Store Item Demand Forecasting

Sean Goh Ann Ray, Shinu Eramangalath Abdulvahab, Phang Kai Ying Eunice
Nanyang Technological University
Singapore 639810

sean0057@e.ntu.edu.sg, shinu001@e.ntu.edu.sg, kphang005@e.ntu.edu.sg

Abstract

In this paper, we propose using a combination of machine learning techniques to tackle a time series forecasting challenge on Kaggle. Specifically, we utilized (1) gradient boosted decision trees (GBDT) to handle the large dataset with categorical features and complex non-linear relationships while avoiding overfitting, (2) 5-fold cross validation, (3) and introduced weighted polynomial curve fitting to the data points to define their importance in the fit. We discuss 4 key experiments as well as their performance and score achieved in the Kaggle competition.

1. Introduction

We participated in the Kaggle Store Item Demand Forecasting Challenge (2018), where we were tasked with predicting the sales volume of 50 items in 10 stores for the first quarter of that year using historical sales data of 5 years. Our predictions were evaluated based on the SMAPE (Symmetric Mean Absolute Percentage Error) metric for accuracy.

1.1. Problem Statement

To develop a machine learning model capable of learning from previous sales data to forecast the sales volume for all products in different stores.

1.2. Challenges of the problem

Based on our analysis of the problem, the following could be some of the potential challenges:

Developing effective models: We will need to experiment with different machine learning algorithms and feature engineering techniques to develop models that perform well on the competition data. This can be time-consuming and require significant computational resources.

Balancing model complexity and interpretability: We will need to balance the complexity of the models with their interpretability. More complex models may achieve higher accuracy but may be difficult to interpret, while simpler models may be easier to understand but may have lower accuracy.

Customer behaviour & external factors: The accuracy of the sales forecasting model can be affected by changes in customer purchasing habits over time. In addition, external factors such as economic conditions, weather, or competition can also influence sales volumes, making it difficult to accurately capture them in the forecasting model.

2. Exploratory Data Analysis

The dataset provided for the Store Item Demand Forecasting Challenge (2018) consists of two datasets - a train dataset and a test dataset. The train dataset contains historical daily sales data spanning five years for 50 different items sold across 10 stores, which totals 913,000 datapoints. The goal is to predict the daily sales of each item in each store for the quarter following the last date in the train dataset, which totals 45,000 datapoints.

2.1. Heatmap

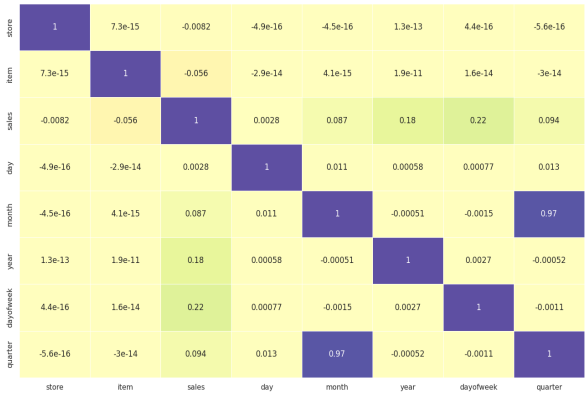

Figure 1: Heatmap of Features

Based on the heatmap, the features generally have weak correlations. However, sales and day of the week, as well as sales and year, are observed to have high correlation.

