Store Sales Prediction Using Gradient Boosting Models

Jaeyoung Choi[1]

Library and Information Science,
Sungkyunkwan University, Seoul, South Korea, 03063
cjengy@gskku.edu

Abstract. Through the rapid developments in machine learning, there have been diverse utilization approaches not only in industrial fields but also in daily life. Implementations of machine learning on financial data, also have been of interest. Herein, we employ machine learning algorithms to store sales data and present future applications for fintech enterprises. We utilize diverse missing data processing methods to handle missing data and apply gradient boosting machine learning algorithms; XGBoost, LightGBM, CatBoost to predict the future revenue of individual stores. As a result, we found that using median imputation onto missing data with the appliance of the xgboost algorithm has the best accuracy. By employing the proposed method, fintech enterprises and customers can attain benefits. Stores can benefit by receiving financial assistance beforehand from fintech companies, while these corporations can benefit by offering financial support to these stores with low risk.

Keywords: Machine learning, Sales Prediction, XGBoost, LightGBM, CatBoost.

1. Introduction

Nowadays, the rapid developments of machine learning have led to various applications. The utilization of these technologies not only includes information retrieval, computer vision, human-computer interactions, robotics, and natural language processing but also approaches in fintech, where finance and technology are integrated.

There have been diverse appliances where machine learning was utilized in financial areas. Recent studies can be divided into two fields: Classification and Regression. One project in classification tasks, is the study [1], which predicts the delinquency of card users with the data of credit card usage and card deposit history. By the implementation of svm(support vector machine) which separates classes through hyperplanes, a binary classification in whether the users will be late payers in the future is attained. With the comparison of additional machine learning models, this study concludes that ensemble-based models have a better accuracy rate than the singular models. Another classification task[2], used the UCI Credit Card Data from Kaggle[3], to compare the performance of machine learning and deep learning on card data. Machine learning methods include models such as logistic regression, svm, and random forest. The deep learning approaches contain models such as the deep neural network(DNN), and convolutional neural network(CNN). As a result, the random forest algorithm recorded the best accuracy among all of the models. Another classification research [4] held by Suwon city, Korea, was accomplished for the detection of tax delinquency of the residents. To investigate this, random forest, logistic regression, recursive partitioning, and neural network models were

utilized. In conclusion, the logistic regression model was chosen, which holds the advantage of being interpretive. For regression tasks, the study[5] using the Rossmann Store Sales data[6] from Kaggle, utilized machine learning on time series data. Lasso regression, random forest, extra tree, arima, and a stacking method of the combination of xgboost, extra tree, and neural network models were suggested to predict the store revenue. The stacking method resulted to be the most accurate model, inferring the excellence of tree-based algorithms on time series data. Another regression task [7] used CNN and recurrent neural network(RNN) to predict the stock price for the top 20 issues in KOSPI. RNN resulted to have an accuracy of 52 percent, which was concluded to be the most accurate model. The study [8] uses an attention method with LSTM to predict the KOSPI stock prices of Korean mobile carrier companies: KT, LGU+, and SKT. The utilization of different types of attention methods accompanied by element-wise dot product context vectors, resulting in the best performance.

As the previous studies display, financial data is mostly analyzed with elementary machine learning methods such as logistic regression, random forest, and svm or with complex neural network models such as DNN, CNN, or RNN. In the contrast, tree-based ensemble methods are rarely utilized on this data. Gradient boosting algorithms such as: xgboost, lightgbm, catboost are usually used in areas of environment[9] , housing[10] and science fields[11][12]. Table 1 enumerates the previous studies mentioned above.

Table 1. Tasks of Past Studies

Task	Goal	Model
Classification	Credit Card Delinquency Prediction[1]	svm, ensemble of svm
Classification	Default Payments of Credit Card Clients in Taiwan[2]	logistic regression, svm, random forest, DNN, CNN
Classification	Suwon Citizen Delinquency prediction[4]	logistic regression, recursive partitioning,random forest, DNN
Regression	Rossmann Store Sales Prediction[5]	[stacking] lasso regression, arima, extra tree, xgboost, DNN
Regression	KOSPI Stock Price Prediction[7]	DNN, CNN, RNN
Regression	KT, LGU+, SKT Stock Price Prediction [8]	attention-bi-LSTM
Classification	PM2.5 in Seoul Prediction[9]	ensemble of xgboost
Regression	Seoul Residential Apartment Price Prediction [10]	[stacking] xgboost, lightgbm
Regression	Load Forecasting [11]	xgboost
Regression	Safety Driver Prediction[12]	xgboost, lightgbm

