Demand Forecasting with Machine Learning Techniques in Supply Chain Management

Runqiu Shi (runqiu)
Boshi Huang (Boshih)

Mentor: Vineet Edupuganti

Abstract

This paper will explain our experimental results for the Stanford CS229 final project. We proposed an ensemble of machine learning methods to achieve demand forecasting, i,e. time-series forecasting. The goal of the project is to find an algorithm that can precisely predict the demand for all kinds of auto parts, hence, to reduce the supply chain management cost. Based on the data characteristics, the SVR, CNN+LSTM, and XGBoost are used in our project, and from the predicted data comparison, we can draw the conclusion that the XGBoost outperforms all other algorithms in our project.

Key Words: Time Series Forecasting, SVR, CNN, LSTM, XGBoost, CS229

1 Motivation

Demand forecasting plays a crucial role in supply chain management because it is the process by which the strategic and operational strategies are devised. Think of it as the underlying hypothesis for strategic business activities and the starting point for most supply chain processes, like raw material planning, purchasing, inbound logistics, cash flow, and manufacturing. However, due to asymmetric information between downstream and upstream, lack of full collaboration in stakeholders, and a well-known 'bullwhip effect", there is mismatch and inefficiency in supply-demand. The quantitative forecast for future demand optimizes the supply chain management system.

The benefits of this optimization boil down into three parts: first, reduction of premium freight due to lowering the risk of capacity constraint; second, the reduction of overhead costs by the optimization of the stock; third, improving operating income and operation efficiency in the corporation by streamlining the operation and supply chain team in the organization.

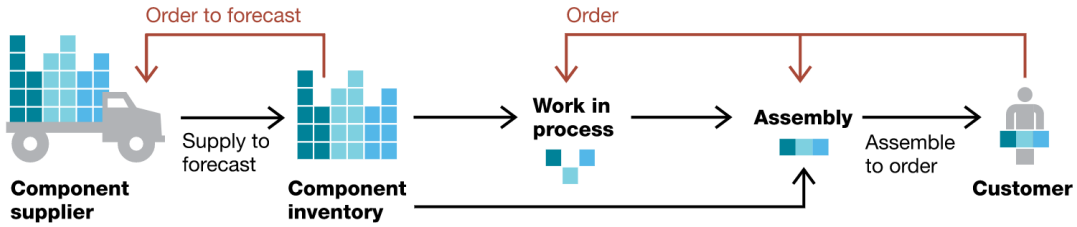

Fig.1 The Flowchart for The Supply Chain

2 Related Work

Several works have focused on comparing the performance of different forecasting techniques in the context of supply chain management. [4] gets the final decision of these models for the proposed system, nine different time series methods, SVR algorithm, and DL model are blended by a new integration strategy which is reminiscent of boosting ensemble strategy. In this way, the final decision of the proposed system is based on the best algorithms of the week by gaining more weight. [1] [2] [3] tests the performance of the deep learning model for time series problems. [5] focuses on LSTM to do the demand forecasting in massive electricity time series and shows the comprehensive analysis for feature selection and datasets analysis. [6] involves traditional statistical models, moving averages, ARMA (Autoregressive Moving Average), and ARIMA (Autoregressive Integrated Moving Average) models. ARMA and ARIMA are the most common methods that are applied to find the best fit of a model to historical values of a time series [7]. Intermittent Demand Forecasting methods try to detect intermittent demand patterns that are characterized with zero or varied demands at different periods. Intermittent demand patterns occur in areas like fashion retail, automotive spare parts, and manufacturing. Modeling intermittent demand is a challenging task because of different variations. One of the influential methods of intermittent demand forecasting is proposed by Croston [8]. Croston's method uses a decomposition approach that uses separate exponentially smoothed estimates of the demand size and the interval between demand incidences. Its superior performance over the single exponential smoothing (SES) method has been demonstrated by Willemain [9].

