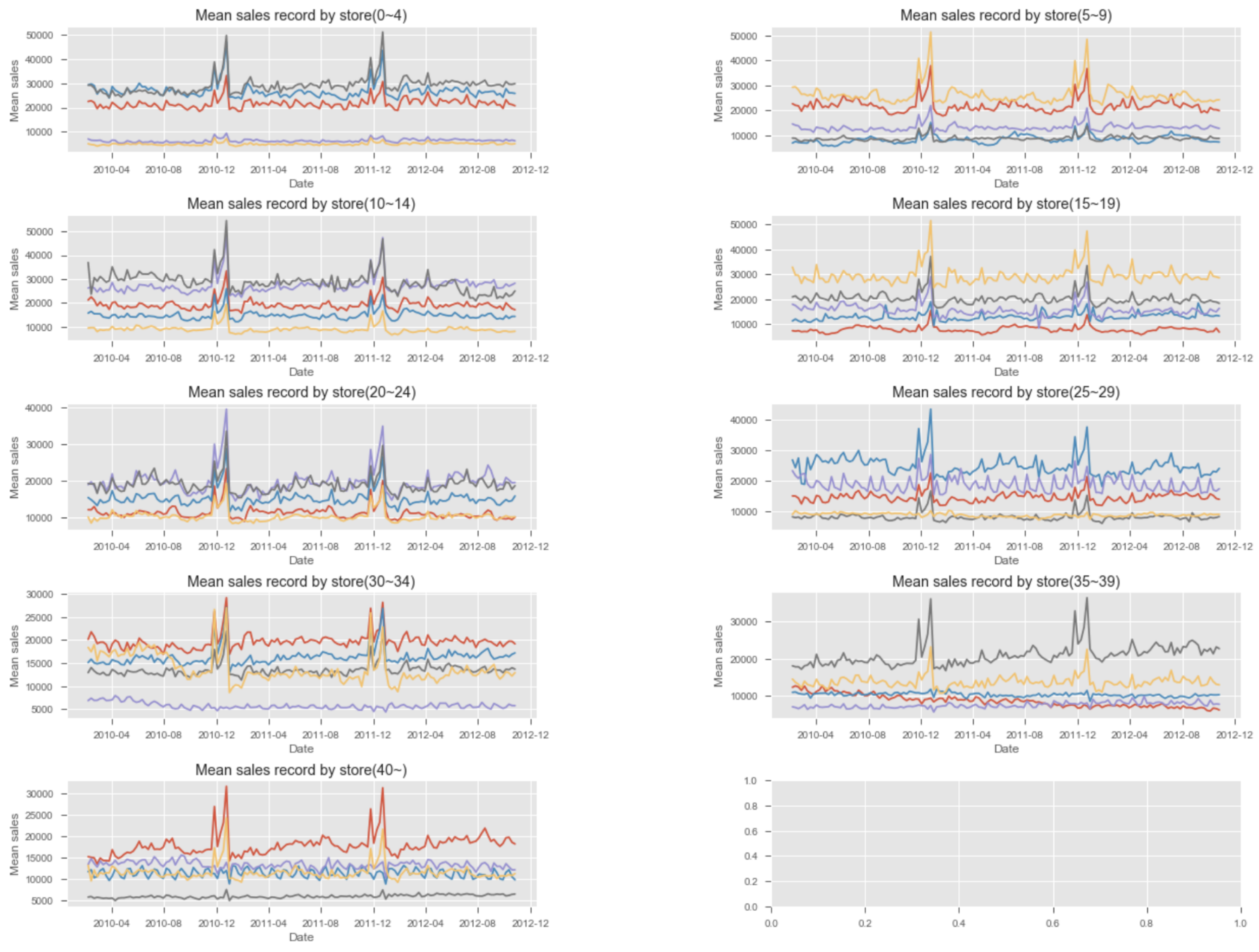
The plot below shows aggregated sales by each day.

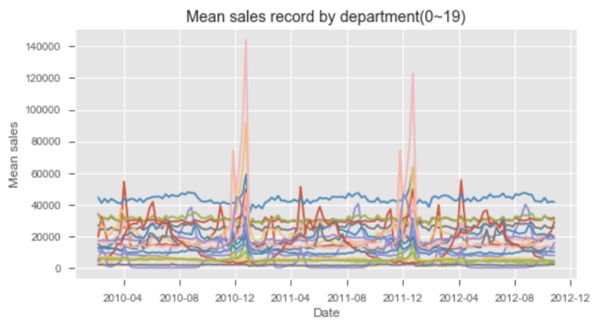


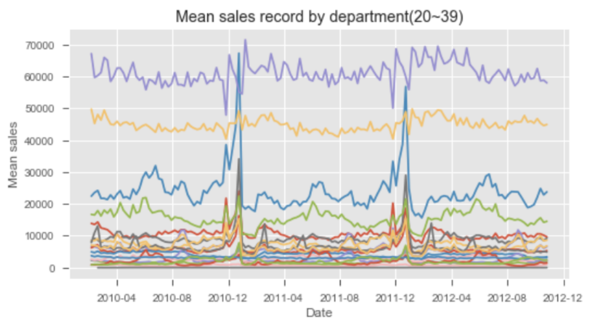
Below chart is the decomposition of Total sales. We can see there is an upward trend in the series. There is also seasonality, huge spikes around every December.