In contrast to these previous research approaches on finance data, this study will focus on implementing tree-based machine learning algorithms to predict the future store revenue[13]. We will first deal with mechanisms of handling missing data in time series data. Furthermore, we will utilize tree-based gradient boosting ensemble models such as xgboost, lightgbm, catboost, and compare the performances. To sum up, this paper will be comprised of the following contents:

- Preliminaries on handling missing data and machine learning algorithms
- Methodologies implemented to the data
- Comparing the accuracy of the combinations of methods in dealing with missing data and machine learning algorithms
- Discussions of the findings
- Conclusion

2. Preliminaries

2.1. Handling Missing Data

Missing Data Mechanisms To deal with the missing data, we first need to understand the reasons why these data instances are missing. We distinguish the missing data pattern to understand the relationship between missingness and the variables in the data matrix. According to the assumptions that define these missing data mechanisms, there are three patterns: MCAR(Missing Completely At Random), MAR(Missing At Random), and MNAR(Missing Not At Random). MCAR is when the missingness does not depend on the values of the data Y. The missing data has no relation with other values, making the nonappearance independent. This assumption does not mean that the pattern itself is random, but rather that the missingness does not depend on other data values. When the missing data is assumed to be MCAR and comprises a small proportion, imputation or deletion methods will have less impact on the variation in the dataset. An assumption that is less restrictive than MCAR is MAR, which is when the missingness depends only on the components of Y that are observed, and not on the components that are missing. This is when we can make inferences of the missing data from the observed data. Most of the statistical imputation methods assert this mechanism. When the distribution of missingness depends on the missing values in the data, it is called MNAR. Each mechanism can be defined by the following equation[14][15].

$$MCAR : P(\gamma|Y_{observed},\ Y_{missing}) = P(\gamma). \tag{1}$$

$$MAR : P(\gamma|Y_{observed},\ Y_{missing}) = P(\gamma|Y_{observed}). \tag{2}$$

$$MNAR : P(\gamma|Y_{observed},\ Y_{missing}) = P(\gamma|Y_{observed},\ Y_{missing}). \tag{3}$$

Handling Missing Data The basic method used to handle missing data is deletion. Deletion is comprised of list-wise deletion and pair-wise deletion. List-wise deletion is removing the entire row of the missing data. When a large proportion is missing, this method will increasingly reduce the dataset, so this method is mainly used when the proportion of missingness is below 10 percent of the dataset. So, when the missing data follow the mechanism of MCAR, list-wise deletion will not introduce any biases into the parameter estimates. When the missing data are under MCAR, the sub-sample of cases with complete data is equivalent to a simple random sample from the original target sample. However, when the missing data follow the mechanism of MAR, the list-wise deletion may introduce biases to the dataset. Another deletion method, Pair-wise deletion, is known as available case analysis. It is when we delete the missing data individually. Compared with list-wise deletion, there are fewer data eliminated, but due to the individual deletion, the sample size for each variable differs. When the data are MCAR, pairwise deletion produces consistent estimates, but in the MAR situation, the pairwise deletion may yield biased estimates. [16]

Another method used when handling missing data is data imputation. This approach is when some guess or estimate is substituted for each missing value. Data imputation consists of two approaches: single imputation and multiple imputation. Single imputation

is to generate one value through a computational process, while multiple imputation is to generate various values for the missing data instance and combine the diverse values into one value. [16] [15] Single imputation includes various methods such as mean imputation where the missing values are replaced with the mean value and median imputation where the missing values are supplied with the median value. These replacements easily make the dataset as a whole, but standard errors computed from the data used to fill the missing data are systematically underestimated and do not account for the imputation uncertainty.[14]

One other method utilized when handling missing data is interpolation, which is making predictions of the missing data. Interpolation includes linear interpolation, and spline interpolation, which utilize the Newton interpolation method. When we have *n+1* values, we use a *n* degree estimate function for the interpolation. To be specific, in the case of linear interpolation, we use two values to estimate the one-degree function. The following equation shows the equation for linear imputation [17].

$$\begin{aligned} a = (x_1, y_1),\ b = (x_2, y_2)\ \ (x_1 < x_2) \\ F_1(x) = b_0 + b_1(x - x_1) \\ b_0 = y_1,\ b1 = \frac{y_2 - y_1}{x_2 - x_1} \end{aligned} \tag{4}$$

Spline interpolation utilizes a polynomial function to smoothly link the data points with a nonlinear action. These two-degree functions use quadratic interpolation and can be explained through the following equation.

