Store Item Demand Forcasting Challenge

Dilukshi Charitha Dissanayake
Department of Computer Science
Blekinge Institute of Technology
Karlskrona, Sweden
didi18@student.bth.se

***Abstract*—This project provides a solution to the store item demand forecasting challenge provided in Kaggle using Xgboost (eXtreme Gradient Boosting) model while performing an automatic hyperparameter optimization.**

Keywords—Time series, Machine Learning, XgBoost

I. Introduction

The time series problem I have selected is to predict 3 months sales for 50 items in 10 different stores which was available in Kaggle competitions [1]. However, to reduce the data file size and to reduce the training time I have selected only 20 items in 5 different stores. The problem I have selected is a supervised learning multi variate time series problem. In my project I have provided a solution using Xgboost algorithm. I have used root mean square error(rmse) to compare my solution with few solutions that have been already provided in Kaggle[2][3].

II. Related work

Time series forecasting has been helpful in many decision makings in large business. Early days it was done manually by stakeholders. However, due to the huge amount of data and complexity of the businesses, many have already focused on automating the process of time series forecasting. Due to the automation of this process there are many models, Machine learning models have been implemented. Few such popular models are, Long short-term memory (LSTM), ARIMA, AdaBoost, XgBoost etc. Many past studies have highlighted that Xgboost has outperformed many other models in time series forecasting[4][5]. Furthermore, many award winning solutions in kaggle have used Xgboost as their model in solving time series problems.[6] Therefore, I have selected Xgboost in solving this time series challenge.

Xgboost is also known as one of the most fastest and accurate models that as being widely used.[6] Scalability factor in Xgboost helps to run much faster than many existing popular solutions. The scalability of XGBoost is due to several important systems and algorithmic optimizations. Since I was lacking on high performing hardware components I needed an model which could perform much faster on any time series problem. Therefore, Xgboost seems to be the most suitable model for my project as evident from many Kaggle competitions as well as from research studies I have went through.

Furthermore, many solutions in the Kaggle have used LSTM, ARIMA models and the solutions that have used xgboost is in poor quality. Which suggests a more improved model of Xgboost. Therefore, I have used automatic hyperparameter optimization technique to improve the results from Xgboost.

III. Methodology

A. Dataset

The original dataset was 18MB in size which included 50 items in 10 different stores sales for 5 years. In order to reduce the dataset size as well as to reduce the training speed I have selected sales for 20 items in 5 different stores for 3 year expansion.

B. Xgboost (eXtreme Gradient Boosting)

Xgboost uses gradient boosting decision tree algorithm. Boosting is an ensemble technique which corrects the errors made by existing models by adding new models. New models are added sequentially until no more improvements can be made.

Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It is called gradient boosting since it uses a gradient descent algorithm to minimize the loss when adding new models.

XGBoost improves upon the base GBM(Gradient Boost Machines) framework through systems optimization and algorithmic enhancements. It achieves system optimization by Parallelization, tree pruning and hardware optimization. Algorithm enhancements are done by Regularization, Sparsity Awareness, Weighted Quantile Sketch and Cross-validation.

Xgboost achieves sequential tree adding using parallelized implementation. Furthermore, XGBoost uses 'max_depth' parameter as specified instead of criterion first, and starts pruning trees backward. This 'depth-first' approach improves computational performance significantly. In order to achieve efficient use of hardware resources Xgboost introduces cache awareness by allocating internal buffers in each thread to store gradient statistics. Further enhancements such as 'out-of-core' computing optimize available disk space while handling big data-frames that do not fit into memory.

In order to prevent overfitting Xgboost uses both L1(Lasso) and L2(Ridge) regularization in penalizing more complex models. XGBoost admits sparse features for inputs by automatically learning best missing value depending on training loss and handles different types of sparsity patterns in the data more efficiently. XGBoost also employs the distributed weighted Quantile Sketch algorithm to effectively find the optimal split points among weighted datasets. The algorithm comes with built-in cross-validation method at each iteration, taking away the need to explicitly program this search and to specify the exact number of boosting iterations required in a single run. [6][7]

C. Root Mean Squared Error as the messurement

In order to compare each model that I have used I used Root Mean Squared Error(RMSE) which is given by following equation.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \hat{x}_i)^2}{N}}$$

$x_i -$ *actual observations*

$\hat{x}_i -$ *estimated time series*

$N -$ *number of non missing data points*

$i -$ *variable i*

In my implementation I have used the built-in function mean_squared_error and passed parameter 'squared = false'. The reason for selecting RMSE was due to its popularity in comparing model's accuracy when it comes to time series problems. Furthermore, I wanted to observe how accurate my solution is as compared to other models. If RMSE returns a lesser value as compared to other models it shows that, that model performs much accurately as compared to the other models.

