# AI6102 Machine Learning Methodologies & Applications
# Store Item Demand Forecasting Challenge

**Lin Jizhi G2302916F S230063@e.ntu.edu.sg**
**Zhou RuoHeng G2303498K zh0121ng@e.ntu.edu.sg**
**Tang Hui Xin G2303490A htang010@e.ntu.edu.sg**

## Abstract

In this paper, we address the time series forecasting challenge on Kaggle using ARIMA and XGBoost models. We apply specific variations of each model to different types of time series data and conduct a comparative analysis of ARIMA and XGBoost based on their performance.

## 1. Introduction

We participated in the Kaggle Store Item Demand Forecasting Challenge (2018), tasked with predicting the sales volume of 50 items across 10 stores for the first quarter of 2018. Our predictions utilized five years of historical sales data and were evaluated based on the Symmetric Mean Absolute Percentage Error (SMAPE) metric for accuracy.

### 1.1 Problem statement

We developed a machine learning model designed to address seasonal variations by learning patterns from previous sales data. This model predicts the sales of all products across different stores, effectively adapting to seasonal trends.

### 1.2 Time Series Forecasting

A time series is a sequence of data points recorded at consistent time intervals, which may be daily, monthly, or yearly. Time series forecasting involves using statistical models to predict future values based on historical data. This process, known as forecasting, aims to project future values that the series will assume. Effective forecasting of time series is often of great business value, as it helps in making informed decisions based on anticipated trends.

### 1.3 Introduction to XGBoost Models

XGBoost is an efficient machine learning algorithm based on gradient boosting decision trees, emphasizing speed and model performance. It enhances prediction accuracy by sequentially constructing the model and implementing improvements iteratively. Key features of XGBoost include built-in regularization to reduce overfitting, automatic handling of missing values, and support for multiple objective functions. Additional capabilities include tree pruning, built-in cross-validation, and hardware optimization, enabling it to run on both stand-alone and distributed systems. Widely used in data science competitions and real-world applications, XGBoost is favored by many data scientists for its flexibility and efficiency.

### 1.4 Introduction to ARIMA Models

ARIMA, which stands for Autoregressive Integrated Moving Average, is a model that belongs to a class of models that forecast a time series using its own past values, including its lags and the lagged forecast errors. This model is applicable to any non-seasonal time series that exhibits patterns and is not merely random white noise. The underlying principle of ARIMA is that information from historical values of the series can alone be sufficient to predict future values.

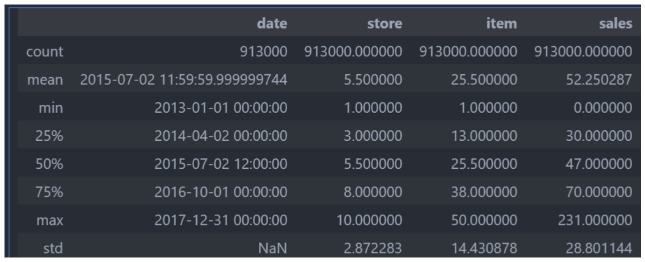
### 1.5 Challenges of problem

- Seasonal and Cyclical Fluctuations: Time series data may exhibit seasonal and cyclical fluctuations, increasing the complexity of forecasting.
- Trend Changes: Time series data may contain trend changes, such as growth or decline trends, which need to be considered in forecasting.
- Data Quality Issues: Data may contain missing values, outliers, or noise, which can affect the accuracy and stability of models.
- Additionally, there may be issues such as nonlinear relationships, overfitting, underfitting, and parameter selection.

## 2. Data Exploration

### 2.1 Statistics

After loading training dataset, the next step is to compute the statistics of this dataset and visualize the dataset to dig out the hidden characteristics, which facilitates the categorization of problem type and the identification of the best model.



| | date | store | item | sales |
|---|---|---|---|---|
| count | 913000 | 913000.000000 | 913000.000000 | 913000.000000 |
| mean | 2015-07-02 11:59:59.999999744 | 5.500000 | 25.500000 | 52.250287 |
| min | 2013-01-01 00:00:00 | 1.000000 | 1.000000 | 0.000000 |
| 25% | 2014-04-02 00:00:00 | 3.000000 | 13.000000 | 30.000000 |
| 50% | 2015-07-02 12:00:00 | 5.500000 | 25.500000 | 47.000000 |
| 75% | 2016-10-01 00:00:00 | 8.000000 | 38.000000 | 70.000000 |
| max | 2017-12-31 00:00:00 | 10.000000 | 50.000000 | 231.000000 |
| std | NaN | 2.872283 | 14.430878 | 28.801144 |

Figure 1: Training Dataset Description

Based on Figure 1 , it can be collected from this table that the distribution of sales is negative skewness, in which mass of sales is concentrated on the right of curve of distribution.