$$\begin{aligned} a = (x_1, y_1),\ b = (x_2, y_2)\ c = (x_3, y_3)\ \ (x_1 < x_2 < x_3) \\ F_1(x) = b_0 + b_1(x - x_1) + b_2(x - x_1)(x_1 - x_2) \\ b_0 = y_1,\ b_1 = \frac{y_2 - y_1}{x_2 - x_1} \\ \therefore b_2 = \frac{\frac{y_3 - y_2}{x_3 - x_2} - \frac{y_2 - y_1}{x_2 - x_1}}{x_3 - x_1} \end{aligned} \tag{5}$$

2.2. Machine Learning Algorithm

Ensemble Method Ordinary machine learning algorithms work by searching the hypotheses space to find the one function, *h*, which represents the best approximation for the data. However, ensemble algorithms have a different approach. Instead of finding the one best hypothesis, *h*, this algorithm constructs a set of hypotheses, ensembles, where each hypothesis has a 'vote' to predict the data. Ensemble learning algorithms can overcome the problems, single hypothesis algorithms suffer, by reducing both the bias and variance of the data. There are two main approaches which subdivide the ensemble learning algorithm. The first method is to construct each hypothesis independently, which leads to a set of hypotheses that are diverse and accurate. This approach forces a learning algorithm to construct multiple hypotheses with different inputs, outputs, or features in each run. To be specific, random forest, which applies the bagging(bootstrap aggregating) algorithm, is an example when the input values are uniformly re-sampled. This produces

a diverse ensemble of independently constructed hypotheses. Another method is to construct the hypotheses in a coupled fashion. This approach differs from the first approach, as each hypothesis is not constructed independently, but as an *additive model*. In detail, this method, boosting, predicts the unseen data by taking a weighted sum of a set of component models. Adaboost algorithms and gradient boost algorithms are examples of this approach.[18]

Gradient Boosting Method The gradient boosting algorithm is a method that utilizes boosting approaches, which is when the component hypothesis influences the choice of other hypotheses[18]. Gradient boosting models start with an initial value $F_0(x)$ and develop additive expansions of the base learners. This is based on the negative gradient(y_i) from the loss, which is calculated by the difference between the predictions of the previous base learner and the actual values. There are diverse methods when calculating the loss. In detail, when the loss function is $L(y_i, f(x_i) = \frac{1}{2}(y_i - f(x_i))^2$, the negative gradient is calculated by the following equations. Through the computation of the gradient, $-\frac{\delta(y_i, f(x_i))}{\delta f(x_i)}$ is calculated, and the negative gradient is attained. In this example, the negative gradient is the the residual$((y_i - f(x_i)))$, so this model is called the residual fitting model. Similarly, as this process, the negative gradient is utilized to fit the next base learner. To be specific, the next base learner takes a new step by the gradient descent process, for the purpose to minimize the loss. The update process is illustrated in Fig. 1. This method is the theoretical foundation of the following algorithms: xgboost, lightgbm, and catboost.[19]

Algorithm 1: Gradient_Boost

$F_0(\mathbf{x}) = \arg\min_\rho \sum_{i=1}^{N} L(y_i, \rho)$

For $m = 1$ to M do:

$\tilde{y}_i = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}, \; i = 1, N$

$\mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^{N} [\tilde{y}_i - \beta h(\mathbf{x}_i; \mathbf{a})]^2$

$\rho_m = \arg\min_\rho \sum_{i=1}^{N} L(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i; \mathbf{a}_m))$

$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}; \mathbf{a}_m)$

endFor

end Algorithm

Fig. 1. Sudo Code for the Gradient Boost Algorithm

XGBoost Xgboost utilizes the gradient boosting method with a tree approach. A scalable tree boosting system, xgboost, is a challenge winning machine learning algorithm and is available as an open-source package[20]. This model introduces improvements in the gradient boosting model, by making the algorithm more efficient. One noticeable difference in the xgboost algorithm is in the regularized objective. For the given data, xgboost minimizes the following regularized loss function.

$$L(\phi) = \sum l(\widehat{y_i}, y_i) + \sum \Omega(f_k)$$
$$where\ \Omega(f_k) = \gamma T + \frac{1}{2}\lambda \|x\|^2 \tag{6}$$
$$(T\ is\ the\ number\ of\ leaves\ of\ the\ tree)$$

One of the key problems to solve in tree learning is finding the best split. Xgboost utilizes two split finding algorithms, the exact greedy algorithm, and the approximate algorithm. The exact greedy algorithm enumerates over all the possible splits on all of the features, while, the approximate algorithm selects split candidates according to the percentiles of the distribution of the features. [21]

LightGBM Although with the improvements made in the xgboost model, the model's efficiency and scalability were still unsatisfactory when it was utilized in high dimensional data. These problems happened due to the scanning process, which is finding possible split points by searching the entire data. To tackle this problem, the Microsoft Research team proposed a new method called lightgbm, which accompanies the implementations of two splitting techniques: gradient-based one side sampling and exclusive feature bundling. As the following Fig. 2 displays, lightgbm uses partial amounts of the training instance.