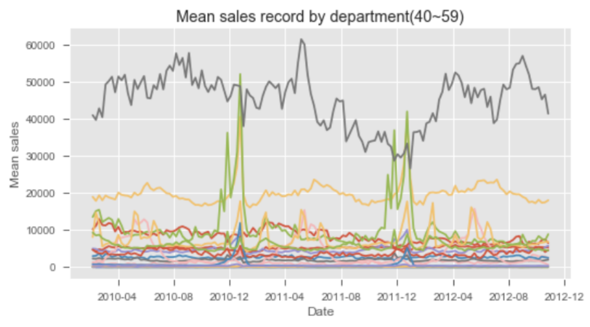


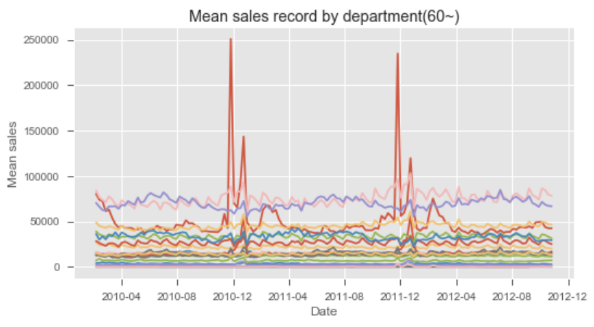
We can see almost the same tendency by each store/department, but some stores/departments seem to have small trend/seasonality.