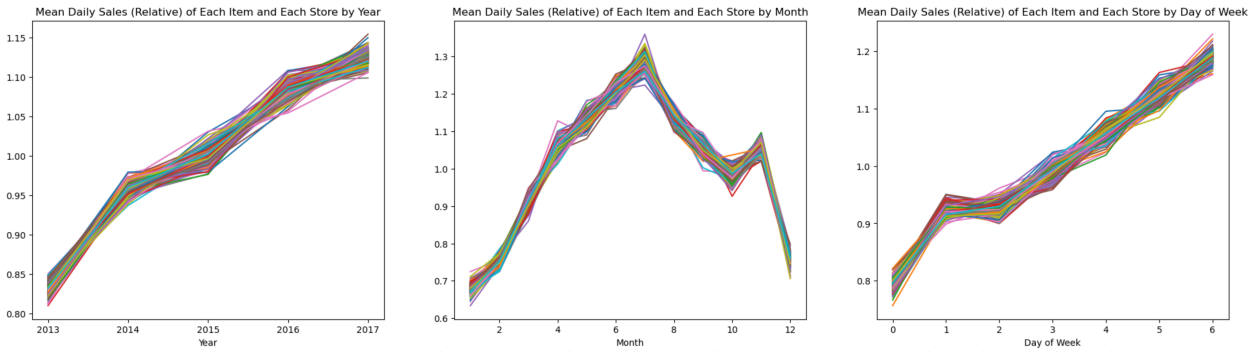

Figure 2: Mean Daily Sales (Relative) of Each Item and Each Store by Year, Month, and Day of Week

2.2. Plots

Time series data usually have patterns based on a time attribute, whether it be day of the week, month, holiday, etc. With the basis that the stores have different number of customers/sales based on its location, but the demand of each item follows a similar pattern, we first obtain the mean daily sales of each of the 50 items in each of the 10 stores, organized by year, month, and day of the week. This produces 3 matrices, named IS_year, IS_month and IS_dow, which are of shape [500, 5], [500, 12] and [500, 7] respectively. The relative mean daily sales of each item and each store are then obtained by dividing it by the mean value across each row, and the plots are shown in Figure 2 above.

A clear increasing sales trend can be observed in the graph plotted by year, but the increase can be seen to be slowing down. This is taken advantaged of and is explained in Section 3.4.

3. Proposed Solution

From Figure 2, the relationship between the features and label is not linear. Therefore, the proposed solution involves multiple techniques used in machine learning methodology. These include:

1. Perform elaborate feature engineering to generate meaningful data from the existing features which helps to capture non-linear trends more accurately and enables precise forecasts of future values.
2. Explore various machine learning models and choose some to experiment on.
3. Utilizing cross validation
4. Fit a polynomial curve to data points whilst introducing weights to each of the datapoints, to define their importance in the fit.

3.1. Feature Engineering

We transformed the date attribute in the initial dataset to generate supplementary features such as day, month, year, day of the week and quarter. Figure 3 shows the transformation of the train dataset. The purpose behind this was to facilitate the easy grouping of time-related events for our examination. This enabled us to study patterns in the data as explained in Section 2, and to compute metrics such as the average daily sales per store per item for each year.

Original Data

date	store	item	sales
2013-01-01	1	1	13
2013-01-02	1	1	11
2013-01-03	1	1	14
2013-01-04	1	1	13
2013-01-05	1	1	10
...	...	...	...
2017-12-27	10	50	63
2017-12-28	10	50	59
2017-12-29	10	50	74
2017-12-30	10	50	62
2017-12-31	10	50	82

913000 rows × 3 columns

Original Data with Additional Date – Related Features

date	store	item	sales	day	month	year	dayofweek	quarter
2013-01-01	1	1	13	1	1	2013	1	1
2013-01-02	1	1	11	2	1	2013	2	1
2013-01-03	1	1	14	3	1	2013	3	1
2013-01-04	1	1	13	4	1	2013	4	1
2013-01-05	1	1	10	5	1	2013	5	1
...	...	...	...	...	...	...	...	...
2017-12-27	10	50	63	27	12	2017	2	4
2017-12-28	10	50	59	28	12	2017	3	4
2017-12-29	10	50	74	29	12	2017	4	4
2017-12-30	10	50	62	30	12	2017	5	4
2017-12-31	10	50	82	31	12	2017	6	4

913000 rows × 8 columns

Figure 3: Feature Engineering on Train Dataset

This process was repeated on the test dataset.

3.2. Model Selection

There are many popular models available for forecasting time series data, which includes linear regression, LSTM, ARIMA, deep neural networks, and decision trees. Considering the nature of the dataset which has:

- Large data volume
- Complex non-linear relationships between the input features and the target variable
- Categorical features

we decided to choose a variation of decision trees, called gradient boosted decision trees (GBDT), mainly for its simplicity and interpretability.