Algorithm 2: Gradient-based One-Side Sampling

Input: I: training data, d: iterations
Input: a: sampling ratio of large gradient data
Input: b: sampling ratio of small gradient data
Input: $loss$: loss function, L: weak learner
models ← { }, fact ← $\frac{1-a}{b}$
topN ← a × len(I) , randN ← b × len(I)
for $i = 1$ ***to*** d **do**
  preds ← models.predict(I)
  g ← $loss$(I, preds), w ← {1,1,...}
  sorted ← GetSortedIndices(abs(g))
  topSet ← sorted[1:topN]
  randSet ← RandomPick(sorted[topN:len(I)], randN)
  usedSet ← topSet + randSet
  w[randSet] × = fact ▷ Assign weight $fact$ to the small gradient data.
  newModel ← L(I[usedSet], − g[usedSet], w[usedSet])
  models.append(newModel)

Fig. 2. Sudo Code for the Gradient-based One-side Sampling Algorithm

To be specific, GOSS keeps all of the instances with large gradients and performs random sampling on the data instances with small gradients. To compensate for the influence of the data distribution, a constant multiplier is introduced for the small gradient data instances. Furthermore, GOSS sorts the gradients by their absolute value and randomly samples the instances. Through this process, GOSS does not use the entire data instances but utilizes a sample of the data when splitting the tree. Exclusive feature bundling is a

method to reduce the number of features used to split the tree. To solve the problems of sparse high-dimensional data, the lightgbm algorithm bundles exclusive features into a single feature. EFB includes two algorithms, which are enumerated in Fig. 3. The first algorithm determines which features should be bundled together and the second algorithm regulates how to construct the bundle. With the implementation of these two techniques, the efficiency in finding the split point, without enumerating the complete dataset, is ensured. [22]

Algorithm 3: Greedy Bundling

Input: F: features, K: max conflict count
Construct graph G
searchOrder ← G.sortByDegree()
bundles ← {}, bundlesConflict ← {}
for i ***in*** *searchOrder* **do**
  needNew ← True
  for $j = 1$ ***to*** *len(bundles)* **do**
    cnt ← ConflictCnt(bundles[j],F[i])
    if *cnt* + *bundlesConflict[i]* $\leq K$ **then**
      bundles[j].add(F[i]), needNew ← False
      break
  if *needNew* **then**
    Add $F[i]$ as a new bundle to *bundles*
Output: *bundles*

Algorithm 4: Merge Exclusive Features

Input: $numData$: number of data
Input: F: One bundle of exclusive features
binRanges ← {0}, totalBin ← 0
for f ***in*** F **do**
  totalBin += f.numBin
  binRanges.append(totalBin)
newBin ← new Bin(numData)
for $i = 1$ ***to*** $numData$ **do**
  newBin[i] ← 0
  for $j = 1$ ***to*** *len(F)* **do**
    if *F[j].bin[i]* $\neq 0$ **then**
      newBin[i] ← F[j].bin[i] + binRanges[j]
Output: $newBin$, $binRanges$

Fig. 3. Sudo Code for the Exckysuve Feature Bundling Algorithm

CatBoost Catboost(categorical boosting) is a gradient boosting tool toolkit, developed at Yandex in 2017. It solves the issues of gradient boosting models such as target leakage and inefficiency in processing categorical values. Whilst the previous gradient boosting algorithm utilizes all of the existing training instances when calculating the proper split points, target leakage and a prediction shift occurs. To solve these problems, lightgbm proposes ordering boosting. At each step of boosting, the ordered boosting algorithm dependently samples the dataset.

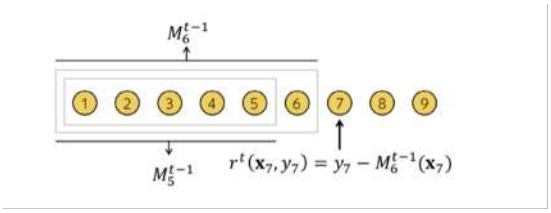

Fig. 4. Example of Ordered Boosting Principle

To be specific, Fig. 4 shows an example of ordered boosting, where a random permutation σ of the training examples are utilized with n different models. The model M_i

is learned using only the first *i* instances in the permutation. At each step, to obtain the residual for the *j*th sample, the model M_{j-1} is utilized to achieve the residual for the calculation of the negative gradient. Fig. 5 displays the algorithm of ordered boosting.