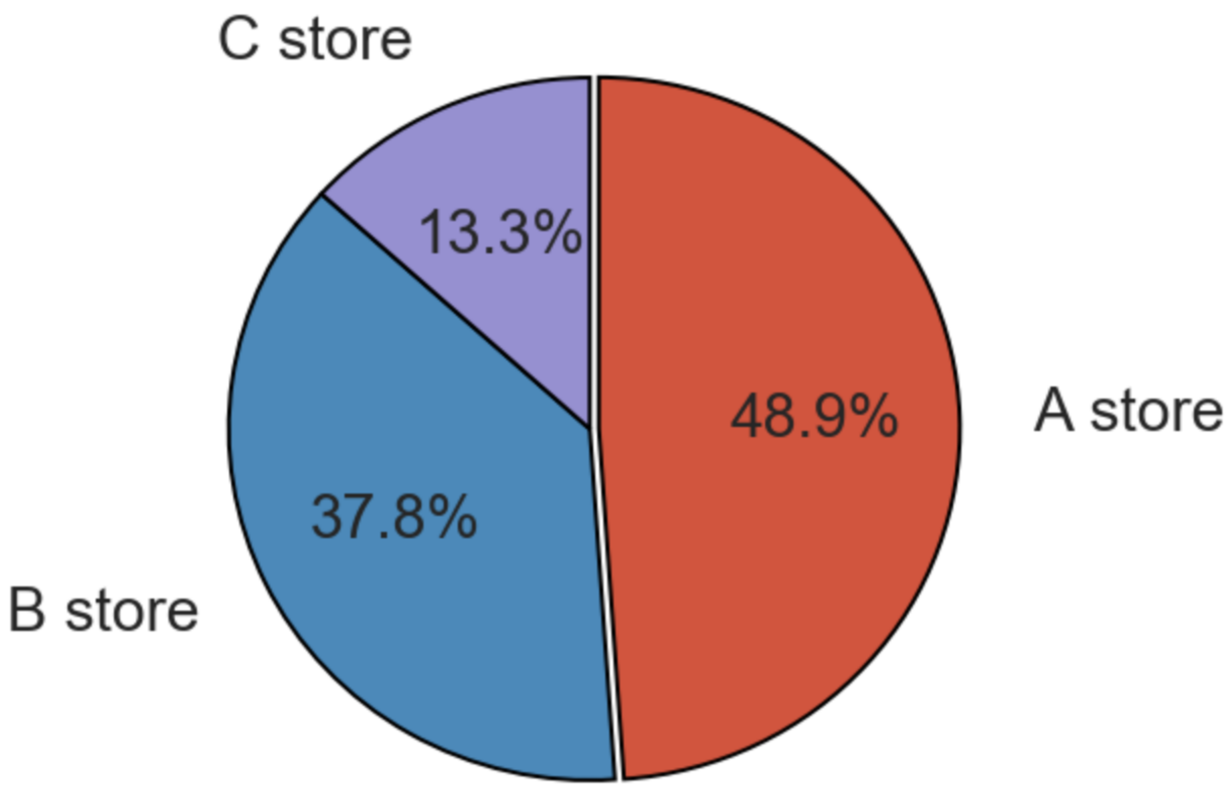


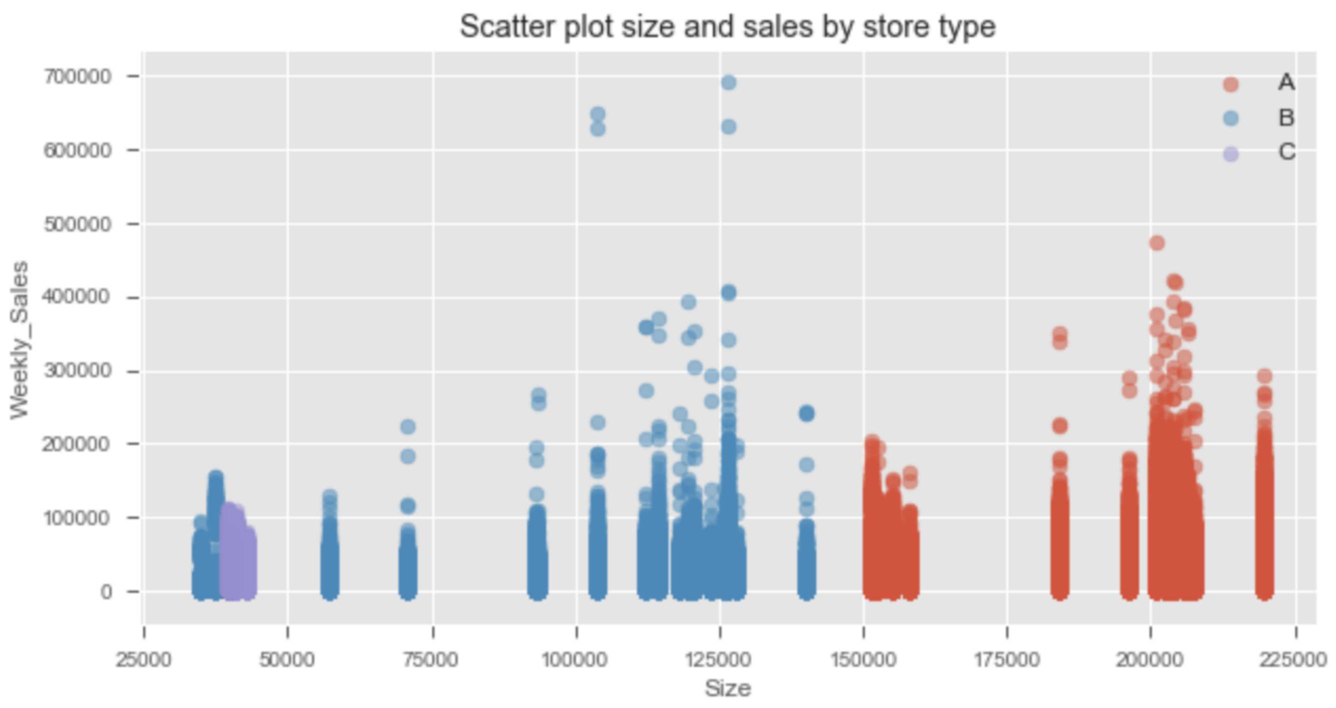






Walmart has three types of stores. big-sized A type store has big sales, whereas small-sized C type has small sales.





Algorithms and Techniques

In this project, I used Random Forest, XGBoost, and Long short-term memory for solving the problem. Weekly sales are the target. For 2 tree-based models(Random Forest & XGBoost), I used all features that were provided as features.csv by kaggle. For LSTM, I used only lag sales as input.

The following parameters are mainly tuned to optimize the model(and a default parameter I tried at first).

- Random Forest
  - `n_estimators`:number of estimators(default: 100)
  - `n_estimators`:minimum sample leaf size(default:)
- XGBoost
  - `alpha`:L1 regularization term on weight(default: 0)
  - `n_estimators`:number of estimators(default: 100)
  - `max_depth`:Maximum depth of a tree(default: 6)
  - `learning_rate`:The shrinkage at every step of tree boosting, used to prevent overfitting(default: 0.3)
  - `min_child_weight`:Minimum sum of weights of all observations required in a child node(default: 1)
  - `colsample_bytree`:The subsample ratio of columns when constructing each tree(default: 1)
  - `colsample_bylevel`:The subsample ratio of columns when constructing each level(default: 1)
  - `reg_lambda`:L2 regularization term on weight(default: 1)
  - `subsample`:Subsample ratio of the training instances(default: 1)
- LSTM
  - number of layers(default:2)
  - the number of hidden units(default:50 by each layer)
  - the sequence length(The number of lags used for the forecast)(default:1)

    - dropout rate for regularization(default:0.5)
    - epochs(The number times that the learning algorithm will work through the entire training dataset. One epoch consists of one full training cycle on the training set.)(default: 100)

Benchmark

I used Decision Tree Regressor as a benchmark model and made a comparison between all models' performance and the benchmark. Decision Tree Regressor is a tree-like model that is simple to understand and interpret, but often relatively unstable. I tuned below parameters to get the best performance.