In recent years, there are several works for optimizing traditional models. Qi et al. present the combination of Ex-Adaboost learning strategy and the DL research based on support vector machine (SVM) and then propose a new Deep Support Vector Machine (DeepSVM) [10]. Sliding window-based support vector regression for predicting is introduced [11]. The more advanced variant of ARIMA, SARIMA, supports univariate time series data with a seasonal component [12].

3 Data Preprocessing

3.1 Preliminary Analysis

Dataset comes from the Tier2 company from the electronic component industry in the industrial field. Besides, given the characteristics of the industry, there are thousands of products in a product line. Shipments of some products are very scarce. Besides, the actual situation is that customers do not place rolling orders but place a huge order for seasonal consumption, which causes shipment data to be zero in the specific month. We decide to predict the future demand based on sub-product lines (a group of products for similar functions and designs in product development).

3.1.1 Booking Quantity VS. Billing Quantity

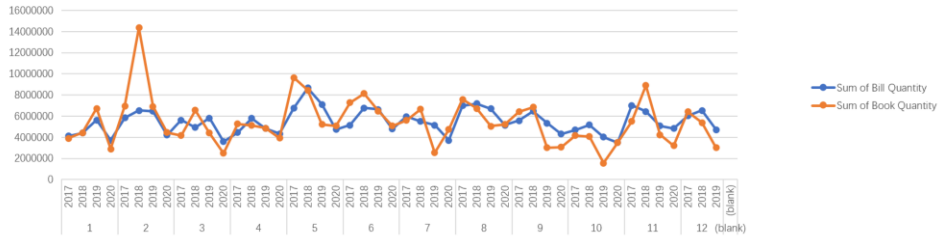

Fig.2 Plot of the Booking and billing Quantity

Booking quantity is the ordered quantity to be shipped this month. The billing quantity is actually shipped this month. And it is the value we should predict in this project. From fig.2, we can see that these two data are highly positively correlated. We could use booking quantity for the short-term reference.

3.1.2 Quarterly Data Analysis

Fig.3 Plot of Quarterly Data

From the fig.3, we can see there is no certainly positive correlation between the year-on-year quarters, and in-depth verification is required.

3.1.3 Monthly Data Analysis

Fig.4 Plot of Monthly Data

The trend between months in four years is similar. For instance, the months of February, May, and August are the peak of shipments.

3.2 Data and Features

The demand forecasting in supply chain management is the time-series problem, which means the

prediction depends on when the sample was taken. All data, including training and test splits, are provided by each month of the year. Our goal is to predict the future bill quantity(shipments) for each month based on the booking quantity (ordered quantity to be shipped) of the current month and past shipments. The provided data includes six sub-product lines. We make the algorithms recognize categorical data by using one-hot encoding. Compared with one-hot encoding, integer encoding has a natural ordered relationship between each other, and machine learning algorithms may harness this relationship and even give more weight to a higher integer value. Besides, some algorithms cannot understand integer encoding. One-hot encoding is the better choice.

Besides, our data is the real one which comes from Tier2 company from the electronic component industry. There are also negative numbers in the data, which represents refunded records. Since the data oscillates very seriously, to avoid more weight to the big value, normalization is introduced in experiments. Since our data doesn't comply with the Gaussian distribution, so we just do max-min scaling other than centering the data by minus the mean and then divide with standard deviation.

In summary, the features include monthly booking quantities and bill quantities for each year, and six parts categories, which are one-hot encoded. And all the data are scaled to [-1, 1]. And the training, validation, and testing split ratio are 7:2:1.

4 Methods

We propose an ensemble of forecasting techniques consisting of Extreme Gradient Boosting (XGBoost), neural network and Long Short-Term Memory (LTSM), and Support Vector Regression (SVR).