D. Dataset preprocssesing

The first step of the process was to see if there are any null values in the dataset. If there exists any, I have removed the raw which consist those null values. However, in my dataset there were no null values. Furthermore, I had to format the date column to recognize it as date values.

E. Exploratory data Analysis(EDA)

I have used python matplotlib library to plot the sales based on each store and items (Fig. 1., Fig. 2.). Based on both plots I was able to observe a seasonality in the sales.

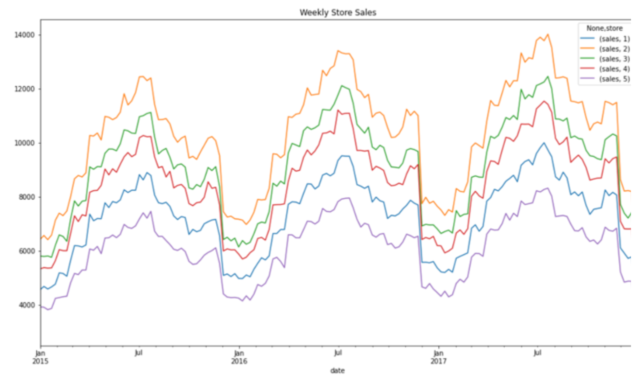


Fig. 1. Weekly store sales.

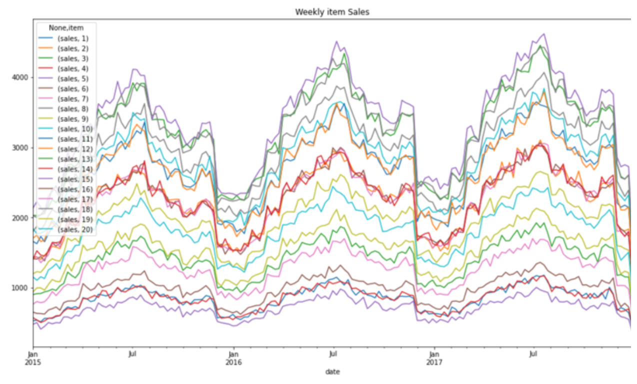


Fig. 2. Weekly item sales

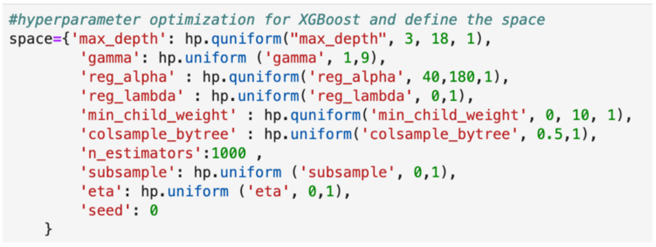
F. Implementation

First, I have rearranged the dataset so that I could use shifting methods. Then I have implemented a function called series_to_supervised to convert the series to a supervised learning problem which is similar to many time series problems. I have used python shift() method to pushed columns forward while giving window size as 29 and lag size as 90. This will be used to forecast 90days ahead. Then I have dropped rows that are not equal to the shifted items and stores. Next, I removed the columns that are not needed such as item(t) and store(t).

Then I have spilt the data set into training and testing, 60% for training and 40% testing.

G. Hyperparameter Optimazation for Xgboost

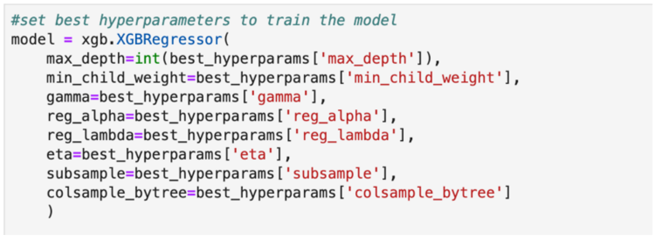
The next step in my implementation was to optimize the hyperparameters that I would pass to the model to be trained. In order to find the best hyperparameters I have built two functions by defining space and objective.



```
#hyperparameter optimization for XGBoost and define the space
space={'max_depth': hp.quniform("max_depth", 3, 18, 1),
        'gamma': hp.uniform ('gamma', 1,9),
        'reg_alpha' : hp.quniform('reg_alpha', 40,180,1),
        'reg_lambda' : hp.uniform('reg_lambda', 0,1),
        'min_child_weight' : hp.quniform('min_child_weight', 0, 10, 1),
        'colsample_bytree' : hp.uniform('colsample_bytree', 0.5,1),
        'n_estimators':1000 ,
        'subsample': hp.uniform ('subsample', 0,1),
        'eta': hp.uniform ('eta', 0,1),
        'seed': 0
    }
```