The key difference between decision tree and GBDT is how multiple trees are grown. Decision tree builds a single tree based on the training data, where the error is collated across the test or validation dataset based on a loss function, and a new decision tree will be grown from scratch to minimize the error by adjusting the splitting feature and value. Using an ensemble of decision trees would 'average' the prediction values across all the grown trees.

In GBDTs, an ensemble of trees is grown sequentially. When one decision tree is grown, the next decision tree is grown based on the gradient of errors of the previous tree, instead of using the dataset. Therefore, using an ensemble of GBDT refers to using all the grown decision trees to minimize the error. This enables GBDT to handle large datasets with categorical features and complex non-linear relationships without overfitting.

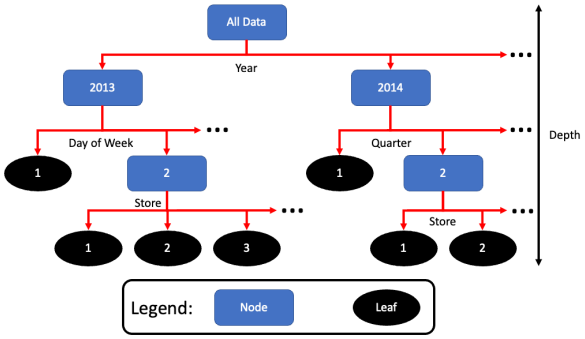

Figure 4: Decision Tree Example

Some common hyperparameters of decision tree models include the depth of the model (Figure 4 shows a depth of 4), the maximum number of leaves, and the number of datapoints per leaf. These hyperparameters affect the accuracy of the model, but also controls the generalization of the model on unseen data.

Two GBDT models were explored, and they are Light Gradient Boosting Machine (LGBM) and Extreme Gradient Boosting (XGB).

The key difference between XGB and LGBM is the way they construct decision trees. XGB uses a depth-first approach to build decision trees, while LGBM uses a leaf-wise approach. The leaf-wise approach can lead to faster training times and lower memory usage, especially on large datasets, but may also lead to overfitting if not properly regularized.

Another difference is that XGB is known to perform well on sparse data, while LGBM is designed to handle categorical features more efficiently. LGBM also includes a built-in feature for handling missing data, which can be useful in some datasets.

While LGBM and XGB are different libraries, they have many overlapping hyperparameters, some of which that were explored are shown in Tables 1 and 2.

Table 1. LGBM Hyperparameters Descriptions

Hyperparameter	**Description**
num_iterations	Maximum number of trees to grow
max_depth	Maximum depth of decision tree
objective	Training objective (to minimize or maximize)
metric	Evaluation metric for validation
num_leaves / max_leaves	Maximum number of leaves in the decision tree
learning_rate	Learning rate of model
bagging_fraction	Portion of data to be 'bagged' without resampling
bagging_freq	Number of iterations before changing the 'bag' of data
lambda_l1	Regularization term for L1 loss
lambda_l2	Regularization term for L2 loss

Table 2. XGB Hyperparameters Descriptions

Hyperparameter	**Description**
num_round	Maximum number of trees to grow
max_depth	Maximum depth of decision tree
objective	Training objective (to minimize or maximize)
eval_metric	Evaluation metric for validation
max_leaves	Maximum number of leaves in the decision tree
eta	Learning rate of model
subsample	Subsample ratio of training instances
colsample_bytree	Subsample ratio of columns when constructing each tree
alpha	Regularization term for L1 loss
lambda	Regularization term for L2 loss

3.3. Cross Validation

A 5-fold cross validation was performed for all experiments conducted. This means that the train dataset was split into 5 subsets at random, and 4 of these subsets will be used for the training with the last subset for validation. This is performed 5 times to shuffle between the subsets, and an average is taken. The random split on the time series data is also possible due to the feature engineering performed as discussed in Section 3.1.