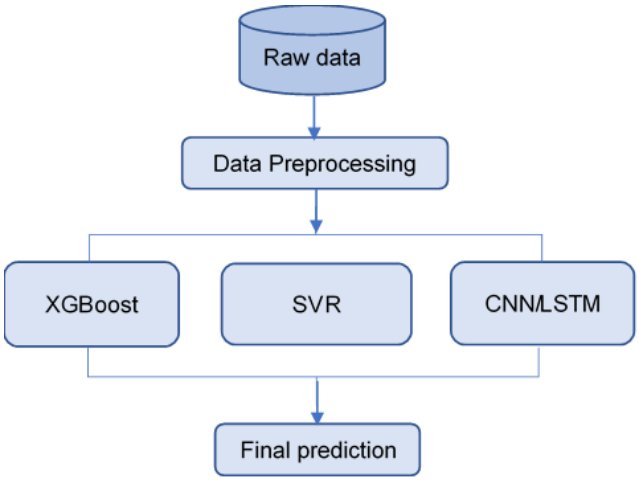

Fig.5 Workflow of the Algorithms

4.1 SVR

SVR with Particle Swarm Optimization (PSO) will be used to deal with the non-linearity and complexity of the data.

The SVR function is as below, denoting the forecasting values.

$$f(x) = \omega\varphi(x) + b$$

The coefficients can be trained by solving the following formulas:

$$\min_{\omega,b,\xi_i,\xi_i^*} \frac{1}{2}||w||^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*)$$

$$s.t \begin{cases} y_i - (< w, \phi(x_i) > +b) \leq \epsilon + \xi_i \\ (< w, \phi(x_i) > +b) - y_i \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases}$$

The polynomial kernel is used in this experiment. The hyperparameters, for example, C and degree, will be adjusted during the experiment by grid search. In the experiment, the Gaussian Radial Basis Functions (RBF) kernel was tried, but an overfitting problem appears. So, the polynomial kernel is the major kernel in the following experiments. The key for SVR in implementation is to turn the time series problem into the

supervised learning problem in the feature map.

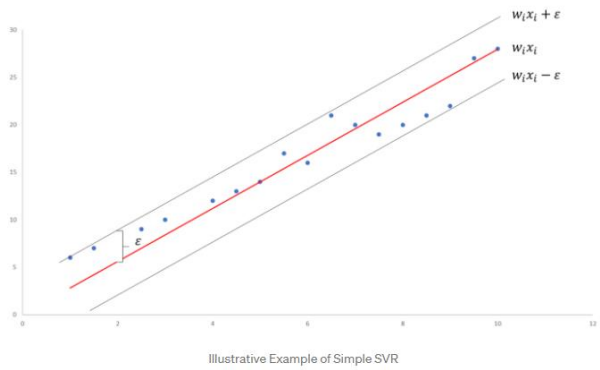

Fig.6 Illustration of Simple SVR

The mathematical expression of SVR is similar to the primal problem of SVM. For parameter C, it is the constraint to control how many penalties should be imposed on the error. When C is very small, SVR is with the soft margin. When C is very big, SVR is with the hard margin. In SVR, the decision boundary is different from SVM. It has an upper bound and a lower bound, which means that any data points falling into this range are not penalized, so ε is introduced to define the range. In the implementation of the SVR algorithm, the default value of ε is used. Tuning C and degree in the polynomial kernel by grid search get a better result.

The SVR algorithm experiment introduces the traditional statistic model, ARIMA, for comparison.

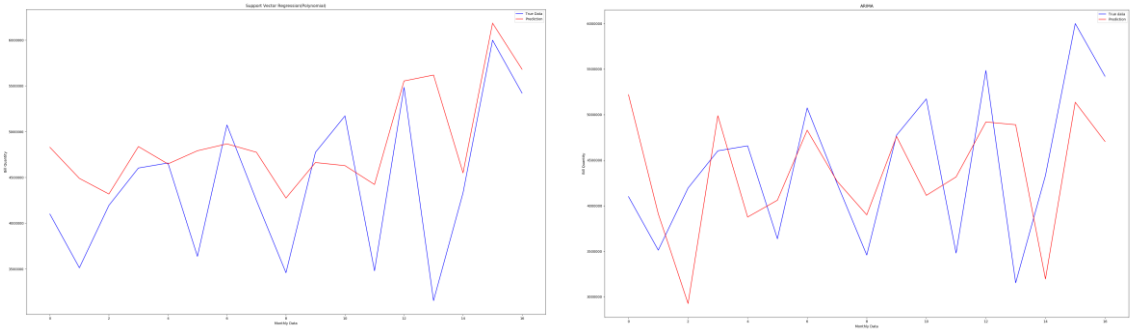

Fig.7 Comparison between the Predicted Data and the True Data (Left: SVR; Right ARIMA)