By examination of missing values, no missing value exists in the whole dataset. Because of negative skewness, detection of outliers yields no significant findings.

Furthermore, it is helpful to determine the time gap between the last day in training dataset and the last day in test dataset. This time gap facilitates the construction of lag feature in following feature engineering.

### 2.2 Visualization

To explore training dataset further, establish visualization diagrams for three categories: 1.daily total count of sales; 2.daily total count of sales of each store; 3.daily total count of sales of each item.

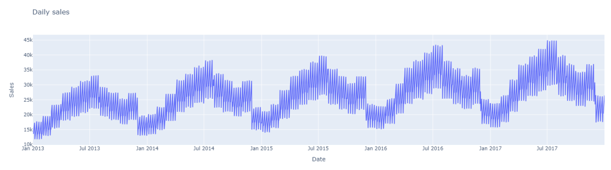
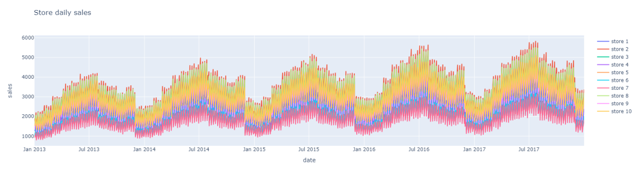
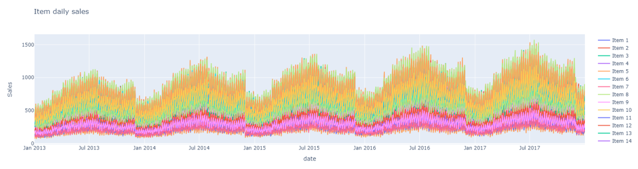


Figure 2: daily sales

In Figure 2, it can be inferred that the dataset has the evident time-series trend, which usually have patterns based on time attributes. There are regular fluctuations inferred by Figure 2: in every year, the sales will increase at the beginning of year from January and reach the peak approximately around July, then decrease to the lowest point gradually until January next year. In addition, by zooming out the diagram, it seems that the sales of every week also shares the similar characteristic: maximize during the middle of week and minimize around the end of week.



(a) group by store



(b) group by item

Figure 3: daily sales of item and store

Based on the visualization of two remaining categories, shown in Figure 3, it can be discovered that they has significant similarity in characteristics of time trend with the first category: daily sales, which means every original feature in training dataset can be solved with similar methods.

### 2.3 Proposed Solutions

It is strongly indicated that this problem belongs to time series problems. Furthermore, there is significant possibility that the training dataset also exhibits lag feature and rolling window feature.

Therefore, the proposed solution involves multiple techniques used in Machine Learning in following step:

(1) Perform elaborate feature engineering from existing original feature.

(2) Apply appropriate Machine Learning models to solve this problem, which are proficient in capturing nonlinear trends more accurately and obtaining more precise forecasts of future values, such as XGBoost, ARIMA.

(3) Utilize cross validation to stabilize the performance of model.

(4) Search the best parameters of model to tune to improve the performance of model.

## 3. Model 1: XGBoost

### 3.1 Mathematics principles behind XGBoost

XGBoost (Extreme Gradient Boosting) is a kind of tree ensemble model which contains a set of classification and regression trees (CART). Because a single tree is not valid, Ensemble model is actually usually used, which sums the prediction of multiple trees together.

During the process of tree boosting in XGBoost, Additive Training is applied because it is intractable to learn all the trees at once. In XGBoost, we fix what we have learned and add one new tree at each step, the derivation shown in the following equations.

$$
\begin{aligned}
&\hat{y_i}^{(0)} = 0 \\
&\hat{y_i}^{(1)} = f_1(x_i) = \hat{y_i}^{(0)} + f_1(x_i) \\
&\hat{y_i}^{(2)} = f_1(x_i) + f_2(x_i) = \hat{y_i}^{(1)} + f_2(x_i) \\
&\hat{y_i}^{(3)} = f_1(x_i) + f_2(x_i) + f_3(x_i) = \hat{y_i}^{(2)} + f_3(x_i) \\
&\vdots \\
&\hat{y_i}^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y_i}^{(t-1)} + f_t(x_i)
\end{aligned}
$$

To obtain the optimal tree at each step, Taylor Expansion of the loss function up to the second order. Therefore, the objective of XGBoost can be defined as follows:

$$
\begin{aligned}
obj^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(x_i)) + \omega(f_t) \\
&\approx \sum_{i=1}^{n} [g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \\
&= \sum_{j=1}^{T} [(\sum_{i \in I_j} g_i) w_j + \frac{1}{2}(\lambda + \sum_{i \in I_j} h_j) w_j^2] + \gamma T \\
&= \sum_{j=1}^{T} [G_j w_j + \frac{1}{2}(\lambda + H_j) w_j^2] + \gamma T
\end{aligned}
$$