3.4. Polynomial Curve Fitting

With the relative mean daily sales of each item and each store increasing year on year, we can do a preliminary estimation for the year 2018 by fitting a polynomial curve to the mean daily sales of each item and each store by year, then concatenating it to the original matrix (IS_year) to obtain a [500, 6] shaped matrix, with an example shown in Figure 5.

	year	2013.0	2014.0	2015.0	2016.0	2017.0	2018.0
item	store						
1.0	1.0	16.506849	18.873973	20.567123	21.721311	22.183562	23.849986
	2.0	23.369863	26.873973	28.421918	30.459016	31.736986	32.469202
	3.0	20.830137	23.742466	25.043836	27.185792	28.542466	29.707788
	4.0	19.104110	22.049315	23.180822	24.658470	25.695890	26.160630
	5.0	13.950685	15.852055	16.961644	17.978142	18.950685	19.528361
...	...	...	...	...	...	...	...
50.0	6.0	42.736986	48.178082	50.178082	54.781421	56.695890	58.688610
	7.0	38.027397	44.353425	45.983562	49.393443	51.350685	52.345955
	8.0	67.887671	77.369863	81.397260	87.019126	91.169863	93.923699
	9.0	57.687671	67.147945	69.060274	75.344262	77.284932	78.592024
	10.0	61.567123	71.032877	74.194521	80.469945	82.904110	84.498365

500 rows × 6 columns

Figure 5: IS_year with Fitted Values for Year 2018

When fitting a polynomial curve to data points, it is possible to introduce weights to each of the datapoints, to define their importance in the fit. When fitting a curve to time series data, it is important to note that recent data have more influence on future data as compared to past data. A polynomial fit is achieved by minimizing the sum of squared residuals as per equation 1:

$$min \sum_{i=2013}^{2017} [w_i(y_i - \hat{y}_i)]^2 \quad (1)$$

where w_i is the weight of datapoint i, y_i is the value at datapoint i, and $\hat{y}_i$ is the curve fitted value at datapoint i. In unweighted fitting, the value of w_i is simply 1. In our approach, w_i is based on equation 2:

$$w_i = \frac{5 * factor^{2018-i}}{\sum_{i=2013}^{2017} factor^{2018-i}} \quad (2)$$

where factor is a hyperparameter ranging from 0 to 1, and the 5 times multiplier is to ensure that the sum of the weights equals to 5, which is the same as that of an unweighted fit. For illustration purposes, Figure 6 shows the possible polynomial fits to the column-averaged IS_year (relative) matrix.

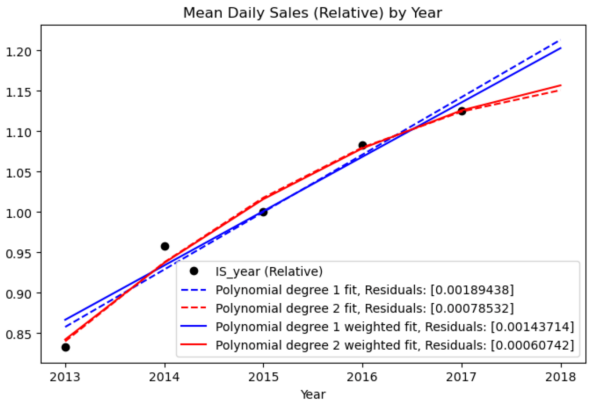

Figure 6: Polynomial Fit Illustration

The weighted fit allows us to place higher importance on recent data, as observed in the polynomial degree 1 fits (blue lines) in Figure 6, where the weighted fit (solid line) is closer to the actual datapoint in the year 2017 as compared to the unweighted fit (dashed line). The same can be said for the polynomial degree 2 fits (red lines) but is not clearly observable. Polynomial fits of higher degrees were avoided as they overfit to the data too much.

Polynomial degree 2 was used as it provided a better fit to the curves, but there were some cases where the extrapolated value for year 2018 may be smaller than that of year 2017. These cases use the polynomial degree 1 fit to follow the strictly increasing trend as seen in Figure 2.

4. Experiment

In this section, we will discuss the performance of several key experiments we conducted to validate the motivations and methodologies stated in Section 3. All experiments discussed utilizes feature engineering of the datasets as well as cross validation. The first model is the baseline GBDT using the LGBM library. The second model builds on top of the first and includes the polynomial curve fitting to scale the daily sales. The third model is the same as the second model but utilizes the XGB library instead. The last model is an ensemble of the second and third model.