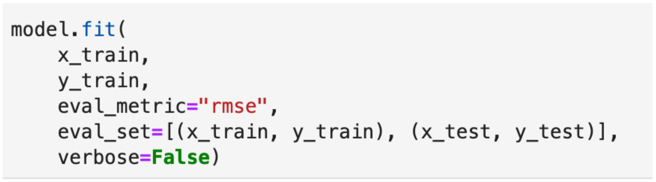
Fig. 3. Parameters to be optimized

Once the best hyperparameters been generated I have passed them to my Xgboost model to be trained.



```
#set best hyperparameters to train the model
model = xgb.XGBRegressor(
    max_depth=int(best_hyperparams['max_depth']),
    min_child_weight=best_hyperparams['min_child_weight'],
    gamma=best_hyperparams['gamma'],
    reg_alpha=best_hyperparams['reg_alpha'],
    reg_lambda=best_hyperparams['reg_lambda'],
    eta=best_hyperparams['eta'],
    subsample=best_hyperparams['subsample'],
    colsample_bytree=best_hyperparams['colsample_bytree']
    )
```

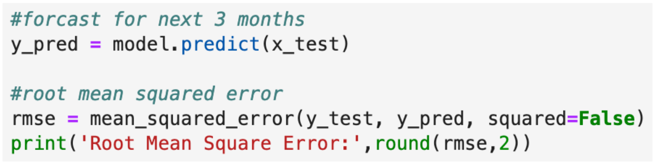
Fig. 4. Parse best parameters to the model



```
model.fit(
    x_train,
    y_train,
    eval_metric="rmse",
    eval_set=[(x_train, y_train), (x_test, y_test)],
    verbose=False)
```

Fig. 5. Fit the model

Then I have predicted the test data as following and calculated the root mean squared error(fig.6.).



```
#forcast for next 3 months
y_pred = model.predict(x_test)

#root mean squared error
rmse = mean_squared_error(y_test, y_pred, squared=False)
print('Root Mean Square Error:',round(rmse,2))
```

Fig. 6. Prediction and RMSE

In order to give different comparison rather than shifting methods, I have implemented another model where I have divided the data set into training and testing while giving the testing set the last 3 months. Then I have optimized the hyperparameters as same in the first model and trained the xgboost model. Results are reported in next section.

Both models (model 1 and model 2) are implemented under the same experimental setup by using the same training and testing data sets as well as setting 'seed=0' to reproduce the data. However, ARIMA model was implemented using different training dataset while LSTM and MLP uses same. ARIMA model considers the entire training data set while LSTM, MLP, model 1 and model 2 uses only a subset of the entire dataset so that it would reduce the time of training. However, every model has used the same testing data set which is last 3 months of the entire dataset (2017.10.01-2017.12.31).

IV. Results and Analysis

In this section I have reported the results I have collected from both models and in the final part of this section I will be comparing my models with other solutions in Kaggle.

TABLE I. MODEL 1 AND MODEL 2 RESULTS

Measurement	**Model**	
	Model 1	***Model 2***
RMSE	19.03	14.41
Time(min)	08:23	01:56