- t is step
- $w$ is vector of scores on leaves
- $T$ is number of leaves
- $I_j = \{i | q(x_i) = j\}$, $q$ is a function assigning each data instances to the corresponding leaves, which each data instance is d-dimensional.
- $G_j = \sum_{i \in I_j} g_i, \quad H_j = \sum_{i \in I_j} h_j$

By resolving above quadratic form, the optimal parameters can be obtained:

$$
\begin{aligned}
w_j^{\star} &= -\frac{G_j}{H_j + \lambda} \\
obj^{\star} &= -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T
\end{aligned}
$$

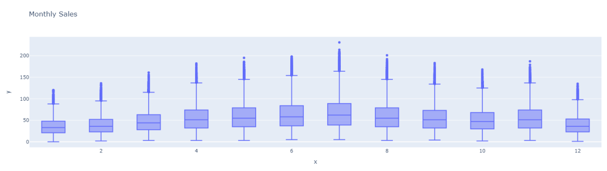
Enumerate all the possible trees at each step is not ideal to pick the best one, XGBoost need to setup criterion to early stop, trying to split leaf into two leaves and measure its gain:

$$
Gain = \frac{1}{2}[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}] - \gamma
$$

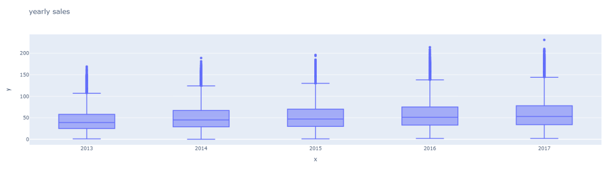
### 3.2 Feature Engineering

**3.2.1 Date Time Features** We divide the original date feature in the original training dataset into more categories of time attributes: year, month, quarter, day of year, day of week, is month start and is month end. The transformation of original date is helpful to convert this problem into supervised time series problem. The purpose behind this way is to facilitate the easy grouping of time-related events for examination. In fact, these can start off simply and head off into quite complex domain-specific areas. The XGBoost model will use these feature to tease out time-of-year or time-of-month type seasonality information.

Therefore, we obtain more time-related feature diagram shown in Figure **??** ,in which we discover more similarity in trends.



(a) Monthly Sales



(b) Yearly Sales

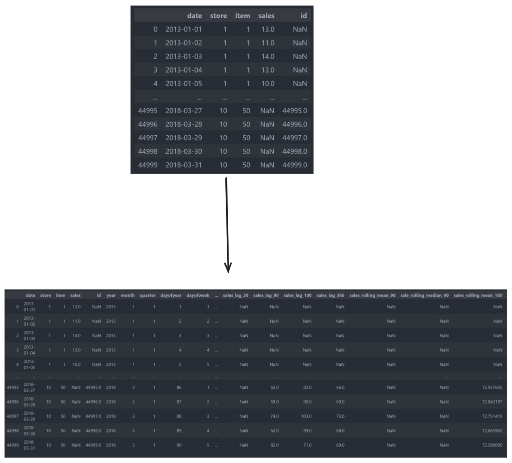
Figure 4: Quartiles of Monthly vs Quartiles of Yearly

**3.2.2 Lag Features** In the realm of time series forecasting, the utilization of lag features represents a conventional methodology whereby such problems are transposed into supervised learning paradigms. This methodology hinges on the prediction of the value at a future time point (t+n) predicated upon the value observed at a preceding time point (t-n), thereby leveraging the inherent periodicity within the data.

We append a month, a year into Lags list, which both show regular fluctuations in Figure 4 In addition. The time gap, between the last day in training dataset and the last day in test dataset, a quarter is also selected as one of Lags.

**3.2.3 Rolling Window Feature** We can calculate summary statistics across the values in the sliding window and include these as features in our dataset. The most useful features of them are mean and standard deviation, also called rolling mean and rolling standard deviation.

In our design, the Rolling Window list is the same as previous Lag list.



| | date | store | item | sales | id |
|---|---|---|---|---|---|
| 0 | 2013-01-01 | 1 | 1 | 13.0 | NaN |
| 1 | 2013-01-02 | 1 | 1 | 11.0 | NaN |
| 2 | 2013-01-03 | 1 | 1 | 14.0 | NaN |
| 3 | 2013-01-04 | 1 | 1 | 13.0 | NaN |
| 4 | 2013-01-05 | 1 | 1 | 10.0 | NaN |
| ... | ... | ... | ... | ... | ... |
| 44995 | 2018-03-27 | 10 | 50 | NaN | 44995.0 |
| 44996 | 2018-03-28 | 10 | 50 | NaN | 44996.0 |
| 44997 | 2018-03-29 | 10 | 50 | NaN | 44997.0 |
| 44998 | 2018-03-30 | 10 | 50 | NaN | 44998.0 |
| 44999 | 2018-03-31 | 10 | 50 | NaN | 44999.0 |

