M5 Forecasting – Accuracy

Estimate the Unit Sales of Walmart Retail Goods Using ML Methods

MAFS6010Z Project 3
LAI Cong (20747850)
LIU Jinghui (20745644)
LU Qiaoyu (20736916)
ZHEN Mengnan (20749066)
Department of Financial Mathematics
Hong Kong University of Science and Technology
Dec. 12, 2021

Workload:

For coding part: LIU Jinghui takes part of Time Series models. LU Qiaoyu takes part of XGBoost model. ZHEN Mengnan takes part of LGBM model. LAI Cong takes part of Neural Network.

For report part: All team members take part of their own model's description and modeling process. LIU Jinghui takes part of Introduction and Data Preprocessing. ZHEN Mengnan takes part of Data Description. LU Qiaoyu takes part of Abstract and Model Comparison & Conclusion.

Abstract

In this report, we introduce how to conduct machine learning analysis on Estimating the Unit Sales of Walmart Retail Goods Using ML Methods. We will explain the modeling process and our model details. For some models, we also try to find the best result by adjusting parameters. We evaluate the model performance by *RMSE* scores. We compare each model by their test score and give a conlusion as well.

1. Introduction

In this competition, we use hierarchical sales data from Walmart, the world's largest company by revenue, to forecast daily sales for the next 28 days. The data, covers stores in three US States (California, Texas, and Wisconsin) and includes item level, department, product categories, and store details. In addition, it has explanatory variables such as price, promotions, day of the week, and special events.

For models, we tried time series models, boosting models, such as LGBM and XGBoost, and deep learning models.

2. Data Description

In this project, we will anticipate daily sales for the next 28 days using hierarchical sales data from Walmart, the world's largest firm by revenue. The data comprises item level, department, product categories, and store details for stores in three US states (California, Texas, and Wisconsin). Price, promotions, day of the week, and special events are among the explanatory variables.

We have five datasets in the project:

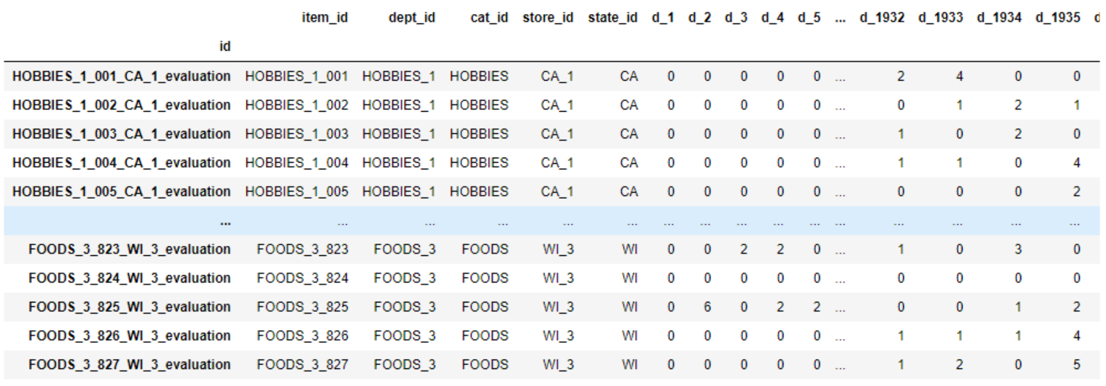
- Calendar - Contains information about the dates on which the products are sold.
- sales_train_validation - Contains the historical daily unit sales data per product and store [d_1 - d_1913]
- sample_submission - The correct format for submissions. Reference the Evaluation tab for more info.
- sell_prices - Contains information about the price of the products sold per store and date.

- sales_train_evaluation - Includes sales [d_1 - d_1941] (labels used for the Public leaderboard)

3. Modeling Process

3.1. Data Pre-processing

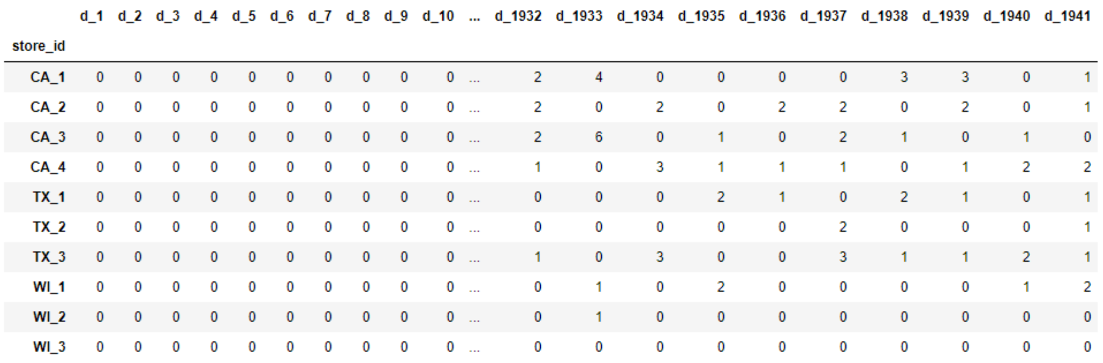
Using sales_train_evaluation dataset for train and validation test (d_1-d_1941), add zero sales for dates d_1942 to d_1969 for further modeling process. Then define the function about memory usage reduction to save memory. Next, melting dataset to aggregate sales by ID. We also merged datasets sales_train_evaluation, calendar and sell_prices. In the end, converted numeric variables into categorical variables for model-building.