ARIMA is an easy-to-use model. But in ARIMA, prediction only depends on the history, so it can't take other features into consideration, such as booking quantity. Besides, it cannot support one-hot encoding for categorical data. So, in the experiment, implementation involves grouping all sub-product lines for comparison. On the mathematical side, the prediction of ARIMA is the linear combination for historical records. From these points, SVR with the nonlinear kernel has comparative advantages. In fact, these two models' performances are close when datasets are normalized.

			p	d	q		C	degree
with normalization	refer to past three months	ARIMA	1-3	0-1	0-3	SVR	np.logspace(-10, 5, 16)	1-7
		Best Parameter	2	1	3	Best Parameter	1	1
		RMSE	0.161			RMSE	0.157	

Table 1 SVR and ARIMA Parameters

4.2 XGBoost

XGBoost is an implementation of the gradient boosting ensemble algorithm for classification and regression[4]. Time series datasets can be transformed into supervised learning using a sliding-window representation.

Unlike decision trees, each regression tree contains a continuous score on each of the leaf; XGBoost uses w_i to represent the score on i-th leaf. For a given example, XGBoost will use the decision rules in the trees (given by q) to classify it into the leaves and calculate the final prediction by summing up the score in the

corresponding leaves (given by w).To learn the set of functions used in the model, XGBoost minimizes the following regularized objective:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

$$\text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

Here l is a differentiable convex loss function that measures the difference between the prediction $\hat{y}^i$ and the target y^i . The second term Ω penalizes the complexity of the model.

As we know, XGBoost is the tree-based classifier/regressor, so when we use the XGBoost, there is no need to normalize the training data. For our training data, to keep the same baseline, we still keep the 1-hot encoding.

When calling the XGBoost regressor, we set the learning rate as default, i.e., 1, set the estimator number as 1000, and the early stopping round as 50. The predicted and true data are shown in Fig.8.

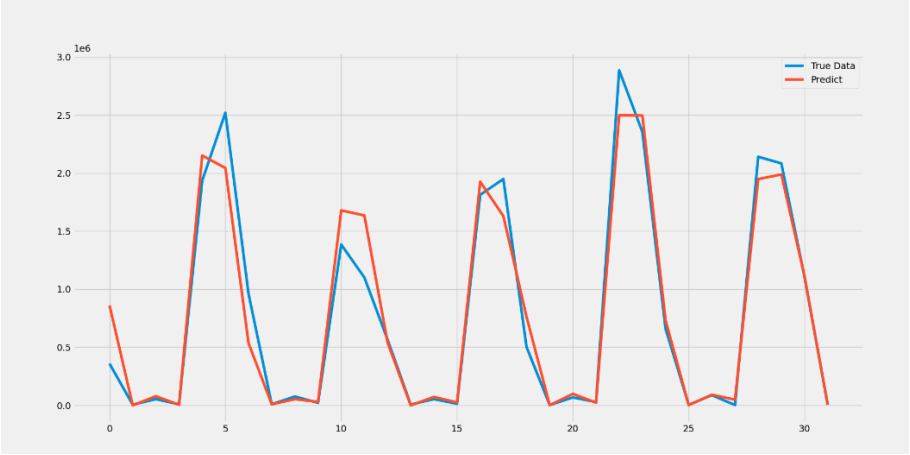

Fig.8 XGBoost Predicted and True Data

To better understand the data and the model and preparing for the future feature reduction, the feature importance is also shown in Fig. 9. From the plot, we can see that the booking quantity (f9) has the highest F score, and followed by fiscal year(f0), month(f2), quarter (f1). This also comply with the preliminary data analysis; the billing quantity has a high correlation with booking quantity and shows the same trend by month in the different fiscal year.