| | date | store | item | sales | id | year | month | quarter | dayofyear | dayofweek | ... | sales_lag_30 | sales_lag_90 | sales_lag_180 | sales_lag_365 | sales_rolling_mean_90 | sale_rolling_median_90 | sales_rolling_mean_180 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2013-01-01 | 1 | 1 | 13.0 | NaN | 2013 | 1 | 1 | 1 | 1 | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 1 | 2013-01-02 | 1 | 1 | 11.0 | NaN | 2013 | 1 | 1 | 2 | 2 | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 2 | 2013-01-03 | 1 | 1 | 14.0 | NaN | 2013 | 1 | 1 | 3 | 3 | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 3 | 2013-01-04 | 1 | 1 | 13.0 | NaN | 2013 | 1 | 1 | 4 | 4 | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 4 | 2013-01-05 | 1 | 1 | 10.0 | NaN | 2013 | 1 | 1 | 5 | 5 | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 44995 | 2018-03-27 | 10 | 50 | NaN | 44995.0 | 2018 | 3 | 1 | 86 | 1 | ... | NaN | 63.0 | 82.0 | 66.0 | NaN | NaN | 72.937642 |
| 44996 | 2018-03-28 | 10 | 50 | NaN | 44996.0 | 2018 | 3 | 1 | 87 | 2 | ... | NaN | 59.0 | 90.0 | 60.0 | NaN | NaN | 72.842197 |
| 44997 | 2018-03-29 | 10 | 50 | NaN | 44997.0 | 2018 | 3 | 1 | 88 | 3 | ... | NaN | 74.0 | 103.0 | 73.0 | NaN | NaN | 72.751419 |
| 44998 | 2018-03-30 | 10 | 50 | NaN | 44998.0 | 2018 | 3 | 1 | 89 | 4 | ... | NaN | 62.0 | 99.0 | 68.0 | NaN | NaN | 72.665902 |
| 44999 | 2018-03-31 | 10 | 50 | NaN | 44999.0 | 2018 | 3 | 1 | 90 | 5 | ... | NaN | 82.0 | 71.0 | 69.0 | NaN | NaN | 72.580000 |

Figure 5: Transformation of Feature Engineering

With the help of these feature engineering methods, Figure 5 shows the conversion of raw dataset into current dataset containing various time-related feature. By analyzing the Correlation of these features, shown in Figure 6, sales and day of week, sale and day of year are observed to have high correlation.

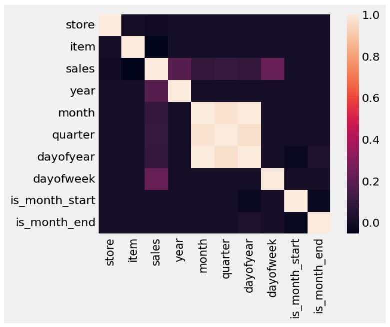


Figure 6: Heatmap of Correlations of Features

## 3.3 Hyper-parameters of Model and Metrics

There are three categories of hyper-parameters of XGBoost model:

1 General Parameters: relate to which booster we are using to do boosting, commonly tree or linear model.

2 Booster parameters: depend on which booster you have chosen.

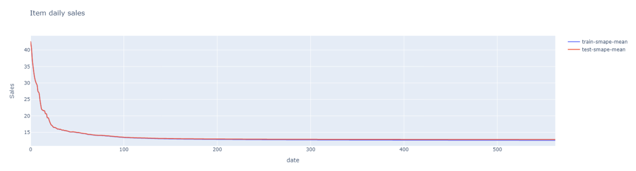
3 Learning task parameters: decide on the learning scenario.

These hyper-parameters shown in Table 1 do not only affects accuracy of our model but also the generalization of model on unseen data. These parameters play key role in following predictions.

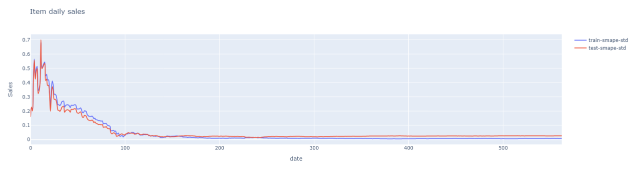
Table 1: Hyper-Parameters of XGBoost

| Parameter | Description | Value |
|---|---|---|
| n_estimators | default=10<br>Number of gradient boosted trees.<br>Equivalent to number of boosting rounds. | 10 |
| learning_rate | default=0.3,<br>alias: learning_rate.<br>step size shrinkage use in update to prevents overfitting. | 0.3 |
| gamma | default=0,<br>alias: min_split_loss<br>Minimum loss reduction required to make a further partition on a leaf node. | 0 |
| max_depth | default=0.6,<br>maximum depth of a tree.<br>0 indicates no limit on depth. | 6 |
| min_child_weight | default=1.<br>Minimum sum of instance weight needed in a child. | 1 |
| subsample | default=1.<br>Subsample ratio of the training instances. | 0.5 |
| colsample_bytree | default=1.<br>is the subsample ratio of columns when constructing each tree. | 0.8 |

## 3.4 Cross Validation



(a) SMAPE mean



(b) SMAPE standard deviation

Figure 7: Evaluation scores with number of tree estimators

We perform a 5-Stratified Fold on our XGBoost model.The Stratified term refers to the method of sampling data such that each fold (subset) of the data set maintains the same class distribution as the original data set. Additionally, because the sequence of data matters in time series, the shuffle is not executed.