For categorical variables, ordinal gradient boosting algorithms apply one-hot-encoding approaches or target statistics to alter these categorical values into numerical values. In the case of one-hot-encoding, it adds a new binary feature for each category, which leads the categorical values to become high dimensional and sparse features, which generates inefficiency. An example of target statistics, greedy target statistics, estimates the average value over the training examples in the same category. This estimate is noisy for low-frequency categories and derives target leakage. To prevent this, catboost introduces ordered TS. Ordered TS relies on the ordering principle, which only utilizes the observed history. An artificial 'time' is introduced and uses the available history to compute its TS. This encourages the model to not use the previous observations according to the artificial time, which solves the problem of target leakage[23].

Algorithm 1: Ordered boosting

input : $\{(\mathbf{x}_k, y_k)\}_{k=1}^{n}$, I;

$\sigma \leftarrow$ random permutation of $[1, n]$;
$M_i \leftarrow 0$ for $i = 1..n$;
for $t \leftarrow 1$ **to** I **do**
  for $i \leftarrow 1$ **to** n **do**
    $r_i \leftarrow y_i - M_{\sigma(i)-1}(\mathbf{x}_i)$;
  for $i \leftarrow 1$ **to** n **do**
    $\Delta M \leftarrow$
     $LearnModel((\mathbf{x}_j, r_j) :$
     $\sigma(j) \leq i)$;
    $M_i \leftarrow M_i + \Delta M$;
return M_n

Fig. 5. Sudo Code for the Ordered Boosting Algorithm

3. Research Methodology

Research Data This study utilizes the data from the Korean data competition platform DACON [24], 'Store Credit Card Sales Prediction Competition'[13]. This data was contributed by FUNDA[25], and the variables are listed in Table 2.

Preprocessing Methods 31% of the variable 'region' was missing data, and 60% of the variable 'type_of_business was missing data, so we eliminated these variables. The

Table 2. dataset Variables

Variable Name	Explanation
store_id	Store ID
card_id	Credit Card ID
card_company	De-identified Card Company Name
transacted_date	Date of Transaction
transacted_time	Time of Transaction
installment_term	Installment Term of Transaction
region	Store Region
type_of_business	Type of Business
amount	Transaction Amount

proportion of missing values for each variable can be seen in Fig. 6. Additionally, we deleted the variables of the following: 'card_id', 'card_company', and 'transacted_time', due to the irrelevance in the predictions regarding store sales.

The dataset included zeros and negative numbers in the revenue(amount variable). To deal with these irregular patterns, we first deleted the rows with zeros and for the negative revenue, we assumed it as refunds and deleted the negative revenue with an identically matching positive revenue value.

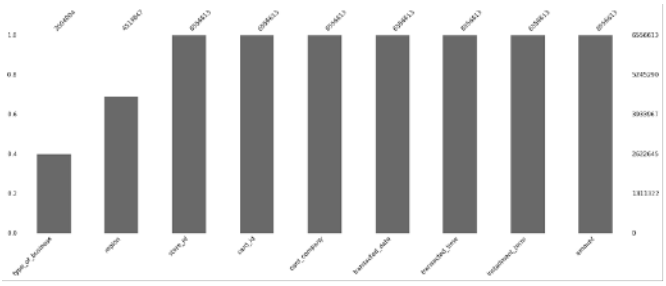

Fig. 6. Missing Data Proportion of the Dataset

As the objective of the analysis was to predict the future three-month revenue for individual stores, we re-sampled the dataset monthly. To be specific, we merged the daily amounts into monthly sales for each store.

Also, to emphasize the monthly revenue, we created new variables displaying past sales. We generated the past three month sales, past four-month sales ... past twelve-month sales as new variables. Because this process utilized the past revenue, it produced various amounts of missing data. This is where we applied methods for handling missing data, such as mean imputation, median imputation, linear interpolation, and spline interpolation.

The following Fig 7, proposes the date difference for each store. Regarding the difference in date, we did not fix the date of the future three-month prediction. Instead, we considered each store individually, selecting the last three months as the test term. To be specific, store_id 1, starts from June 2017 and ends in February 2019. In this case, the

training term was June 2017 to November 2018 and the test term was December 2018 to February 2019. Another example from Fig 7 is store_id 772, which is expressed as the square-line and includes data until October 2018. For this store, the training term was June 2016 to July 2018 and the text term was August 2018 to October 2018. As shown through other examples in Fig 7, the stores had variant end dates, making it impossible to consider the identical duration for prediction.