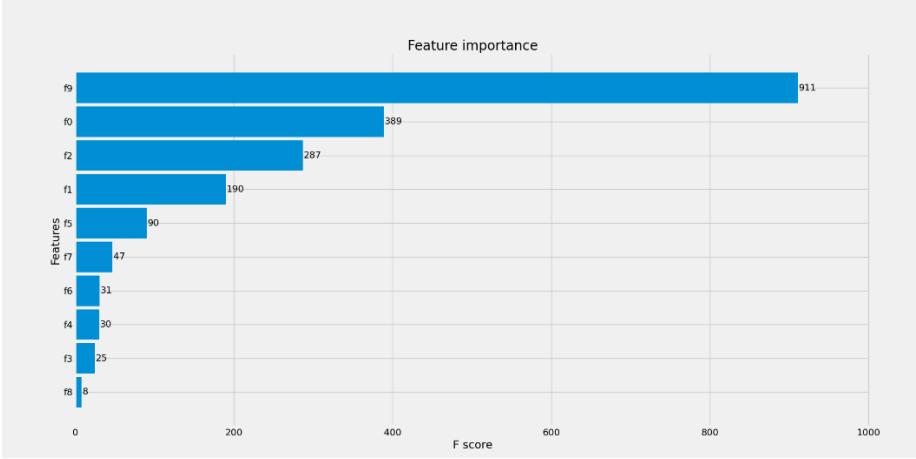

Fig.9 Feature Importance

4.3 LSTM+ CNN

Long short-term memory recurrent neural networks are an improvement over the general recurrent neural networks, which possess a vanishing gradient problem. As stated in Hochreiter et al. [1], LSTM RNNs address the vanishing gradient problem commonly found in ordinary recurrent neural networks by incorporating gating functions into their state dynamics. At each time step, an LSTM maintains a hidden

vector h and a memory vector m responsible for controlling state updates and outputs. More concretely, Graves et al. [2] define the computation at time step t as follows:

$$\mathbf{g}^u = \sigma(\mathbf{W}^u \mathbf{h}_{t-1} + \mathbf{I}^u \mathbf{x}_t)$$
$$\mathbf{g}^f = \sigma(\mathbf{W}^f \mathbf{h}_{t-1} + \mathbf{I}^f \mathbf{x}_t)$$
$$\mathbf{g}^o = \sigma(\mathbf{W}^o \mathbf{h}_{t-1} + \mathbf{I}^o \mathbf{x}_t)$$
$$\mathbf{g}^c = \tanh(\mathbf{W}^c \mathbf{h}_{t-1} + \mathbf{I}^c \mathbf{x}_t)$$
$$\mathbf{m}_t = \mathbf{g}^f \odot \mathbf{m}_{t-1} + \mathbf{g}^u \odot \mathbf{g}^c$$
$$\mathbf{h}_t = \tanh(\mathbf{g}^o \odot \mathbf{m}_t)$$

where σ is the logistic sigmoid function, represents element-wise multiplication, W^u, W^f, W^o, W^c are recurrent weight matrices and I^u, I^f, I^o, I^c are projection matrices.

The network we are using is 3 LSTMs with Dropout Layers; in addition, three layers convolution and ReLU layers are also deployed [3]. In our experiment, we use the serial connection other than concatenation in [3], see fig. 10.

