# Future Sales Forecasting with Multiple Machine Learning Approaches

**Data Mining Term Project - Final Report**

**Team members**: Haotian Wang, Hsiao-Yuan Chen, Di Wu, Junjie Tang, Xiang Liu

## Abstract

Proposes a combination of machine learning approaches for a sales forecasting case. With 1C company's historical dataset, this paper conducts in-depth analysis of data attributes and heuristically combines features. After applying traditional regression and popular ensemble algorithms, it carefully evaluates results and uses intelligent tuning methods to find the optimal model. This paper offers insights on data preparation and model selection. Eventually, the prediction result is among the best.

## 1. Introduction

Sales forecasting has always been an indispensable and critical part in the commercial enterprises. It is based on past historical data to predict future sales and is the foundation for companies to make future plans and adjust their budgets (Lancaster & Reynolds, 2002). Research has shown that sales forecasting improves performance of the supply chain (Bayraktar er al., 2008; Zhao et al., 2002). Machine learning methods have been extensively researched and used in large amounts of sales forecasting. However, when dealing with real-world cases, analyzing datasets and finding the optimal machine learning models can still be quite complex.

This paper is based on the "Predict Future Sales" Kaggle competition. We analyze the historical time series sales of 1C company, which is one of largest Russian software and games selling companies. Our project goals are:

- Grasp a full understanding of historical datasets from January 2013 to October 2015 and figure out the inner relationship of data attributes.
- The main goal is applying a set of machine learning methods to predict total sales for each product in the next month, November 2015
- Analyze the results and try several approaches to achieve optimal performance of models. Obtain insights of this process and purpose novel theories that can be implemented in the future.

### 1.1 Related Work

Traditional regression methods are generally used in forecasting. Kohli et al. (2021) used linear regression to do the sales prediction and compare it with the results with KNN method. Wu et al. (2018) introduced several regression methods and used linear regression to predict the purchase amount on black Friday. Lu et al. (2017) used a hybrid technique which combines Lasso and gradient regression to predict the future sale prices of houses.

Boosting algorithms are widely used in time series data prediction. Krishna et al. (2018) used AdaBoost and GBDT algorithms to forecast retail sales and GBDT outperformed AdaBoost for both RMSE and $R^2$. Dairu et al. (2021), Gurnani et al. (2017), Liang et al. (2019) used XGBoost and/or LightGBM to predict sales of different stores. XGBoost and LightGBM usually performed better prediction compared to traditional regression methods in this research.

Recently, both RNN and LSTM have proved their successes in time series data processing areas such as speech recognition, language translation and image captioning. Salvin et al. (2017) used RNN and LSTM models to predict the stock price which is time series data. Li et al. (2020) applied the LSTM model to predict future sales in 28 days based on the Walmart sales dataset and also showed that the LSTM model outperforms other methods like SVM and linear regression. Intuitively, the assignment to predict the future sales based on historical sales data can be considered as a time series problem, thus LSTM can be used as one approach to predict the time-series data.

### 1.2 Project Outline

The outline contains 5 main steps:

**Data Collection**: Collect training and test sets and do sanity checks to make sure datasets have good consistency for use.

**Data Initial Analysis**: Exploratory analysis and visualization on the initial datasets helps us gain intuitive understanding of the data, knowing more about general trends and key features.

**Data Preparation**: Firstly, we do data preprocessing, including dealing with outliers, absurd values, duplicate data etc. Secondly, after preprocessing the datasets, in the feature engineering part, we do feature extraction, related information extraction and feature fusion to get a final dataframe for latter prediction

**Initial Prediction**: We select a series of models based on whether their principles and characteristics meet the project goals. And we set the initial parameters then do prediction.

**Results Analysis**: After obtaining the results via RMSE, we continue to analyze the models and tuning models' parameters. Based on this, we adjust the models and evaluate their performance. At last, we came up with our conclusion.