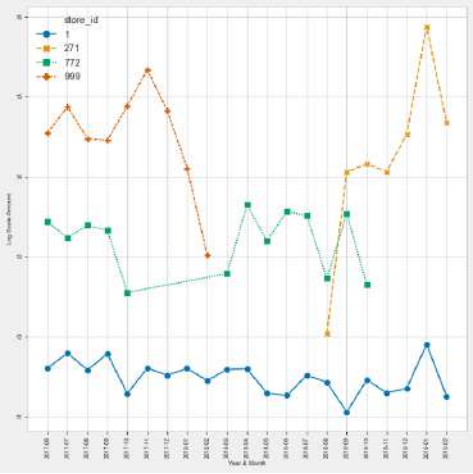

Fig. 7. Difference Between Time Series Plots for Store ID 1, 271, 772 and 999

Final Variables After the preprocessing process, we concluded the following as the final variables which are shown in Table 3.

Table 3. Details in Final Variables

Variable name	Explanation
store_id	Store ID
month	Month of Transaction Date
year	Year of Transaction Date
three_month	Total Transaction Amount three months before the transaction date
four_month	Total Transaction Amount four months before the transaction date
five_month	Total Transaction Amount five months before the transaction date
six_month	Total Transaction Amount six months before the transaction date
seven_month	Total Transaction Amount seven months before the transaction date
eight_month	Total Transaction Amount eight months before the transaction date
nine_month	Total Transaction Amount nine months before the transaction date
ten_month	Total Transaction Amount ten months before the transaction date
eleven_month	Total Transaction Amount eleven months before the transaction date
twelve_month	Total Transaction Amount twelve months before the transaction date
amount	Total Transaction Amount on a monthly basis

4. Results

The results section is organized in the following sections. First, an explanation of the datasets produced through the missing data management and an explanation of the evaluation metric. Second, detailed information about the models which were utilized. Third, the results of the missing data management mechanisms and machine learning algorithms. Finally, visualizations of the model's predictions.

Datasets and Evaluation Metric The datasets we utilized for the training process was a total of five, which included the dataset with the missing data, and the datasets where the missing data were filled through the mean imputation, the median imputation, the linear interpolation, and the spline interpolation. The mean imputation dataset handled the missing data by filling it with the mean value of revenues for each store. The median imputation dataset was composed of a similar approach. Diversely, the linear interpolation dataset was formulated with a linear prediction for the missing data in each store. The spline interpolation was produced in an akin approach.

The evaluation metric we used to measure the magnitude of errors was the mean absolute error(MAE). This metric measures the average magnitude of the errors in a set of predictions. In detail, the metric calculates the average of the absolute difference between the actual values and the predictions. MAE is calculated by the following equation.

$$MAE = \frac{\sum |y_t - \hat{y}_t|}{n}. \tag{7}$$

Model Utilization With the five datasets in mind, we utilized xgboost, lightgbm, catboost to make predictions for future revenue. To prevent over-fitting issues, we used a stratified five-fold cross-validation method to maintain every store in the training process. We also optimized the parameters for every model by the mixture of random grid search and Bayesian optimization.

4.3 Model Results Table 4 shows the MAE score results for each dataset and model which were achieved by various amounts of optimization.

Table 4. MAE Test Score Results

	Xgboost	Lightgbm	Catboost
NA	321279.78	**320953.13**	360355.44
Mean	323330.41	326836.75	**353011.75**
Median	**320487.36**	325040.74	353409.89
Linear	323462.43	323223.69	359235.33
Spline	327696.05	323269.50	354742.81

The xgboost model did not have internal methods that can deal with categorical values, so we composed dummy variables for the categorical variables by one-hot-encoding. As we can speculate from Table 4, for xgboost, the median imputation dataset had the best accuracy, followed by the NA dataset. For the lightgbm algorithm, the combination

of the NA dataset had the acute accuracy, which was followed by the spline interpolation dataset. Lastly, for the catboost model, the mean imputation dataset was the finest match. As Table 4 displays, we concluded that the combination of xgboost and the median imputation dataset had the lowest MAE score, which disclosed the best combination. This is trailed by the association of the NA dataset and the lightgbm algorithm. The selected combination for the catboost model was the mean imputation dataset which had the lowest accuracy among all three models. By the following Fig 8, we can compare the error rate for each model.