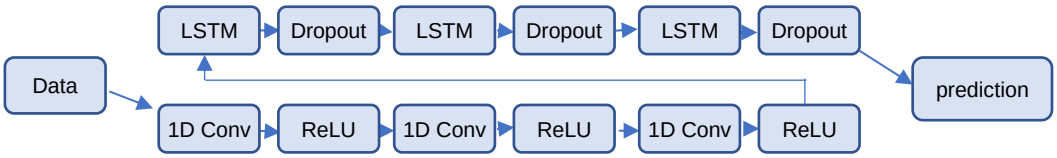

Fig.10 The Improved CNN+LSTM Work Flow

The pure CNN method is also tried before we use the LSTM in our experiment. And then, we add one LSTM (we call it mode 1) and then add two more LSTMs in our improved version (mode 2).
We set the batch size as 16 and run 1000 epochs with early stopping and the validation split as 0.2.

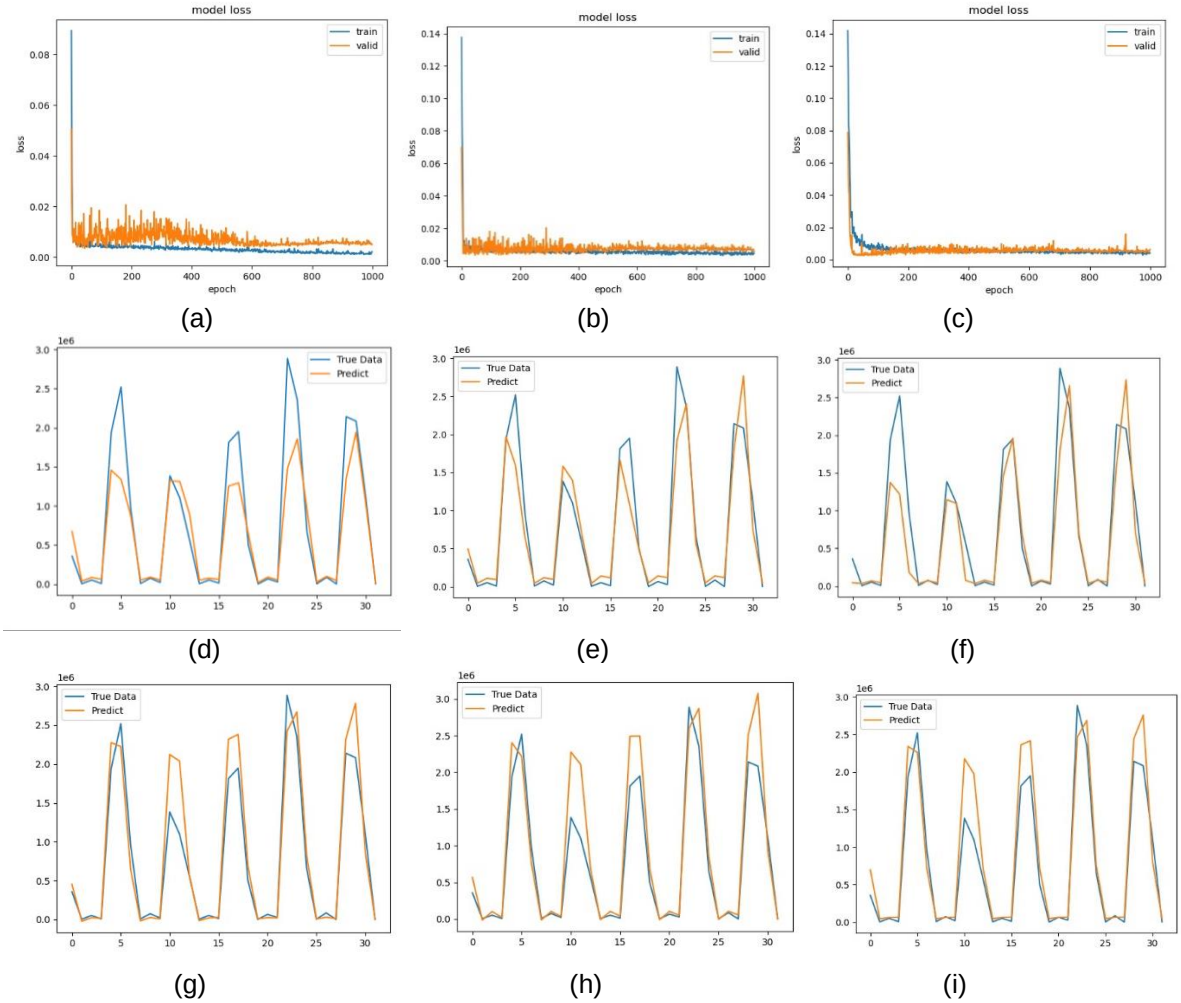

Fig.11 Loss and Result Comparison
(a), (b), (c) is the loss of CNN, CNN+LSTM, CNN+ 3 LSTMs, respectively.
(d), (e), (f) is the predicted data and true data plot of CNN, CNN+LSTM, and CNN+3 LSTMs, respectively.
(g), (h), (i) is the predicted data and true data plot with specific epochs of CNN, CNN+LSTM, and CNN+3 LSTMs, respectively.

Surprisingly, we find that the early stopping doesn't work as we defined and expected, which causes the validation error to go up until 1000 epoch finished. When manually checking the training loss and validation loss, we found that the validation loss is very bumpy along the epochs; there is no way to meet the early stopping requirement. And the rear of the 1000 epochs, the validation error is