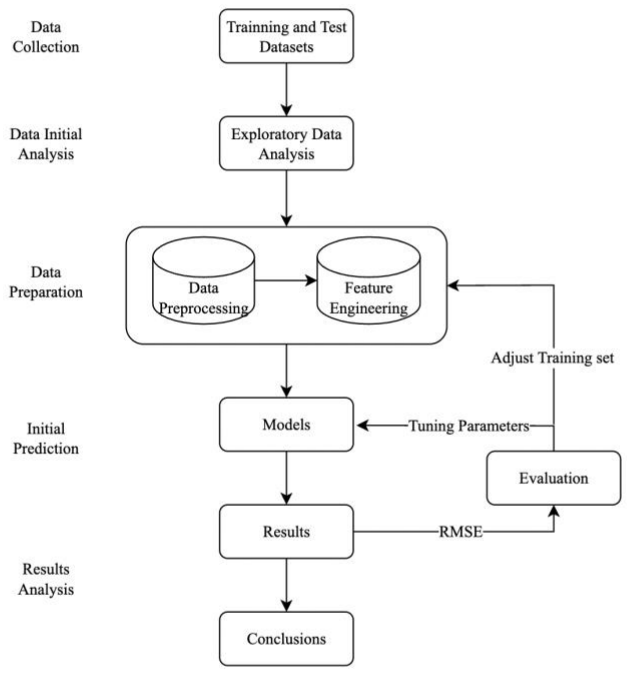

*Figure 1.* Project outline and flow.

**Contributions**: Firstly, this project finds the in-depth relationship within the data attributes, offering insights on how to preprocess data and do reliable feature engineering. Secondly, experimenting with different models finds which is the best. We also concluded methods of tuning parameters. Thirdly, the lag feature we tried is very useful for time-series problems.

## 2. Data Exploration and Pre-processing

The datasets are provided by kaggle and it contains 4 training files including features like ID, shop_id, item_id, item_category_id, item_cnt_day, item_price, date, date_block_num, item_name, shop_name, item_catgory_name. The sales have 84 categories of items in 60 Russian shops, a total of 2935849 time series data.

### 2.1 Exploratory Data Analysis

As Figure 2 shows, the sales have gone through a decline from 2013 to 2015 on a year-on-year basis. Sales usually decrease in the first half of the year, increase in the second half of the year, and finally reach their peak in each December.

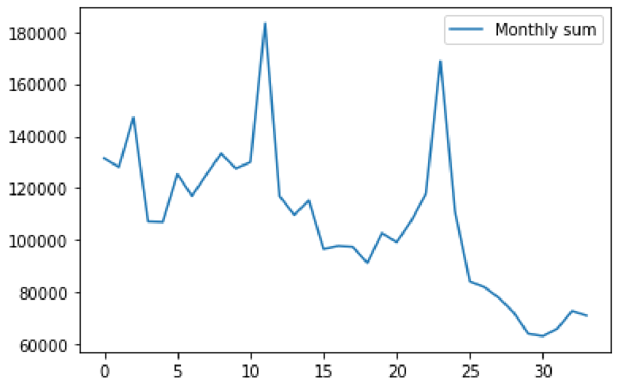

*Figure 2: Total sales in last 34 months*

The name of the shop can be divided into location, brand, and an optional nickname. The sales of the shop are closely related to its location and the number of shops in the city it locates. The name of the category can be separated into two parts, board category and detailed information.

### 2.2 Pre-processing

We have 4 steps in the data preprocessing.

#### 2.2.1 Downcast

Downcasting is to store the data with less information but keep enough information at the same time. We do this in order to save memory when doing feature engineering.

#### 2.2.2 Analysis of Training set

**Sales per store**: By analyzing the sales of the store in the last six months, we find that some stores are new while some are closed. For the new store, we can use the 33rd month's data to predict the 34th month's sales. For closed stores, the sales can be set to zero.

**Removing outliers**: We found out that there are huge gaps in item_price and item_count_day by boxplotting. For item_price, we remove the outliers to let data fall between 0 to 50,000. For item_cnt_day, we remove outliers that are larger than 1000.