Given the requirement of problem, SMAPE is determined as the metric to evaluate the errors between actuals and forecasts. The definition of SMAPE is shown as follows:

$$SMAPE = \frac{100}{n}\sum_{t=1}^{n}\frac{|F_t - A_t|}{(|A_t| + |F_t|)/2}$$

Not only do we apply 5-Stratified Fold to stabilize our model performance, but also we take the early stopping strategy to avoid over-fitting and decrease the cost and complexity. If in the next n rounds the evaluations don't improve, the model will find the optimal number of boosting rounds which is equivalent to number of tree estimators. The following Figure 7 indicates the changes of SMAPE scores with changes of number of tree estimators.

Shown in Figure 7, with the increase of number of estimators, the SMAPE mean and standard deviation between train dataset and validation set gradually converge around $num\_of\_estimators = 500$, which helps to determine the final parameter $num\_of\_estimators$.

### 3.5 Search the optimal Hyper-Parameters to tune

The whole process of tuning XGBoost model can be described in following steps.

1 Determine the number of tree estimators, which we have completed in the Cross Validation section.

2 Determine the learning_rate.

3 Optimization of the max_depth and min_weight parameters.

4 Adjustment of the subsample and colsample_bytree parameters.

We apply the BayesSearchCV method to tune step by step. BayesSearchCV is a parameter tuning method based on Bayesian optimization, typically used for hyper parameter optimization in machine learning models. Bayesian optimization is an iterative optimization method that uses previous model evaluation results to guide the selection of the next set of parameters, aiming to more efficiently search the parameter space.

The whole search parameter grid space shows in the following code block:

```
tuned_params_grid
= {'max_depth': np.arange(3,12,1),
'min_child_weight':
np.arange(0.05, 1.0, 0.05),
'gamma': np.arange(0.0,5,0.05),
'learning_rate':
np.arange(0.05,0.3,0.005),
'subsample': np.arange(0.5,1.0,0.05),
'colsample_bytree':
np.arange(0.5,1.0,0.05),
}
```

Finally, the best parameters after tuning shows in the following Table 2.

And we visualize the final XGBoost model in two aspects: feature importance graph and tree graph, respectively shown as Figure 8 and Figure 9.

Table 2: optimal parameters after tuning

| Parameter | tunned Value |
|---|---|
| n_estimators | 563 |
| learning_rate | 0.22 |
| gamma | 4.7 |
| max_depth | 9 |
| min_child_weight | 0.25 |
| subsample | 0.8 |
| colsample_bytree | 0.7 |

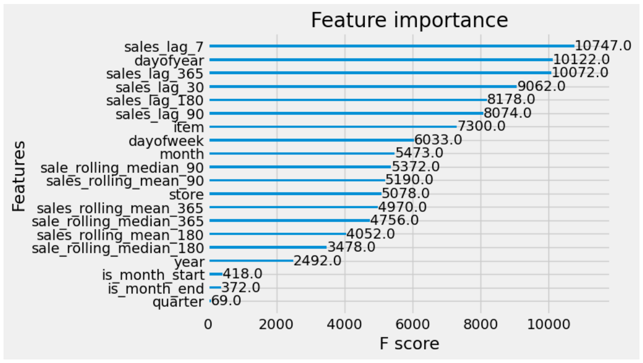


Figure 8: Feature Importance

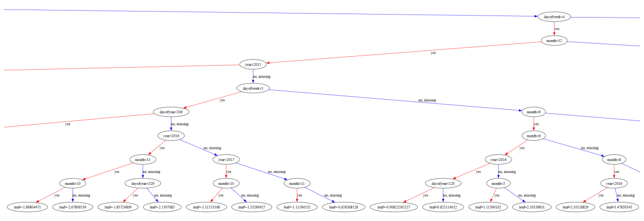
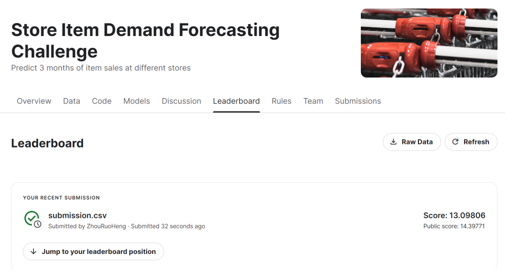