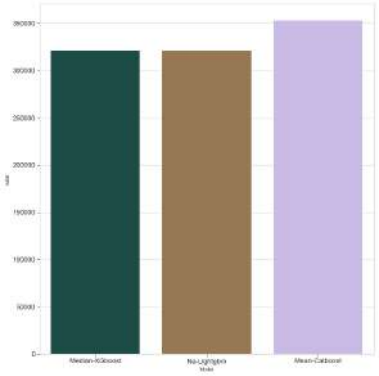

Fig. 8. MAE Barplot for the Final Models

5. Discussion

What is the best dataset-model combination? From Table 4, we can conclude that the xgboost model with the median imputation dataset has the lowest error rate. When comparing the median-xgboost with the NA-lightgbm combination, we can speculate a severely limited difference between the error. This result may indicate that the median-xgboost is the best combination for this dataset. However, from the following Table 5, we can speculate that the xgboost model does have the highest accuracy, but is also associated with the highest time consumption. This time measurement table shows a different aspect of the machine learning algorithms. When comparing with the lightgbm model, which only depletes 9 seconds to make predictions, xgboost consumes severe time, consuming over 3000 seconds.

Table 5. Comparison of MAE with Time Measurements

Dataset	Model	Mae	Time(s)
Median	xgboost	320487.36	3537.50
NA	lightgbm	320953.13	9.70
Mean	catboost	353011.75	35.70

With the small-scale difference in MAE and the large-scale difference in time consumption, we can conclude that the missing data method-machine learning model combination is not implicitly detectable.

Can we compare the accuracy through visualizations? With the final model selections from Table 4, we are capable of visualizing the predictions. The following Fig 9 shows the future three months' store revenue of thirty randomly selected stores. The actual amount is expressed with an x-line, the predictions made by the xgboost model with median imputation is the triangle-line, the predictions made by the lightgbm model with NA dataset is shown by the rectangle-line, and the predictions made by the catboost model with the mean imputation dataset is visualized with the diamond-line. From the graph visualization, we can easily comprehend that the xgboost and lightgbm model gradually follows the actual sales compared to the catboost model.

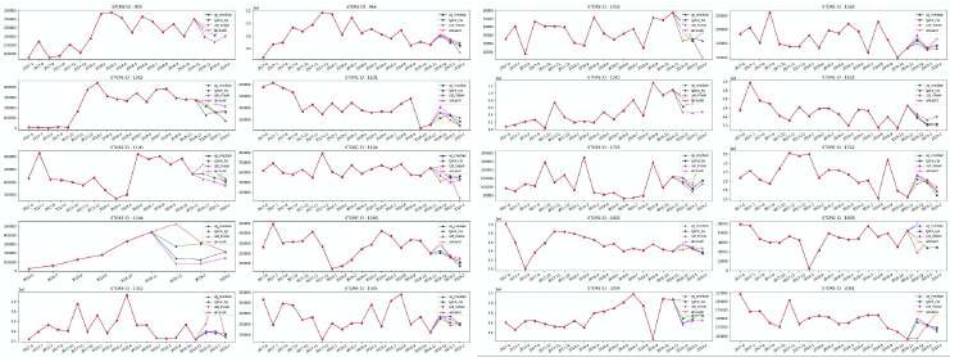

Fig. 9. Comparisons of MAE through Graph Visualization

6. Conclusions

This paper focuses on the methods of handling missing data and the accuracy associated with the utilization of gradient boosting model algorithms. We were able to compare the accuracy for each imputation or interpolation method when combined with machine learning algorithms. For the missing data in the store sales data, we created five datasets: NA(a dataset including the missing data), mean imputation(a dataset where the missing values are filled with mean values), median imputation(a dataset where the missing values are filled with the median values), linear interpolation(a dataset where the missing values are filled by the predictions composed by linear interpolation), and spline interpolation(a dataset where the missing values are filled by the predictions composed by spline interpolation). These datasets independently utilized three gradient boosting models: xgboost, lightgbm, and catboost, making fifteen combinations. We conclude that the combination

of xgboost and mean imputation has the lowest error rate. Although, when regarding the time efficiency, this combination is incomparable with the lightgbm model with the NA dataset, as the latter is 400 times faster.

Nowadays we can easily find applications of data in fintech fields, such as convenient data-driven purchase mechanisms, P2P loans, and financial item recommendation.[26] With these services in mind, we believe that this paper can be a background for fintech enterprises to display new methods in evaluating and aiding future customers. The missing values in real-life customer data can be filled through the introduced missing data handling methods. This can assist fintech enterprises by restoring valuable variables, which will lead to more diverse insights by the application of these variables. Additionally, with the help of gradient boosting ensemble methods, fintech firms can enforce personalized aid services to customers. Through the applications of these methods, not only enterprises but also customers can attain benefits. Customers who have low credibility but have a high probability to compensate for repayment can benefit by receiving financial assistance beforehand from fintech enterprises since the future revenue predictions accomplished by tree-based machine learning speculate the potential of these customers. Fintech companies can offer financial instruments to customers who have high probabilities of repayment. This leads to lower financial risk for the enterprise, making a win-win situation for both the customer and the corporation.