**Replacing absurd value**: We also observe that there are negative prices of products and negative numbers of products. We replace these negative prices with the average price of the same product in the same store. We also set the negative number of products sold per day to zero to represent that there is no product sold that day.

#### 2.2.3 Analysis of test set

The test set has 5,100 items and 42 stores. That's exactly 5100 * 42 = 214,200 items-store combinations. It can be divided into three broad categories:

- 363 items did not appear in the training set, 363*42=15,246 items – store combinations had no data, accounting for about 7%.
- 87,550 items - Store combinations are items that only appear, not combinations. Accounted for about 42%
- 111,404 item - store combinations are all present in the training set. Accounts for about 51%.

#### 2.2.4 SHOP DATA CLEANING AND ENCODE

**Changing name to lowercase**: We change all the shop names to lowercase and remove special characters or space, in order to eliminate the factors that will cause errors during the compilation.

**Removing duplicate data**: We observed that some stores with different shop id may have inheritance relationship or even be the same shop. For example, shop 11 only shows the 25th month and shop 10 only missed the data on the 25th month. We deduce that they are the same shop.

**Features information division and encoding**: The shop data contains some other information. We separate them into the new features and encode them with unique ids.

#### 2.2.5 ITEM CATEGORY FEATURE

Just like what we have done with the shop feature, we separate category features into these two new features. Then, we encode the type and subtype features to give them a unique id.

### 2.3 Feature Engineering

#### 2.3.1 INFORMATION MERGE

**Monthly sales statistics**: Firstly, we need to count the monthly sales of every item in every shop of the training set. We get the feature about monthly sales by merging the features shop_id, item_id and date_block_num. Finally, we add the new feature item_cnt_month.

**Related information merge**: Based on previous parts, we already processed the data like shop, item and category. We need to merge the related information of these datasets to prepare for our final dataframe to do prediction.

#### 2.3.2 LAG FEATURES

The future sales prediction is a time-series problem. An important way to do feature engineering for the time series problem is adding lag features which are historical information. Lag features are the classical way that time series forecasting problems are transformed into supervised learning problems. The lag features are basically the target variable but shifted with a period of time, it is used to know the behavior of target values in the past. In this problem, the behavior of our target value months ago can be applied.

- Historical information of monthly sales (each item-store)
- Historical information about the average monthly sales (all item-stores)
- Historical characteristics about the mean value of monthly sales (per item)
- Historical characteristics about monthly sales (per store) mean
- Historical characteristics about the mean of monthly sales (each item category)
- Historical characteristics about monthly sales (item category-store) mean
- The six lag features above are directly related to the item, store, and item category.
- Historical characteristics about the mean of monthly sales (item category_type)
- Historical characteristics about the mean value of monthly sales (item-item category_type)
- Historical characteristics about monthly sales (store_city) mean
- Historical characteristics about monthly sales (item-store_city) mean
- Historical characteristics about the mean value of monthly sales (item-item category_subtype)
- Historical characteristics about item price
- Prices changes over the last six months

The lag features above may be useful， since the trend of sales is also possible related to the features like item_price, item type, subtype, city. All the features are lagged 1,2,3,6,12 months respectively.

#### 2.3.3 OTHER FEATURES

In addition to the lag features above, the number of days per month, the sales of items in the store at the beginning and end are also the important features that influence the sales. Therefore, we also add them to the final dataframe.

#### 2.3.4 NA VALUE PROCESSING

Because 12 months are used as the lagged feature, a large amount of data must be the NA value. The original feature of the first 11 months is deleted, and we need to fill it with 0 for the NA value.