Since the predictions of these models are of different scale, we utilized mean absolute percentage error (MAPE) metric from the LGBM to compare our models, as well as the Kaggle score, which is based on SMAPE. MAPE goes by the following equation 3:

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \quad (3)$$

where y_i is the sales value and $\hat{y}_i$ is the predicted sales value. Using this percentage-based metric allows a fair comparison instead of another metric, such as L1 or L2 loss, which uses the immediate difference in values.

4.1. Model 1 – Baseline LGBM

The baseline model uses a 5-fold cross validation on GBDT model using the LGBM library, as discussed in Sections 3.2 and 3.3. Additionally, feature engineering was performed on the datasets to produce the day (of the month), month, year, day of week and quarter (of the year) features. This is shown in Figures 7 and 8 for the train dataset and the test dataset respectively.

	store	item	day	month	year	dayofweek	quarter	sales
0	1	1	1	1	2013	1	1	13
1	1	1	2	1	2013	2	1	11
2	1	1	3	1	2013	3	1	14
3	1	1	4	1	2013	4	1	13
4	1	1	5	1	2013	5	1	10
...	...	...	...	...	...	...	...	...
912995	10	50	27	12	2017	2	4	63
912996	10	50	28	12	2017	3	4	59
912997	10	50	29	12	2017	4	4	74
912998	10	50	30	12	2017	5	4	62
912999	10	50	31	12	2017	6	4	82

Features | Target

Figure 7: Train Dataset for Baseline GBDT

	store	item	day	month	year	dayofweek	quarter
0	1	1	1	1	2018	0	1
1	1	1	2	1	2018	1	1
2	1	1	3	1	2018	2	1
3	1	1	4	1	2018	3	1
4	1	1	5	1	2018	4	1
...	...	...	...	...	...	...	...
44995	10	50	27	3	2018	1	1
44996	10	50	28	3	2018	2	1
44997	10	50	29	3	2018	3	1
44998	10	50	30	3	2018	4	1
44999	10	50	31	3	2018	5	1

Features

Figure 8. Test Dataset for Baseline LGBM

Several experiments were conducted to find the combination of hyperparameters that provided the smallest metric error, and they are shown in Figure 9 below. The factor hyperparameter of the weight scaling (discussed in Section 3.4) that produced the smallest error was found to be 0.85.

```
params = {
    'max_depth': 6,                 # Maximum depth of decision tree
    'boosting_type': 'gbdt',        # Gradient Boosting Decision Trees
    'objective': 'regression_l2',   # To minimise the L2 loss
    'metric': 'rmse',               # Use rmse metric on validation
    'num_leaves': 32,               # Maximum number of leaves in decision tree
    'learning_rate': 0.1,           # Learning Rate
    'bagging_fraction': 0.6,        # Bagging Fraction
    'bagging_freq': 5,              # Bagging Frequency
    'lambda_l1': 0.12,              # Lambda of L1 Regularization
    'lambda_l2': 0.1,               # Lambda of L2 Regularization
}
```

Figure 9: Hyperparameters of LGBM Training

The average MAPE on the cross validation is 13.37%. The predicted sales of the test dataset were averaged across the 5 folds and then rounded to the nearest integer, based on the assumption that the items being sold cannot be sold in fractional portions. The prediction was then submitted to Kaggle, which achieved a score of 12.97494 as shown in Figure 10.

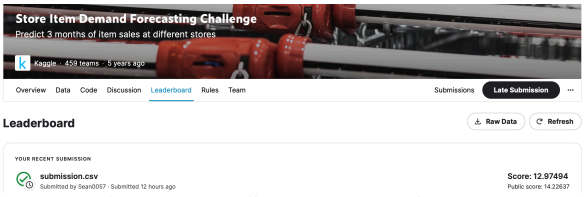

Figure 10: Baseline LGBM Kaggle Score

4.2. Model 2 – LGBM with Polynomial Scaling

The baseline LGBM model predicts the daily sales directly. The idea of this model was to predict the relative daily sales with respect to the yearly average instead of directly predicting the daily sales itself. This is done as the train dataset does not have any information for the year 2018, and doing the prediction directly may lead to inaccuracies. With the year-based scaling, some of the inaccuracies may be resolved.