Figure 9: Part of xgboost tree graph

## 6. Submission

**The result of XGBoost model achieved a private Kaggle score of 13.09806 as shown in Figure 10. The leaderboard of this competition is 12.58015. Our XGBoost score places us at 203$^{th}$ out of 461 (Top 44%)**



Store Item Demand Forecasting Challenge

Predict 3 months of item sales at different stores

Leaderboard

YOUR RECENT SUBMISSION

submission.csv

Submitted by ZhouRuoHeng · Submitted 32 seconds ago

Score: 13.09806

Public score: 14.39771

Figure 10: Private Scores of Kaggle

## 4. Model 2: ARIMA

In this section, we will be exploring the possibility of utilising ARIMA - Autoregressive Integrated Moving Average model to tackle the problem at hand.

There are many variations of the ARIMA model, aimed at targeting different types of time series data. To decide on which form of ARIMA to use, we need to take a more in depth look at the data itself. For illustration purposes, we would be zooming in one specific time series from the data and perform seasonal decomposition on it.

### 4.1 Seasonal Decomposition Analysis

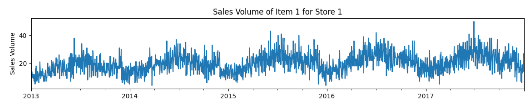


Figure 11: Sales Volume of Item 1 for Store 1

Figure 11 captures the sales volume of item 1 made by store 1 throughout the 5 year period. In order to decide on the components of the ARIMA model, we would need to zoom in further on the different aspects of the data.

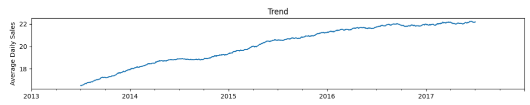


Figure 12: Trend

Figure 12 captures the trend observed in the selected dataset. It is evident that there exists an upward trend in the data. With this understanding, it would make sense to include autoregression in the model to take the trend into account.

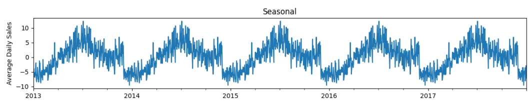


Figure 13: Seasonality

We could clearly see that there is a recurring pattern in the data. From the diagram, there is a fluctuation in sales of close to 10 over the course of a year. Due to the seasonality present in the data, we would have consider variations of ARIMA models which has the capability to analyse seasonal trends.

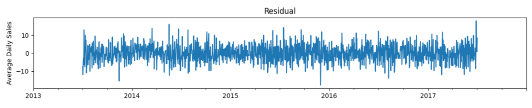


Figure 14: Residual

From the residual plotted, we can tell that there exists outliers in the data. Including a moving average component in our model would help to smooth out the effects of the outliers.

### 4.2 Selection of Model

Due to the seasonality of the data, one option that was considered was the Seasonal Autoregressive Integrated Moving Average (SARIMA) model. SARIMA stems off from ARIMA with the enhanced capability to process seasonal components present in a time series. It brings a new set of hyperparameters, similar to that of a base ARIMA model, which caters to the seasonal component of the data. Additional to that, there is also an additional parameter included to indicate the seasonality period of the data.

After experimenting with analysis using SARIMA however, we identify that there are limitations that comes with SARIMA. Due to the nature of our data set which encompasses daily data, we would need to run SARIMA on a 365 period cycle and that is computationally intensive and expensive. Most machines are unable to handle compute tasks of this sort.

An alternative we have discovered which would be able to handle the 365 period cycle analysis would be the ARIMA but with Fourier Series as the exogenous variable.

### 4.3 Implementation

A Fourier Series is the expansion of a periodic function $f(x)$ in terms of an infinite sum of sines and cosines(Weisstein, Eric W 2024).

In our implementation, we have chosen to generate the Fourier Series using the following formulas:

$$\sin(2 \times \pi \times order/period)$$

$$\cos(2 \times \pi \times order/period)$$

Once the series has been generated, we can then proceed to feed the variable into the ARIMA model for training. For tuning the parameters, we have opted to use the function auto_arima() from package Pmdarima. The auto_arima() function automates the manual fitting of the model and helps to select the best order for the model.

Here is how the model performs against the validation set.