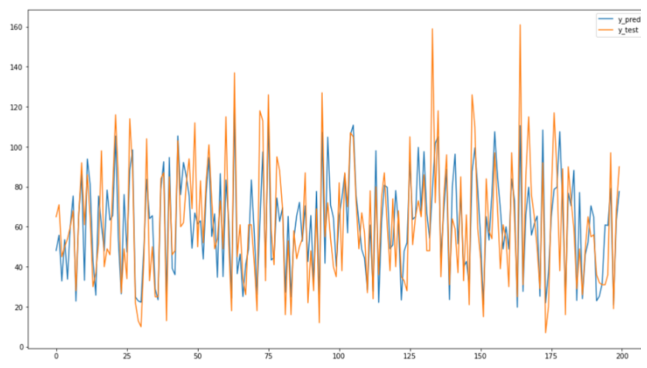


Fig. 7. Model 1 prediction vs actual

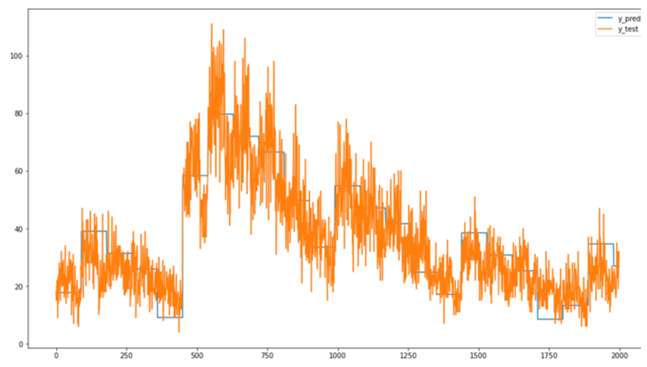


Fig. 8. Model 2 prediction vs actual

Based on the results I have received, model 1 has higher rmse as compared to model 2. Furthermore, model 1 takes more time in training the model than model 2. It indicates that model 2 performs better than the model 1. However, when I have plotted the actual vs predictions for both models, model 1 has a pattern much similar to actual while on model 2, model patterns indicate some constant values for each range. These two models indicates that there could be more improvements that I could have done to improve the results.

TABLE II. MODEL COMPARISON

Measurement	**Model**				
	Model 1	***Model 2***	***ARIMA***	***LSTM***	***MLP***
RMSE	19.03	14.41	19.60	19.97	18.35

The table 2 contains the results I have abstracted from 3 different solutions that have been submitted as best solutions as the time, ARIMA, LSTM, MLP [2][3]. As seen on the table model 2 outperforms all other models according to the rmse values. My model 1 performs much closer to the other suggested models. However, MLP model have performed much better than my model 1, ARIMA and LSTM though it is not a significant difference.

V. Conclusions

As seen from the results and analysis, model 2 which as implemented using xgboost without shifting methods have performed better as compared to other models. However, there are many more ways to improve the results of that model. Such as removing seasonality before training the model. Which I believe something that I have missed in improving the models. Also, while comparing the models I do believe taking the same training size, testing size and same data sequence should have been considered. However, due to time constrains I was unable to consider all the aspects. Furthermore, my model results could have been more different as compared to the other solutions which are on Kaggle if I have used the whole data set(training and testing). However, the ARIMA model which uses the entire dataset does not perform better as compared to my models. Which highlights Xgboost model is much better as compared to the ARIMA model in solving this time series problem. In the conclusion, my models performance could have been improved much more even though they still seems to outperform other solutions in kaggle.

References

[1] "Store Item Demand Forecasting Challenge." https://kaggle.com/c/demand-forecasting-kernels-only (accessed Jan. 12, 2021).

[2] "Deep Learning for Time Series Forecasting." https://kaggle.com/dimitreoliveira/deep-learning-for-time-series-forecasting (accessed Jan. 12, 2021).

[3] "Time Series - ARIMA, DNN, XGBoost Comparison." https://kaggle.com/enolac5/time-series-arima-dnn-xgboost-comparison (accessed Jan. 12, 2021).

[4] "Air Pollutant Concentration Prediction using Ensemble of Machine Learning Techniques," 2018. /paper/Air-Pollutant-Concentration-Prediction-using-of/79964c82c128094e37e2c8fa9b91ce3da0bccaeb (accessed Jan. 12, 2021).

[5] M. Alim, G.-H. Ye, P. Guan, D.-S. Huang, B.-S. Zhou, and W. Wu, "Comparison of ARIMA model and XGBoost model for prediction of human brucellosis in mainland China: a time-series study," *BMJ Open*, vol. 10, no. 12, p. e039676, Dec. 2020, doi: 10.1136/bmjopen-2020-039676.

[6] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, Aug. 2016, doi: 10.1145/2939672.2939785.

[7] V. Morde, “XGBoost Algorithm: Long May She Reign!,” *Medium*, Apr. 08, 2019. https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d (accessed Feb. 12, 2021).