- Decision Tree
    - max_depth: 30
    - min_samples_split: 30
    - max_leaf_nodes: 360

The performance(WMAE) of this benchmark is 4,215(validation data), 6,875(test data).

III. Methodology

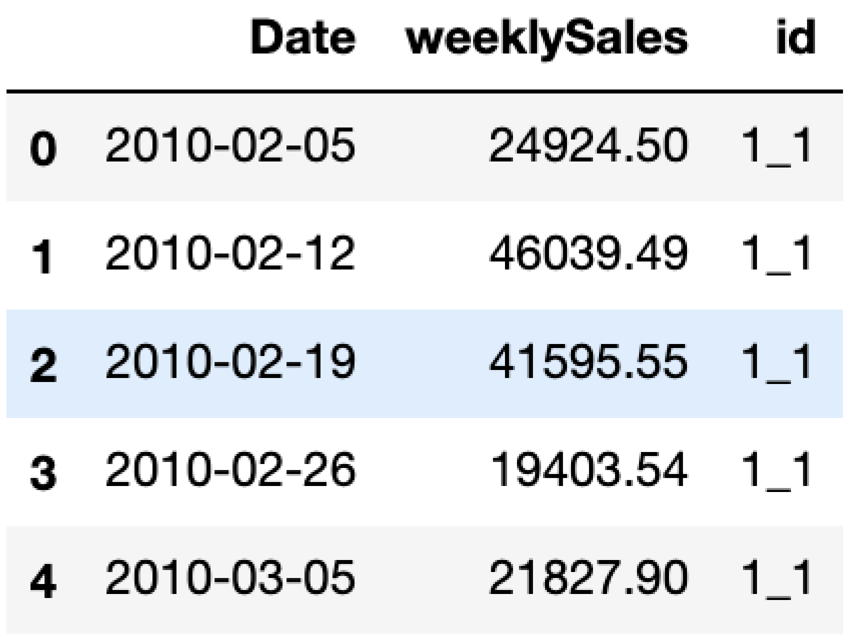
Data Preprocessing

I conducted some feature engineering to make the given dataset to fit tree-based machine learning models.

- isHoliday data is bool type, so I cast that to integer[0,1].
- Markdown data has some NaN / negative values, so I convert that to zero.
- Converted Date(yyyy-mm-dd format) to "year", "month", "week", and "day".
- Add cumulative week count "n_days" since there is an upward trend in the series.
- Store data is categorical, so I used One-Hot encoding.

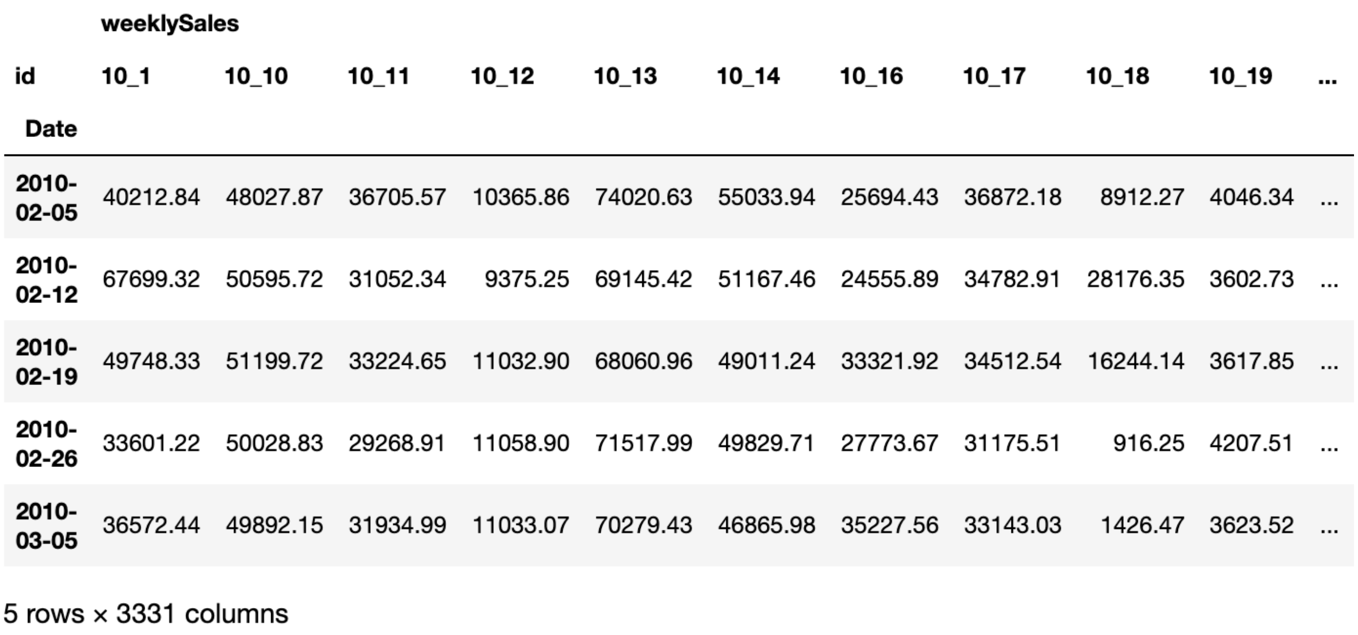
For only LSTM, I used only sales as input but sales data must be reshaped since the input data for LSTM must be in the form of a 3-dimensional array. The three dimensions are Samples, Time Steps (the sequence length), and Features. Also, the input needs to be scaled. Below are detailed data preprocessing steps for LSTM.