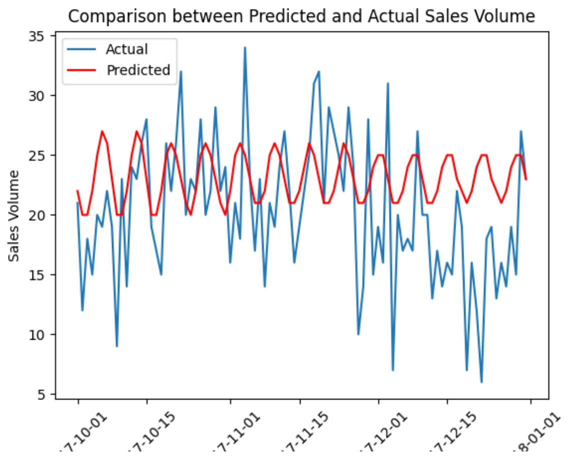


Figure 15: Actual VS Predicted

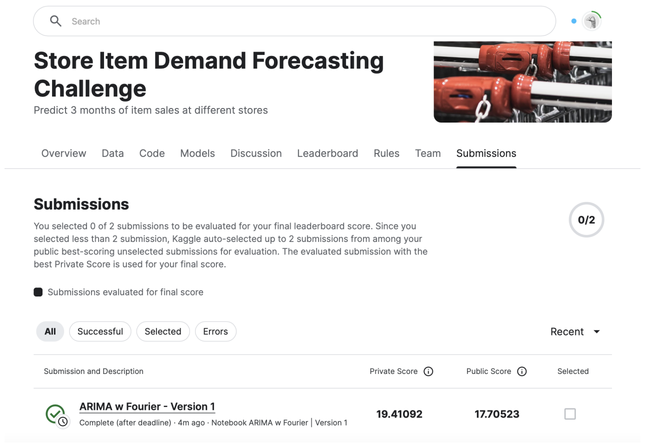
For a more quantitative approach of evaluating our model, we will be computing the Root Mean Squared Error (RMSE) value using the following formula:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}$$

We get a value of **6.3224**.

## 4.4 Result Analysis

After training, we now run the trained models on the given test set and submit the results to Kaggle for assessment. The following figure 16 shows the results provided by Kaggle.



Store Item Demand Forecasting Challenge

Predict 3 months of item sales at different stores

Submissions

You selected 0 of 2 submissions to be evaluated for your final leaderboard score. Since you selected less than 2 submission, Kaggle auto-selected up to 2 submissions from among your public best-scoring unselected submissions for evaluation. The evaluated submission with the best Private Score is used for your final score.

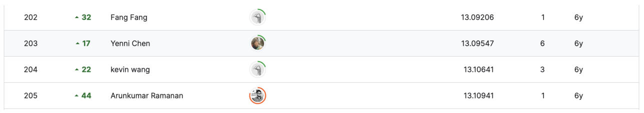
■ Submissions evaluated for final score

| Submission and Description | Private Score | Public Score | Selected |
|---|---|---|---|
| ARIMA w Fourier - Version 1<br>Complete (after deadline) · 4m ago · Notebook ARIMA w Fourier \| Version 1 | 19.41092 | 17.70523 | ☐ |

Figure 16: Kaggle Scores Using ARIMA with Fourier

The model does not fit as well as we were hoping for it to. Possible causes could be due to the rather significant presence of noises in the dataset which could be observed from the frequent spikes in the Residual plot, which ARIMA is not particularly good at dealing with.

# 5. Conclusion



| | | | | | | |
|---|---|---|---|---|---|---|
| 202 | ▲ 32 | Fang Fang | | 13.09206 | 1 | 6y |
| 203 | ▲ 17 | Yenni Chen | | 13.09547 | 6 | 6y |
| 204 | ▲ 22 | kevin wang | | 13.10641 | 3 | 6y |
| 205 | ▲ 44 | Arunkumar Ramanan | | 13.10941 | 1 | 6y |

Figure 17: Training Dataset Description

In this paper, XGBoost outperforms ARIMA in seasonal forecasting, so we ultimately choose the XGBoost model. The leaderboard scores shown in Figure 16 are fixed because the Kaggle competition has ended. However, according to our best performing model with a Kaggle score of 13.09806, we ranked 203rd out of 461 participants. Despite facing challenges during the Kaggle Store Item Demand Forecasting Challenge (2018), it served as a valuable learning opportunity that enhanced our team's understanding and proficiency in data science and machine learning techniques. Here are the main insights we gained from our experience.

## 5.1 The impact of noise.

Noise can have an impact on time series forecasting, especially when its intensity is high. It can make it difficult for the model to capture the true trends and patterns, leading to inaccurate predictions. Therefore, when modeling, it's necessary to employ techniques to reduce the influence of noise, such as smoothing techniques, filters, or using more sophisticated forecasting models.

## 5.2 About using XGBoost

The performance of XGBoost heavily depends on parameter selection, requiring time to adjust parameters for optimal performance. When modeling with XGBoost, it's important to be mindful of overfitting, which can be mitigated through regularization techniques and cross-validation.