References

1. J. W. Kim and W. C. Jhee. Credit card delinquency prediction model based on data mining approach. In *Proceedings of the Korea Intelligent Information System Society Conference*, pages 232–239, Seoul, Korea, 2011. Korea Intelligent Information Systems Society.
2. J. M. Yoon. Effectiveness analysis of credit card default risk with deep learning neural network. *Journal of Money Finance*, 33(1):151–183, 2019.
3. Kaggle. Uci credit card dataset. [Online]. Available: https://www.kaggle.com/uciml/default-of-credit-ca rd-clients-dataset.
4. E. C. Kim and B. J. Yoo. Case study of analyzing credit information and public big data : Development of prediction model for probability of collecting tax in arrears and enhancement of tax information system. In *Proceeding of 58th the Korea Society of Management information Systems Conference*, pages 6–12, Seoul, Korea, 2018.
5. B. M. Pavlyshenko. Machine-learning models for sales time series forecasting. *Data 2019*, 4(1):15, 2019.
6. Kaggle. Rossmann store sales. [Online]. Available: https://www.kaggle.com/c/rossmann-store-sales.
7. J.H. Lee. *Stock price prediction model using deep learning*. Masters dissertation,Soongsil University, Soongsil University, Seoul, 2016.
8. Y.J.Kim. *Attention Mechanism Based Using Bi-Directional LSTM Stock Price Forecasting Model*. Masters dissertation,Hanbat National University, Hanbat National University,Daejeon, 2019.
9. H. Kim. The prediction of pm2.5 in seoul through xgboost ensemble. *Journal of the Korean Data Analysis Society*, 22(4):1661–1671, 2020.
10. I. H. Kim and K. S. Lee. Tree based ensemble model for developing and evaluating automated valuation models : The case of seoul residential apartment. *Journal of the Korean Data Information Science Society*, 31(2):375–389, 2020.

11. J. Y. Oh Y. G. Lee and G. B. Kim. Interpretation of load forecasting using explainable artificial intelligence techniques. *The Transactions of the Korean Institute of Electrical Engineers*, 69(3):480–485, 2020.
12. S. I. Jang and K. C. Kwak. Comparison of safety driver prediction performance with xgboost and lightgbm. In *Proceeding of Korea Institute of Infomation Technology Conference*, pages 360–362, Seoul, Korea, 2019.
13. Dacon. Card sales prediction contest. [Online]. Available: https://dacon.io/competitions/official/140472/overview/.
14. R. J. A. Little and D. B. Rubin, editors. *Statistical Analysis with Missing Data*, volume 2. Hambrug, NJ: John Wiley Sons Inc., 2014.
15. S. R. Lee. *Comparison of algorithms for the missing data imputation methods*. Masters dissertation,Hankuk University of Foreign Studies, Hankuk University of Foreign Studies,Seoul, 2020.
16. P. D. Allison, editor. *Missing Data*, volume 07-136 of *Series on Quantitative Applications in the Social Sciences*. Thousand Oaks, CA: Sage University Papers, 2001.
17. J. H. Jeong. *Electricity Load Forecasting with Weather Data for Commercial Buildings based on Machine Learning*. Masters dissertation,Cheongju University, Cheongju University,Cheongju, 2018.
18. T. Dietterich. Ensemble learning. In M.A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, volume 2, pages 3–7. Cambridge, MA: The MIT Press., 2002.
19. J. Friedman. Greedy function approximation: A gradient boosting machine. In M.A. Arbib, editor, *The Annals of Statistics*, volume 29, pages 1189–1194. Cambridge, MA: The MIT Press., 2001.
20. Xgboost. Python package for xgboost. [Online]. Available: https://github.com/dmlc/xgboost.
21. T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. [Online]. Available: https://arxiv.org/abs/1603.02754.
22. T. Finley T. Wang W. Chen W. Ma Q. Ye G. Ke, Q. Meng and T. Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 3149—-3157, Long Beach: CA, 2017.
23. A. Vorobev L. Prokhorenkova, G. Gusev and A. Gulin V. A. Dorogush. Catboost: unbiased boosting with categorical feature. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 6639—-6649, Montreal, Canada, 2018.
24. Dacon.korea data competition platform. [Online]. Available: Available:https://dacon.io/.
25. Funda-korea p2p finance finance enterprise. [Online]. Available:https://www.funda.kr/v2/.
26. Banksalad magazine. [Online]. Available: https://banksalad.com/contents/C