Developing on the baseline model, a polynomial curve fitting is performed on the IS_year data as described in Section 3.4. The mean daily sales of each item and each store can be used to scale down the sales value for different years, then scaled back up for the predicted sales value for the year 2018. The training data is shown in Figure 11, where the sales (target) are scaled down by the isy_mean column. The isy_mean values are based on the IS_year matrix, are only used for scaling and are not part of the training features used in the GBDT model training.

The test data for this model is shown in Figure 12. It is the same as that in the baseline GBDT discussed in the previous subsection, with the additional isy_mean column used for scaling up the predicted sales for the year 2018.

	store	item	day	month	year	dayofweek	quarter	sales	isy_mean
0	1	1	1	1	2013	1	1	0.787552	16.506849
1	1	1	2	1	2013	2	1	0.666390	16.506849
2	1	1	3	1	2013	3	1	0.848133	16.506849
3	1	1	4	1	2013	4	1	0.787552	16.506849
4	1	1	5	1	2013	5	1	0.605809	16.506849
...	...	...	...	...	...	...	...	...	...
912995	10	50	27	12	2017	2	4	0.759914	82.904110
912996	10	50	28	12	2017	3	4	0.711666	82.904110
912997	10	50	29	12	2017	4	4	0.892597	82.904110
912998	10	50	30	12	2017	5	4	0.747852	82.904110
912999	10	50	31	12	2017	6	4	0.989095	82.904110

Features Target Scaling

Figure 11: Training Data for LGBM with Scaling

	store	item	day	month	year	dayofweek	quarter	isy_mean
0	1	1	1	1	2018	0	1	23.849986
1	1	1	2	1	2018	1	1	23.849986
2	1	1	3	1	2018	2	1	23.849986
3	1	1	4	1	2018	3	1	23.849986
4	1	1	5	1	2018	4	1	23.849986
...	...	...	...	...	...	...	...	...
44995	10	50	27	3	2018	1	1	84.498365
44996	10	50	28	3	2018	2	1	84.498365
44997	10	50	29	3	2018	3	1	84.498365
44998	10	50	30	3	2018	4	1	84.498365
44999	10	50	31	3	2018	5	1	84.498365

Features Scaling

Figure 12: Test Dataset for LGBM with Scaling

The same hyperparameter settings were initially used (as per Figure 9) to see performance improvements directly based on the MAPE and Kaggle score. After several experiments to adjust the hyperparameters, it was found that the initial settings achieved the best improvements and Kaggle score for this LGBM model with polynomial curve fit scaling.

The average MAPE on the cross validation is 10.78%, which is an improvement over the baseline model. The predicted sales of the test dataset were first averaged across the 5 folds, followed by scaling up based on the respective isy_mean values and then rounded to the nearest integer. This was then submitted to Kaggle, which achieved a score of 12.61699 as shown in Figure 13.

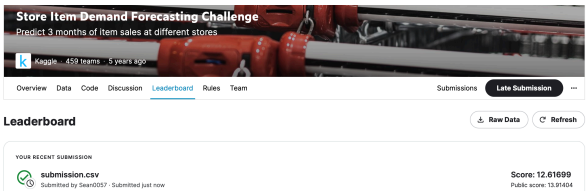

Figure 13: Improved LGBM Kaggle Score

4.3. Model 3 – XGB with Polynomial Scaling

The datasets from Section 4.2 were reused to train a model similar to that in Section 4.2, but utilizes the XGB library instead of the LGBM library. Several experiments were conducted to find the best hyperparameter settings which gives the smallest metric error and best Kaggle score, and they are shown in Figure 14.

```
params = {
        'num_round':1000, # Number of boosting rounds
        'max_depth': 6, # Maximum depth of a tree
        'eval_metric': 'mape', # Mean Absolute Percentage Error
        'subsample': 0.8, # Subsample ratio of the training instances.
        'colsample_bytree': 0.8, #Subsample ratio of columns when constructing each tree.
        'objective': 'reg:squarederror', # Regression type objective functions
        'eta': 0.08,  # learning rate
        'alpha': 0.1,   # L1 regularization parameter
        'lambda': 0.1   # L2 regularization parameter
    }
```