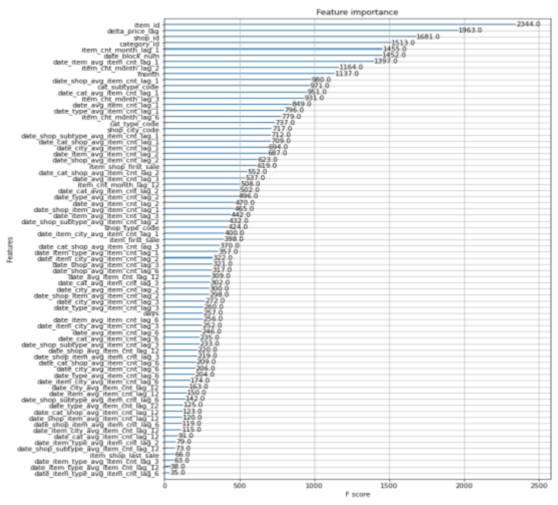

*Figure 3*: Feature importance plot with XGBoost

After getting the papared data, we use XGBoost to plot the features' importance in Figure 3. We also try to drop some features to reduce calculation. As Figure 4 has shown, the performance of the model increases slightly when it comes to more than 20 features. So, we just take 20 features which are comparably important to make the prediction. But the final RMSE increases from 0.91766 to 0.9364, which is obviously not ideal. As a result, we still decide to maintain the whole features.

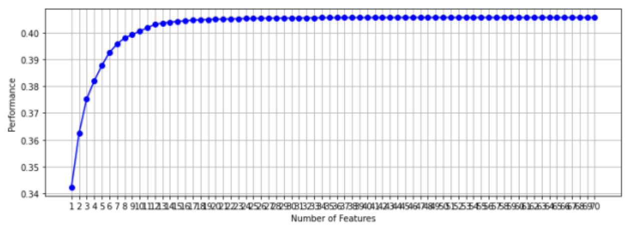

*Figure 4*. Performance with number of features increases

## 3. Modeling

With the fully prepared datasets, in the modeling section, we explain the selection of models and set initial parameters for models.

### 3.1 Model Selection

Please refer to appendix for initial parameter settings.

#### 3.1.1 Linear Regression

This is the most common regression method. It is the simplest model which minimizes the sum of squared error. While it is simple and easy to explain, it has more limitations and is not able to prevent overfitting.

#### 3.1.2 Ridge/Lasso Regression

Lasso and Ridge have the advantages of preventing overfitting compared to simple linear regression. Ridge regression uses L2 norms, and it has the advantage of preventing suffering from multicollinearity problems. In addition, it is easy to compute. Lasso regularization is similar to Ridge regression but uses L1 norms. Lasso has advantages of feature selection because it can make the coefficient down to 0.

#### 3.1.3 Bayesian Ridge Regression

Bayesian Ridge regression is a Bayesian model with ridge regression. The probabilistic model is:

$$p(y|X, \omega, \alpha) = N(y|X\omega, \alpha)$$

where output y is assumed to be Guassian distributed around $X\omega$, and $\alpha$ is a random variable to be estimated from data. $\omega$ is given by a spherical Gaussian.

#### 3.1.4 RandomForest

RandomForest is an ensemble learning approach based on Bagging which can handle classification and regression problems well. It can achieve high accuracy and can effectively run-on large datasets with many features even without feature selection. It not only can handle discrete data, but also continuous data, and there is no need to normalize the data set. It is unlikely to happen overfitting by using Čeh et al. (2018) shows that RF does give good predictions to the data type with the nature like price.

#### 3.1.5 AdaBoost

Adaptive Boosting is a basic boosting algorithm. It improves the results of weaker learners by taking a weighted sum of these results AdaBoost supports both classification and regression. As a basic boosting method, we use it in comparison with XGBoost and LightGBM.

#### 3.1.6 Gradient Boosting

Gradient Boosting is also a typical boosting algorithm. It relies on the intuition that the best possible next model, when combined with the previous models, minimizes the overall prediction errors. The key idea is to set the target outcomes from the previous models to the next model in order to minimize the errors (Kumar, 2020).