- In this project, we need to predict sales regarding all combinations of Stores-Departments. I concatenated "Store" and "Department" to "id". The number of unique values("id") is 3,331, which means the number of features for the LSTM is 3,331.



	Date	weeklySales	id
0	2010-02-05	24924.50	1_1
1	2010-02-12	46039.49	1_1
2	2010-02-19	41595.55	1_1
3	2010-02-26	19403.54	1_1
4	2010-03-05	21827.90	1_1

- Then I reshaped the dataset(values of "id" to column headers) by using pivot orientation.



	weeklySales										
id	10_1	10_10	10_11	10_12	10_13	10_14	10_16	10_17	10_18	10_19	...
Date											
2010-02-05	40212.84	48027.87	36705.57	10365.86	74020.63	55033.94	25694.43	36872.18	8912.27	4046.34	...
2010-02-12	67699.32	50595.72	31052.34	9375.25	69145.42	51167.46	24555.89	34782.91	28176.35	3602.73	...
2010-02-19	49748.33	51199.72	33224.65	11032.90	68060.96	49011.24	33321.92	34512.54	16244.14	3617.85	...
2010-02-26	33601.22	50028.83	29268.91	11058.90	71517.99	49829.71	27773.67	31175.51	916.25	4207.51	...
2010-03-05	36572.44	49892.15	31934.99	11033.07	70279.43	46865.98	35227.56	33143.03	1426.47	3623.52	...

5 rows × 3331 columns

- The input data has weekly sales of different stores/departments that highly vary in magnitudes. The input data needs to be scaled to make neural network learning and converge faster. I used Min-Max scaler to scale the input in the range of 0 to 1.
- I set a rolling window of length 1 week to create lag sales, moved this window along the time series. Then the dataset became a 3-dimensional array(142-1-3,331).
- Finally, I split the data into training and validation period(80% training set – 20% validation) Since this is time-series data, I split the data in sequential order(not randomly).

Implementation

<u>Tree-based models</u>

- Using all features as mentioned above.

- The objective function was 'reg:linear'(XGBoost).

<u>LSTM</u>

The LSTM model was implemented by using the Keras module. I used the Relu activation function and dropout layers. The final layer was the output dense(3,331 neurons) with a linear activation function. The loss function was mean squared error, and the optimizer was Adam(RMSprop with momentum). After forecasting one step ahead sales, combined the forecast with the input to forecast the next step ahead. Finally, I inverted the transforms to return the values back into the original scale.

The model's performance is evaluated by its WMAE score on the test data.

Refinement

I conducted an improvement of the performance of The tree-based machine learning algorithmx by Hyper parameter tuning. The following parameters were tuned by using **GridSearchCV**.

<u>Random Forest</u>

min_samples_split:

- Initial value: 0
- Search values: [0,2,4,6,8,10,12]
- Final value: 8

min_samples_leaf:

- Initial value: 1
- Search values: [1,5,10,12,14,16,18,20,30,50,100]
- Final value: 1

max_depth:

- Initial value: 10
- Search values: [10,20,30,50,70,90,110]
- Final value: 110

n_estimators:

- Initial value: 50
- Search values: [20,50,100,200,350,360]
- Final value: 200

<u>XGBoost</u>

alpha:

- Initial value: 0
- Search values: [0, 5,10,30,50,100]
- Final value: 10

n_estimators:

- Initial value: 50
- Search values: [5,10,30,50,100,300,400,500]
- Final value: 500

max_depth:

- Initial value: 5
- Search values: [5,10,12,14,16,18,20,30,50,100]
- Final value: 14

learning_rate:

- Initial value: 0.3
- Search values: [0.1,0.3,0.5,0.7,1]
- Final value: 0.1

min_child_weight:

- Initial value: 1
- Search values: [0,0.2,0.4,0.6,0.8,1]
- Final value: 0

colsample_bytree:

- Initial value: 1
- Search values: [0.1,0.3,0.5,0.7,0.85,0.9,1]
- Final value: 0.85

colsample_bylevel:

- Initial value: 1
- Search values: [0.5,0.6,0.8,1]
- Final value: 0.8

reg_lambda:

- Initial value: 1
- Search values: [0,1,2,4,6,8,10]
- Final value: 2

subsample:

- Initial value: 1
- Search values: [0.5,0.6,0.8,1]
- Final value: 0.8

For LSTM, below are final parameters I tried to tune, and they gave the best result.