even worse than that of the beginning. To make it easy, we manually check the easy stopping point, and retrain the model and generate the predicted data as the final results, as shown in Fig.11 (g), (h), (i).

4.4 Result Comparison

Our data is not toy data nor downloaded from the internet; it comes from an electronic parts provider. The data has a big fluctuation along the month of the year. And there are even a lot of negative values for the booking and billing quantity. That means there are some refunds in that period.

To compare the result of all the methods mentioned above, we plot all the predicted results in one figure, fig.12. From the figure, we can see that all the algorithms captured the trend of the data, but it's hard to tell which one is the best.

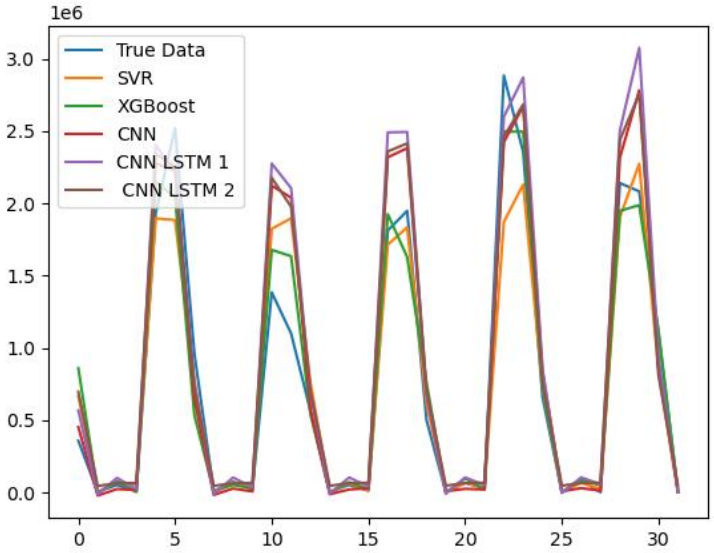

Fig.12 Predicted Data of all Algorithms and True Data

Therefore, we plot all the MSE (Mean Square Error) of all the algorithms, fig. 13. From the figure, we can see that the XGBoost gives the lease MSE, which means it is the best algorithm from our data. And the second-best model is the CNN+LSTM mode 2 with early stopping.