#### 3.1.7 XGBoost

XGBoost also uses decision trees as base learners, combining many weak learners to make a strong learner. By adopting several important systems and algorithmic optimizations, this system runs more than ten times faster than existing popular solutions on a single machine and scales to billions of examples in distributed or memory-limited settings (Chen et al., 2014).

#### 3.1.8 LightGBM

LightGBM implements the GBDT algorithm, supports high-efficiency parallel training, and has faster training speed, lower memory consumption, better accuracy, and can quickly process massive amounts of data. lightGBM has been optimized as follows on the traditional GBDT algorithm to avoid the flaws of XGBoost like large space consumption, large time consumption and being unable to optimize the cache. (Weng et al., 2019)

3.1.9 LSTM

Long short-term memory (LSTM) is a special RNN, mainly to solve the problem of gradient disappearance and gradient explosion in the training process of long sequences. In short, compared to ordinary RNNs, LSTM can perform better in longer sequences. Since the future sales prediction is a time-series problem, the LSTM will be very useful for sequential data (Weng et al., 2019).

3.1.10 EXTRATREES

Extra Trees, also known as Extremely Randomized Trees, is an ensemble of decision trees. It has a lot in common with RandomForest. The main different points for Extra Trees are that it splits nodes by choosing cut points fully at random and that it uses the whole learning sample to grow the trees. (Geurts et al., 2006)

### 3.2 Tuning Parameters

The hyperparameter-tuning process is mainly based on the sklearn packages: sklearn.model_selection.GridSearchCV or sklearn.model_selection.RandomizedSearchCV. We specify the range of hyperparameters to be searched and perform cross-validation in pairs. This method will obtain the optimal solution within the range.

### 3.3 Model Evaluation Metric:

Since we have no access to the true target value, the evaluation of prediction results is implemented on Kaggle website by uploading the result file. Results are evaluated by root mean squared error (RMSE). True target values are clipped into [0,20] range. RMSE is defined by the following equation:

$$RMSE(\hat{\theta}) = \sqrt{MSE(\hat{\theta})} = \sqrt{E((\hat{\theta} - \theta)^2)}$$

where $\hat{\theta}$ is the estimated value of the parameter and θ is the true target value.

## 4. Results

### 4.1 Initial Results

In the first place, we get the RMSE results of each model as shown in the *Figure* below:

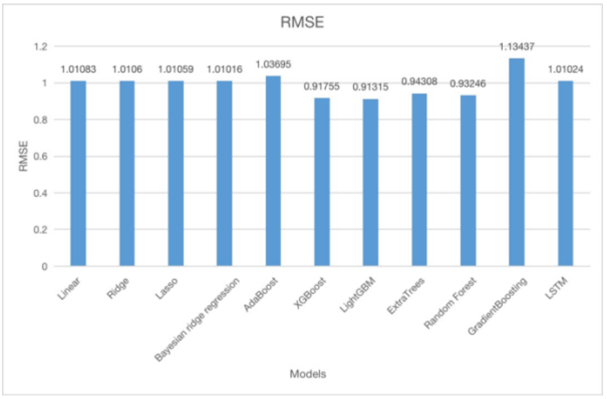

*Figure 5*: RMSE of initial models

**Initial Findings**: From the initial model fitting, we can conclude that overall, ensemble methods like XGBoost, LightGBM, RF achieve the best performance among all the models. Additionally, there are several key points we have found: Firstly, some ensemble methods like AdaBoost and Gradient Boosting surprisingly have poorer results than linear, ridge, lasso, and Bayesian regression methods. One reason is that these two ensembles have not been tuned and just fit with default settings, so their potential performance is highly limited. In addition, with the excessive large training data, Adaboost and Gradient Boosting are possibly overfitted. Although nearly all of the models are overfitted, these two suffer the most.