## 6. Appendices

### 6.1 Overview of Project Development Process

**The source codes of project can be founded on Github and Kaggle Respectively. The Links are as follows:**

- Github Project Link
- Kaggle XGBoost notebook Link
- Kaggle ARIMA notebook Link

**The Roadmap of Project shows in Figure 18:**

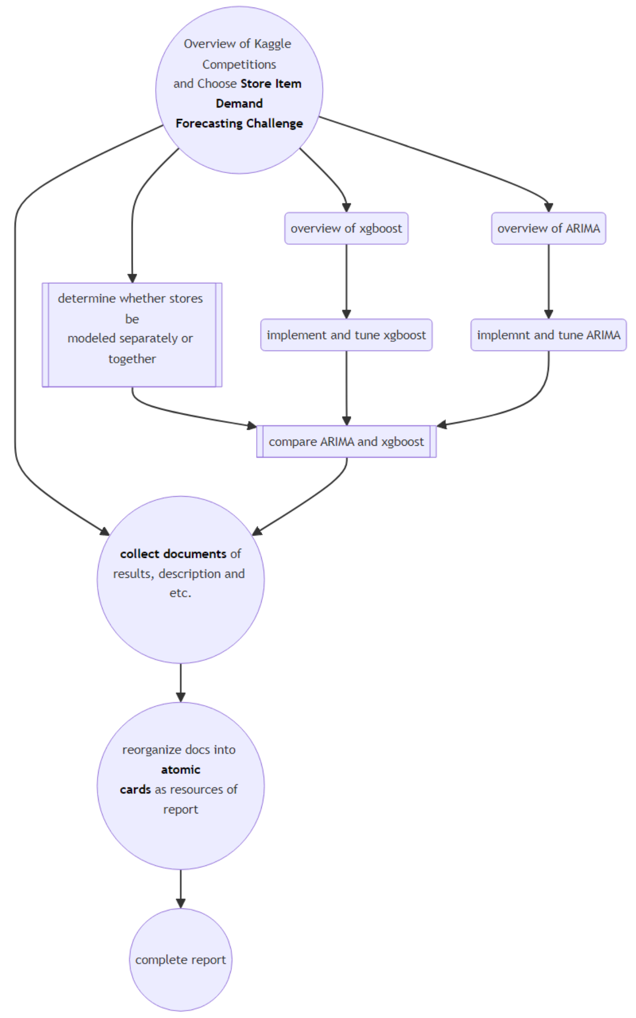
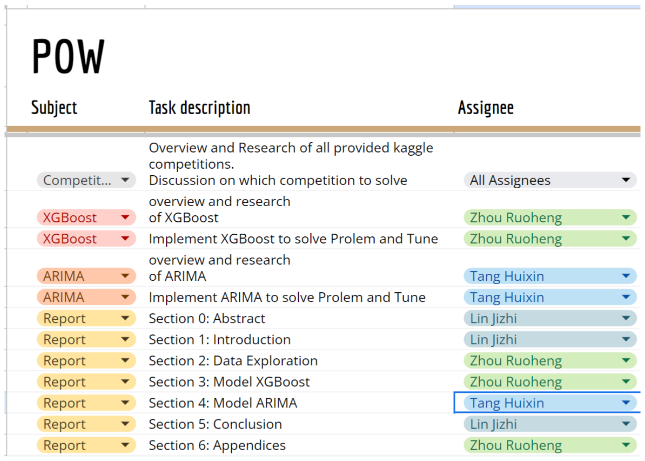

Overview of Kaggle Competitions and Choose Store Item Demand Forecasting Challenge
overview of xgboost
overview of ARIMA
determine whether stores be modeled separately or together
implement and tune xgboost
implemnt and tune ARIMA
compare ARIMA and xgboost
collect documents of results, description and etc.
reorganize docs into atomic cards as resources of report
complete report

Figure 18: Project RoadMap

### 6.2 POW(proof of work)



POW

| Subject | Task description | Assignee |
|---|---|---|
| Competit... | Overview and Research of all provided kaggle competitions. Discussion on which competition to solve | All Assignees |
| XGBoost | overview and research of XGBoost | Zhou Ruoheng |
| XGBoost | Implement XGBoost to solve Prolem and Tune | Zhou Ruoheng |
| ARIMA | overview and research of ARIMA | Tang Huixin |
| ARIMA | Implement ARIMA to solve Prolem and Tune | Tang Huixin |
| Report | Section 0: Abstract | Lin Jizhi |
| Report | Section 1: Introduction | Lin Jizhi |
| Report | Section 2: Data Exploration | Zhou Ruoheng |
| Report | Section 3: Model XGBoost | Zhou Ruoheng |
| Report | Section 4: Model ARIMA | Tang Huixin |
| Report | Section 5: Conclusion | Lin Jizhi |
| Report | Section 6: Appendices | Zhou Ruoheng |

Figure 19: Proof of Work

POW Sheet Online Link

## References

Weisstein, Eric W. 2024. Fourier Series. https://mathworld.wolfram.com/FourierSeries.html. Accessed: 2024-04-25.