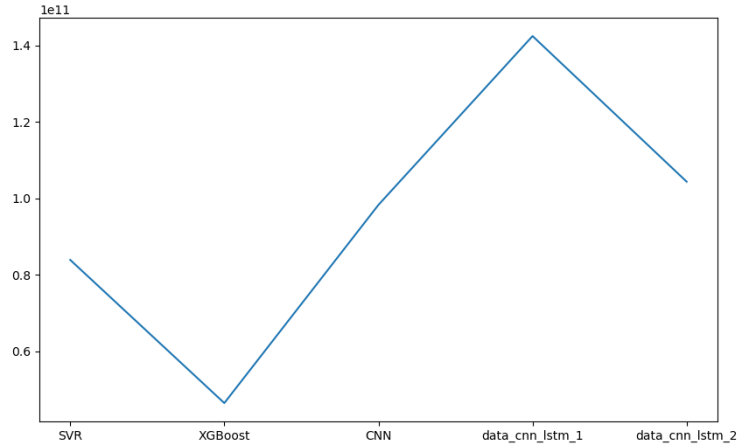

Fig.13 MSE of all Algorithms

5 Conclusion

As we all know, there is a "no free lunch" theorem in the machine learning world, which means no single machine learning algorithm is universally the best-performing algorithm for all problems [13]. Base on this theorem, we know that some algorithms may generally perform better than others on a certain type of problem, but every algorithm has disadvantages and advantages due to the prior assumptions that come with that algorithm. We tried SVR, CNN, CNN+LSTM, XGBoost, and all the algorithms can predict the trend of data. From the fig.13, we can see that the XGBoost outperforms all other algorithms. But maybe after a couple of mouth of years, especially the year of 2020 and 2021, the auto market is dramatically changed due to the chip shortage. We believe, with our ensemble method, we can predict reasonable billing quantity with the historical data.

We will continue this course project after we finish CS229. In the next step, we will focus on the improvement of XGBoost to reduce the MSE by adding more features like competition coefficient in the niche market.

6 Acknowledgement

We want to acknowledge the Stanford 2021 spring CS229 teaching team. It's our honor to learn from such a responsible teaching team, and we did learn a lot from the lectures, discussions, and homework problem sets. The class starts from the linear regression and moves to the naïve Bayesian and neural network. Unsupervised learning and semi-supervised learning are also included. And in the last part, the reinforced learning is introduced. In the TA session, the decision tree, measuring metrics, and ML advice are also introduced. All the knowledge we learn helps us on this project and will, of course, help us to dive deeper into the machine learning field in the future.

7 Reference

[1] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.
[2] A. Graves et al., Supervised Sequence Labelling with Recurrent Neural Networks. Springer, 2012, vol. 385.
[3] Karim Fazle, Majumdar Somshubra, Darabi Houshang, etc., LSTM fully convolutional networks for time series classification, IEEE Access, 6. p1662—1669, 2018
[4] Tianqi Chen, Carlos Guestrin, XGBoost: A Scalable Tree Boosting System, KDD'16, August 13-17, 2016
[5] A Single Scalable LSTM Model for Short-Term Forecasting of Massive Electricity Time Series
Application of machine learning techniques for supply chain demand forecasting
[6] R. Hyndman and G. Athanasopoulos, Forecasting: Principles and Practice, OTexts, Melbourne, Australia, 2018, http://otexts .org/fpp2/.
[7] D. Gujarati, Basic Econometrics, Mcgraw-Hill, New York, NY, USA, 2003.
[8] J. D. Croston, "Forecasting and stock control for intermittent demands," Operational Research Quarterly, vol. 23, no. 3, pp. 289–303, 1972.
[9] T. R. Willemain, C. N. Smart, J. H. Shockor, and P. A. DeSautels, "Forecasting intermittent demand in manufacturing: a comparative evaluation of Croston's method," International Journal of Forecasting, vol. 10, no. 4, pp. 529–538, 1994.
[10] Z. Qi, B.Wang, Y. Tian, and P. Zhang, "When ensemble learning meets deep learning: a new deep support vector machine for classification," Knowledge-Based Systems, vol. 107, pp. 54–60, 2016
[11] Yukimasa Kanedam, Hiroshi Mineno Sliding window-based support vector regression for predicting micrometeorological data 2016
[12] Shuojiang Xu, Hing Kai Chan Forecasting the demand of the aviation industry using hybrid time series SARIMA-SVR approach 2019
[13] No free lunch theorem, https://en.wikipedia.org/wiki/No_free_lunch_theorem