### 4.2 After-tuning Results

After ranking the models' performances, we choose the best ones to tune. Random Forest is not tuned due to technical limitations. The results after tuning specific models shown in the figure 6 below:

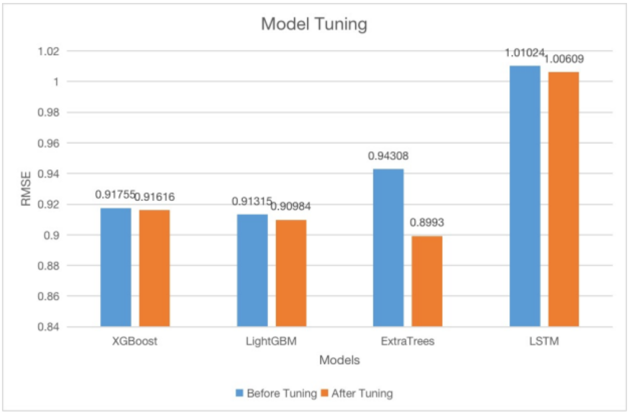

*Figure 6*: Comparison of models before and after tuning

**After-tuning Findings**: After tuning, all selected models attain better results. To our surprise, Extratrees Regressor achieved the best score higher than XGBoost and LightGBM which are considered the best ones on forecasting. The main reason could be that Extratrees Model is well pruned and its variance problems better solved after finding the optimal min_samples_split and min_samples_leaf. Another is that Extratrees may find high pairwise correlations by randomized splits. While LightGBM benefits most from more suitable num_leaves, min_data_in_leaf, XGBoost archives higher scores by adjusting max_depth, gamma and learning rate.

Although GridSearch and RandomizedSearch are useful in improving the final performance of models, they consume a great amount of time and have high requirements for hardware especially dealing with large datasets.

### 4.3 Final Table

*Table 1: Best models and performance*

| Tuning Models | RMSE | Key Parameters |
|---|---|---|
| XGBoost | 0.91616 | max_depth<br>gamma<br>learning rate |
| LightGBM | 0.90984 | num_leaves<br>min_data_in_leaf |
| ExtraTrees | 0.89930 | max_depth<br>min_samples_split<br>min_samples_leaf |

## 5. Conclusions

### 5.1 Summary

Initially, we chose this topic because it has enough datasets, and it is closely related to ensemble methods we want to explore after researching and evaluating. During the project, firstly, in the exploratory data analysis and preprocessing sections, we have nearly done everything that is necessary and there seems to be no more work to be done. It is possible there is something we missed since Russian names can be tricky. Furthermore, we work on the feature engineering and implement lag features in terms of time, shop's name, shop's location, and other related information. This helps us to learn the indirect relationship between sales and data features.

### 5.2 Lessons Learned

- Always spend enough time on data preparation, including data preprocessing and feature engineering. This guarantees the quality of data you will use in the modeling section. Our dataset seems a bit redundant, so correspondingly the amount of calculation is greatly increased.
- Tuning hyperparameters with gridsearch and randomized search should be the final improvement because this process takes much longer than we thought, and it has very high demand on machines. Ours are not.
- Also, tuning is highly dependent on local ranges, so it often achieves the local optimals. This is needed to solve.
- Predicting time-series data, some derived historical information must be considered. But not all. Selecting a set of approachable and useful attributes helps most.

### 5.3 Future Work

- Further research on the feature extraction and selection process. For example, drop those attributes with less importance.
- Tuning hyperparameters on a larger scale and more diverse types of parameters. Try to achieve the global optimal.
- Using GPU to adjust and run complex deep learning approaches like LSTM.
- Novel approaches include adding an expert system on deciding which features to use for specific months. There could be a new way to tune parameters like parallel tuning or use Bayesian Optimization.

**Code Source:** https://github.com/DDZRNL/Future-Sales-Prediction.git

**Related codes, complete parameter settings and results (appendix) please refer to GitHub.**

## References

Bayraktar, E., Lenny Koh, S. C., Gunasekaran, A., Sari, K., & Tatoglu, E. (2008). The role of forecasting on bullwhip effect for E-SCM applications. *International Journal of Production Economics*, *113*(1), 193–204. https://doi.org/10.1016/j.ijpe.2007.03.024

Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).