LSTM

sequence length:

- Initial value: 1

- Final value: 4 (which means 4 weeks/1 month.)

number of layers(and number of hidden units):

- Initial value: 2 with hidden units of 50-50
- Final value: 3 with hidden units of 200-200-200

dropout:

- Initial value: 0.5
- Final value: 0.1

number of epochs:

- Initial value: 100
- Final value: 10,000

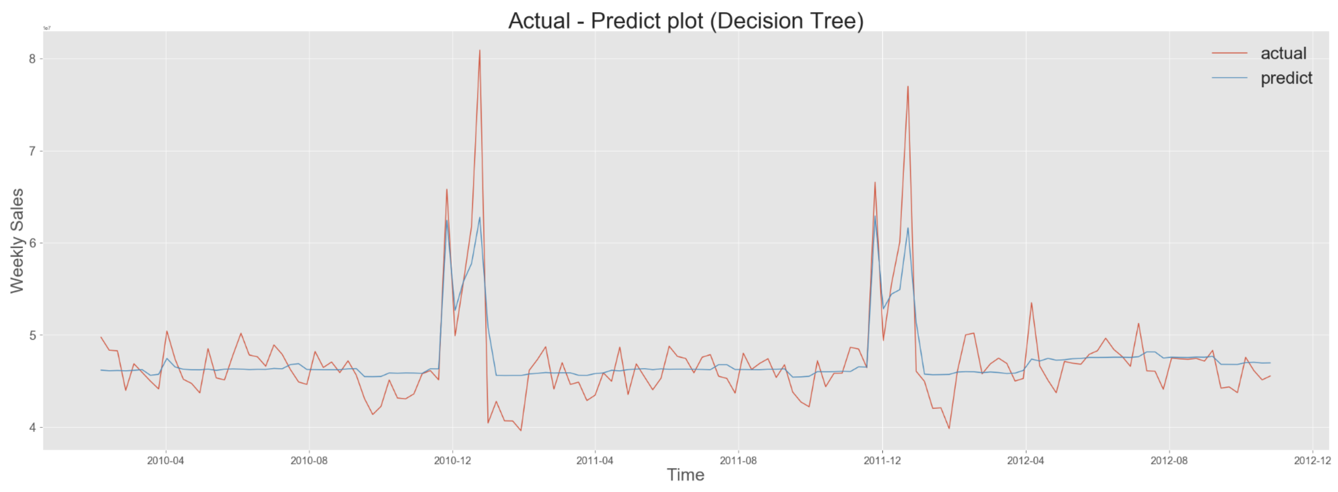
IV. Results

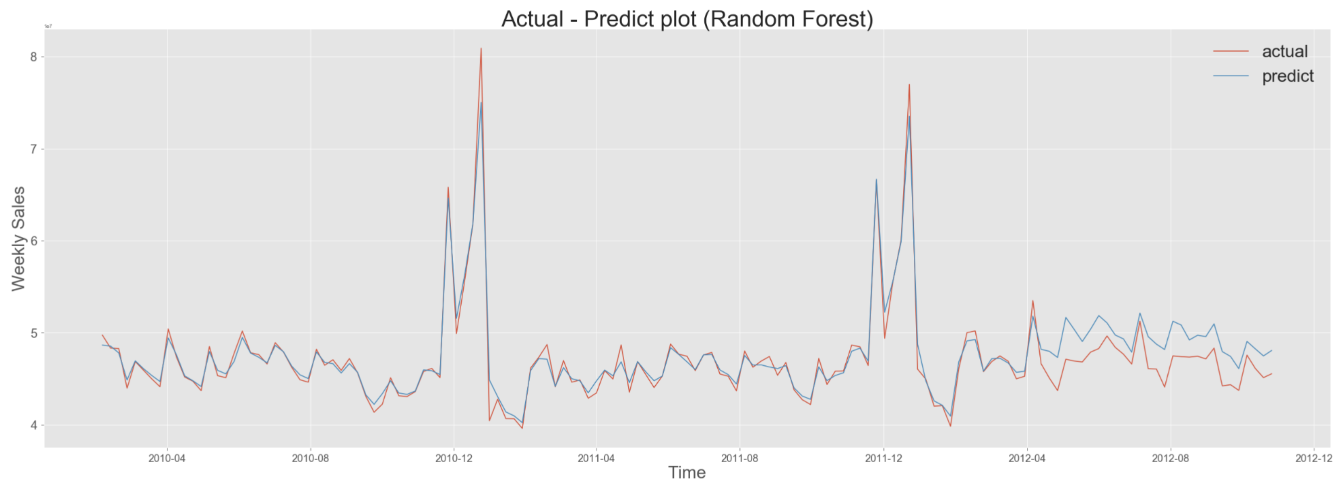
Model Evaluation and Validation

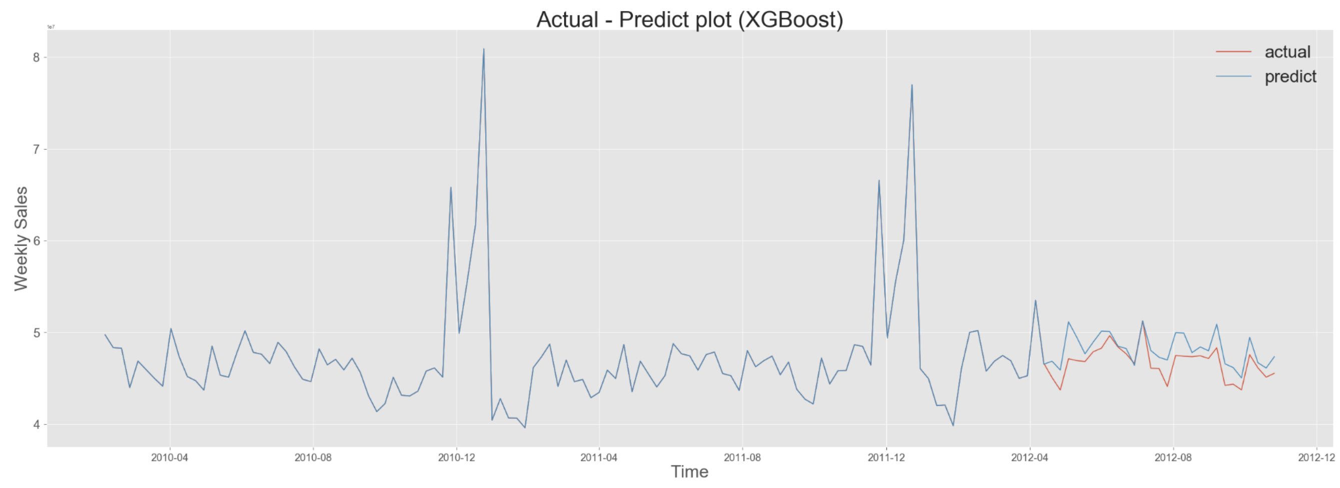
I evaluated all models' performance by using the validation data. The below table shows the model performance(WMAE) of all models.