	item_id	dept_id	cat_id	store_id	state_id	d_1	d_2	d_3	d_4	d_5	...	d_1932	d_1933	d_1934	d_1935	d
id																
HOBBIES_1_001_CA_1_evaluation	HOBBIES_1_001	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	0	...	2	4	0	0	
HOBBIES_1_002_CA_1_evaluation	HOBBIES_1_002	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	0	...	0	1	2	1	
HOBBIES_1_003_CA_1_evaluation	HOBBIES_1_003	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	0	...	1	0	2	0	
HOBBIES_1_004_CA_1_evaluation	HOBBIES_1_004	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	0	...	1	1	0	4	
HOBBIES_1_005_CA_1_evaluation	HOBBIES_1_005	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	0	...	0	0	0	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
FOODS_3_823_WI_3_evaluation	FOODS_3_823	FOODS_3	FOODS	WI_3	WI	0	0	2	2	0	...	1	0	3	0	
FOODS_3_824_WI_3_evaluation	FOODS_3_824	FOODS_3	FOODS	WI_3	WI	0	0	0	0	0	...	0	0	0	0	
FOODS_3_825_WI_3_evaluation	FOODS_3_825	FOODS_3	FOODS	WI_3	WI	0	6	0	2	2	...	0	0	1	2	
FOODS_3_826_WI_3_evaluation	FOODS_3_826	FOODS_3	FOODS	WI_3	WI	0	0	0	0	0	...	1	1	1	4	
FOODS_3_827_WI_3_evaluation	FOODS_3_827	FOODS_3	FOODS	WI_3	WI	0	0	0	0	0	...	1	2	0	5	

Figure 1 – Train dataset

As shown above, the evaluation and the validation data are in this pattern, where there are many zeros. In order to preprocess the data, we intend to use singular value decomposition because the matrix is so sparse. So we can reduce noise and obtain important parts from data matrix.



	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_10	...	d_1932	d_1933	d_1934	d_1935	d_1936	d_1937	d_1938	d_1939	d_1940	d_1941
store_id																					
CA_1	0	0	0	0	0	0	0	0	0	0	...	2	4	0	0	0	0	3	3	0	1
CA_2	0	0	0	0	0	0	0	0	0	0	...	2	0	2	0	2	2	0	2	0	1
CA_3	0	0	0	0	0	0	0	0	0	0	...	2	6	0	1	0	2	1	0	1	0
CA_4	0	0	0	0	0	0	0	0	0	0	...	1	0	3	1	1	1	0	1	2	2
TX_1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	2	1	0	2	1	0	1
TX_2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	2	0	0	0	1
TX_3	0	0	0	0	0	0	0	0	0	0	...	1	0	3	0	0	3	1	1	2	1
WI_1	0	0	0	0	0	0	0	0	0	0	...	0	1	0	2	0	0	0	0	1	2
WI_2	0	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0	0	0	0	0	0
WI_3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

Figure 2 – Submission

To iterated over the departments, producing a data matrix of unit sales of each product in all 10 stores during the time range of training dataset, the data pattern shown as above. This allowed for pooling of data across the stores, within the same department. Thus, we first performed singular value decomposition on the data matrix and then replacing the data with a reduced-rank approximation of itself before forecasting, where we selected top 90% of the singular values of each product.

In neural networks model, we try to use all (d_1 - d_1941) or a part of (d_1602 – d_1941) sales_train_evaluation dataset for train and validation test. Then, we extract weekday, month, events, and SNAP availability of stores as features and process the events columns to binary (any event:1, no event:0). Finally, 75% or 66.7% of sample data are extracted randomly from the dataset.

3.2. Feature Engineering

In addition, we also considered about the information of prices. On the one hand, prices economically influence customer behavior; on the other hand, prices vary from time to time especially in festivals. Therefore, we also create a data matrix of sales revenue of each product in all 10 stores during the time range of training dataset, which the data pattern is same as a data matrix of unit sales. We tried to predict revenues first and then get predictions of sales by dividing prices. Intuitively, we thought less unit sales for higher prices.

We also consider converting the categorical variables to dummy variables. And we will convert some of the data type in order to save memory usage. Besides, we also do some feature construction based on the original dataset, such as 'price_change_t1' and 'price_change_t365' (price change over 1 day and over 1year).