Dairu, X., & Shilong, Z. (2021). Machine learning model for sales forecasting by using XGBoost. *2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE)*. https://doi.org/10.1109/iccece51280.2021.9342304

Deepdivelm. (2021, January 21). *Feature Engineering/LIGHTGBM/exploring performance*. Kaggle. Retrieved December 10, 2021, from https://www.kaggle.com/deepdivelm/feature-engineering-lightgbm-exploring-performance/notebook#4.-Analysing-Model-Output.

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern classification*. Willey-Interscience.

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, *63*(1), 3–42. https://doi.org/10.1007/s10994-006-6226-1

Gordotron85. (2020, July 26). *Future sales XGBoost - top 3%*. Kaggle. Retrieved December 10, 2021, from https://www.kaggle.com/gordotron85/future-sales-xgboost-top-3#Modelling.

Gurnani, M., Korke, Y., Shah, P., Udmale, S., Sambhe, V., & Bhirud, S. (2017). Forecasting of sales by using fusion of Machine Learning Techniques. *2017 International Conference on Data Management, Analytics and Innovation (ICDMAI)*, 93–101. https://doi.org/10.1109/icdmai.2017.8073492

Krishna, A., V, A., Aich, A., & Hegde, C. (2018). Sales-forecasting of retail stores using Machine Learning Techniques. *2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS)*, 160–166. https://doi.org/10.1109/csitss.2018.8768765

Kumar, A. (2020, June 30). *Introduction to the gradient boosting algorithm*. Medium. Retrieved December 11, 2021, from https://medium.com/analytics-vidhya/introduction-to-the-gradient-boosting-algorithm-c25c653f826b.

Lancaster, G., & Reynolds, P. (2002). *Marketing made simple*. Made Simple.

Li, X., Du, J., Wang, Y., & Cao, Y. (2020). Automatic sales forecasting system based on LSTM network. *2020 International Conference on Computer Science and Management Technology (ICCSMT)*, 393–396. https://doi.org/10.1109/iccsmt51754.2020.00088

Liang, Y., Wu, J., Wang, W., Cao, Y., Zhong, B., Chen, Z., & Li, Z. (2019). Product marketing prediction based on xgboost and LIGHTGBM algorithm. *Proceedings of the 2nd International Conference on Artificial Intelligence and Pattern Recognition - AIPR '19*, 150–153. https://doi.org/10.1145/3357254.3357290

Lu, S., Li, Z., Qin, Z., Yang, X., & Goh, R. S. (2017). A hybrid regression technique for house prices prediction. *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, 319–323. https://doi.org/10.1109/ieem.2017.8289904

Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2017). Stock price prediction using LSTM, RNN and CNN-sliding window model. *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 1643–1647. https://doi.org/10.1109/icacci.2017.8126078

Weng, T., Liu, W., & Xiao, J. (2019). Supply Chain Sales Forecasting based on LIGHTGBM and LSTM combination model. *Industrial Management & Data Systems*, *120*(2), 265–279. https://doi.org/10.1108/imds-03-2019-0170

Wu, C.-S. M., Patil, P., & Gunaseelan, S. (2018). Comparison of different machine learning algorithms for multiple regression on Black Friday sales data. *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, 16–20. https://doi.org/10.1109/icsess.2018.8663760

Zhao, X., Xie, J., & Leung, J. (2002). The impact of forecasting model selection on the value of information sharing in a supply chain. *European Journal of Operational Research*, *142*(2), 321–344. https://doi.org/10.1016/s0377-2217(01)00300-9

Čeh, M., Kilibarda, M., Lisec, A., & Bajat, B. (2018). Estimating the performance of Random Forest versus multiple regression for predicting prices of the apartments. *ISPRS International Journal of Geo-Information*, *7*(5), 168. https://doi.org/10.3390/ijgi7050168