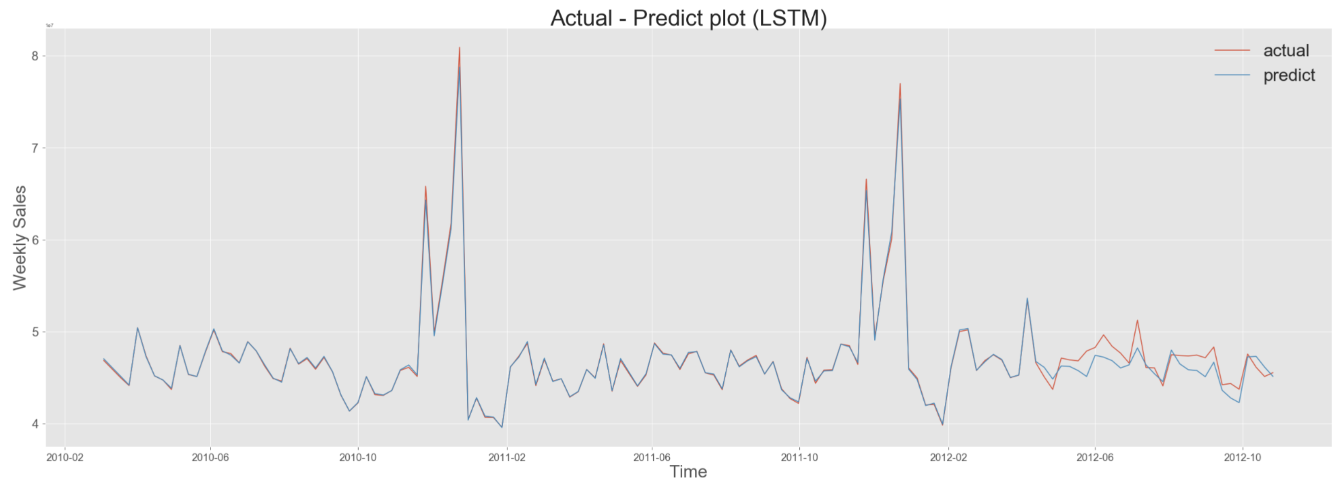
Model	WMAE(on val data)
Decision Tree(BM)	4,431
Random Forest	2,239
XGBoost	2,133
LSTM	1,527

I also checked the aggregated sales chart(the red line shows actual / the blue line shows predict) to check the accuracy of each model's forecast(on train and validation data).









We can see that the models did a good job of fitting both the training and the validation datasets compared to the decision tree result.

WMAE result shows the LSTM model performed the best among all models.