Figure 14: Hyperparameters of XGB Training

The average MAPE on the cross validation is 13.08%. The predicted sales of the test dataset were first averaged across the 5 folds, followed by scaling up based on the respective isy_mean values and then rounded to the nearest integer. This was then submitted to Kaggle, which achieved a score of 12.61769 as shown in Figure 15.

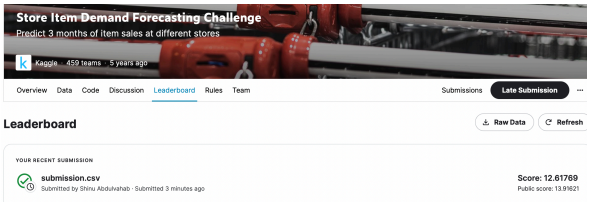

Figure 15: Kaggle Score for XGB

4.4. Ensemble of LGBM and XGB

The final model simply takes an ensemble of models 2 and 3 to create the final predictions. It involves combining the predictions of models 2 and 3 to improve overall performance. In model averaging, the predictions of each model are weighted and then averaged together to produce a final prediction. The code to merge the results is shown in Figure 16.

```
# Take the average of the predictions
ensemble_pred = (pred_lgbm + pred_xgb_fold ) / 2
# Store the predictions
out_df = pd.DataFrame({'id': ID.astype(np.int32), 'sales': ensemble_pred})
#out_df['sales'] = out_df['sales'].round()
out_df.to_csv('submission_ensemble.csv', index=False)
out_df.head()
```

Figure 16: Averaging the predictions.

Uniform weighting was applied for each model's prediction as they performed very similarly in the Kaggle score achieved. The result of this model achieved a private Kaggle score of 12.61624 as shown in Figure 17, an improvement to the score achieved by models 2 and 3.

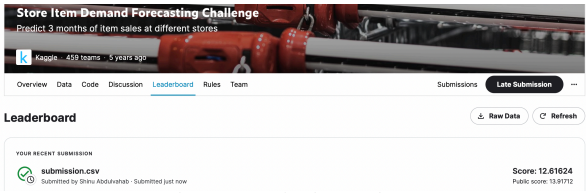

Figure 17: Final Kaggle score

5. Conclusion

The leaderboard score shown in Figure 18 below is fixed as the Kaggle competition is a completed one. But based on our best performing model of Kaggle score 12.61624 places us at 71st out of 461 (Top 15.5%).

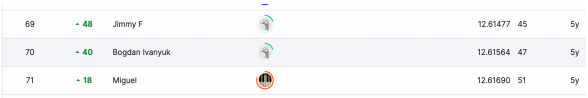

Figure 18: Kaggle Leaderboard

Despite encountering difficulties in the Kaggle Store Item Demand Forecasting Challenge (2018), it was a valuable learning experience that improved our group's knowledge and skills in data science and machine learning methodologies. Here are the key takeaways from what we learned.

5.1. Prediction of Sales vs Relative Sales

As per Sections 4.1 and 4.2, predicting the relative sales instead of the sales itself shows an improvement in both MAPE and Kaggle scores. This could be due to the combination of utilizing the preliminary prediction of the polynomial curve fitting and having the model work with smaller target values instead of large values.

5.2. Ensemble Learning

Ensemble learning is easily understood to improve model performance in classification problems. Here, we have observed how it can help improve solutions to time series regression problems by combining multiple models which perform relatively well individually, as described in Section 4.4.

References

[1] Brownlee, J. (2022). Time Series Forecasting. Retrieved April 2023, from https://machinelearningmastery.com/time-series-forecasting/.

[2] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. Ann. Statist. 29 (5)1189 - 1232, October 2001