Justification

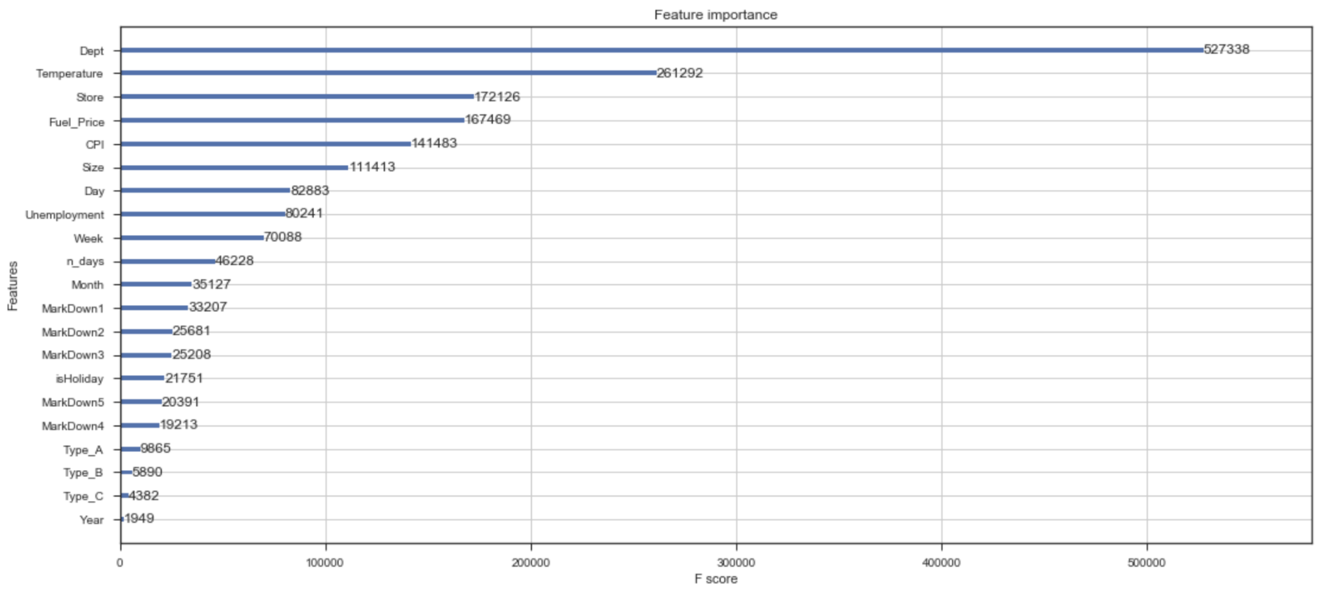
Kaggle withholds weekly sales data in the test data period, but the WMAE score can be confirmed by submitting the forecast on test data to Kaggle. The below table shows the model performance(WMAE) of all models. Random Forest, XGBoost, and The LSTM perform better than the benchmark Decision Tree. Although LSTM model uses only lagged sales data, the model achieved the best WMAE on the testing data.

Model	WMAE(on test data)
Decision Tree(BM)	6,875
Random Forest	4,425
XGBoost	4,288
LSTM	3,278

V. Conclusion

Free-Form Visualization

The below chart created by using xgboost `plot_importance` function shows the feature importance. We can see "Department" has the highest importance and the importance score is significantly high compared to other features. This is one of the reasons the LSTM using only lag sales by each "id"("Store" & "Department") could achieve a good WMAE score.



Reflection

The process used for this project can be summarised with the following steps:

1. Define the objective, the solution, and the metric to evaluate the performance
2. Explore and Analyze the dataset
3. Preprocess the dataset for the machine learning models
4. Train the models on the training data
5. Evaluate the models on the validation data
6. Refine the models by tuning hyperparametes
7. Evaluate the models on the test data

During the process, I was surprised by the LSTM's performance. At first, I thought the XGBoost model with more features would result in better prediction. However, the LSTM showed the best result by using only lag sales.

One disadvantage of the LSTM I felt was that preprocessing the dataset for the LSTM and running the LSTM networks was time-consuming.

I believe the LSTM model is very useful to solve time-series forecast problem in the case there is strong seasonality and trend in time series data.

Improvement

In this project, Basically, I only used original features and skipped basic time-series preprocessing (checking correlogram, making time-series data stationary, etc.) since I had to take a lot of time for hyperparameter tuning(also, I wanted to focus on applying machine learning models). If I were to continue with this project, adding features, such as other economic indicators(GDP growth, consumer confidence index, etc.) is one of the improvements I should explore. Also, conducting basic time-series preprocessing and using a state-space model could help to improve the accuracy since this sales data obviously has seasonality and trend as I mentioned in the Data Exploration section.

References

[1] R. Adhikari and R. K. Agrawal, 'An introductory Study on Time Series Modeling and Forecasting', arXiv:1302.6613, 2013.
[2] K. Bandara, P. Shi, C. Bergmeir, H. Hewamalage, Q. Tran and B. Seaman, 'Sales Demand Forecast in E-commerce using a Long Short-Term Memory Neural Network Methodology', arXiv:1901.04028v2, 2019.