4. Model Details

4.1. Time Series Models

We tried three kinds of time series models to find the best one. We also tried to combine them by averaging results from each model, to see whether the prediction is improved by stacking or not.

4.1.1. Exponential Moving Average

Exponential smoothing is a rule of thumb technique for smoothing time series data using the exponential window function. Whereas in the simple moving average the past observations are weighted equally, exponential functions are used to assign exponentially decreasing weights over time. It is an easily learned and easily applied procedure for making some determination based on prior assumptions by the user, such as seasonality.

4.1.2. SARIMA

ARIMA models are applied in some cases where data show evidence of non-stationarity in the sense of mean, where an initial differencing step (corresponding to the "integrated" part of the model) can be applied one or more times to eliminate the non-stationarity of the mean function (i.e., the trend). When the seasonality shows in a time series, the seasonal-differencing could be applied to eliminate the seasonal component. In this project, we tried ARIMA and SARIMA models but found that these two kinds of models performed badly and trained very slow.

4.1.3. STL

STL stands for Seasonal and Trend decomposition using Loess. This is a statistical method of decomposing a Time Series data into 3 components containing seasonality, trend and residual. Trend gives you a general direction of the overall data. Whereas seasonality is a regular and predictable pattern that recur at a fixed interval of time. Randomness or Noise or Residual is the random fluctuation or unpredictable change. This is something which we cannot guess. Lastly, loess is a regression technique that uses local weighted regression to fit a smooth curve through points in a sequence. The STL decomposition result of a product shown as below. The top is time series of product unit sales; the second is the trend of the product in different stores; the third plot is the seasonality pattern; the last one is the model residual. We can see that STL model manage the data well especially for later time range, showing obvious seasonal and trend patterns and approximately uniformly distributed residuals.

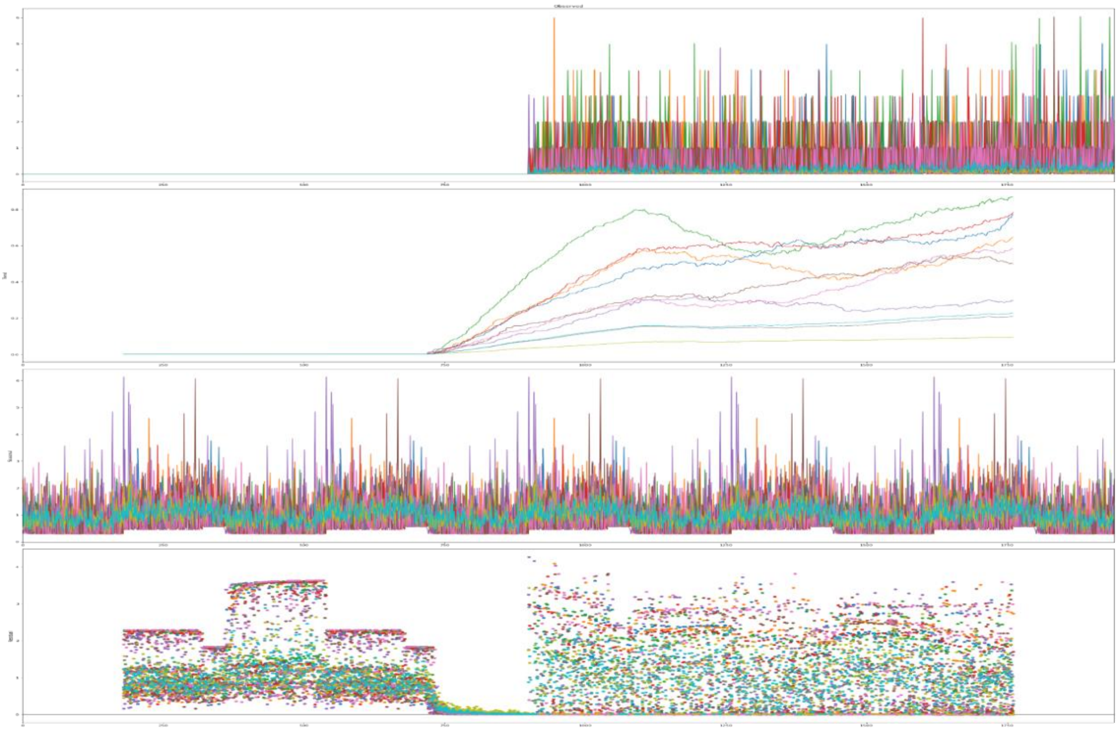


Figure 3 - STL decomposition results

First, we create train (d<1914), validity (1914≤d<1942) and test (d≥1942) datasets for unit sales and sales revenues. In the project, we used *RMSE* to find optimal parameters and models. Then chose the parameter which had the minimum *RMSE* every time to building the model and then predict the sales.

Parameters	**Description**	**Values**
Order	Order for ARIMA, EMA	(0,1,0) to (1,1,1)
Seasonal	Seasonal term or seasonal differencing	365 or none

Table 1- Parameter Description of TS

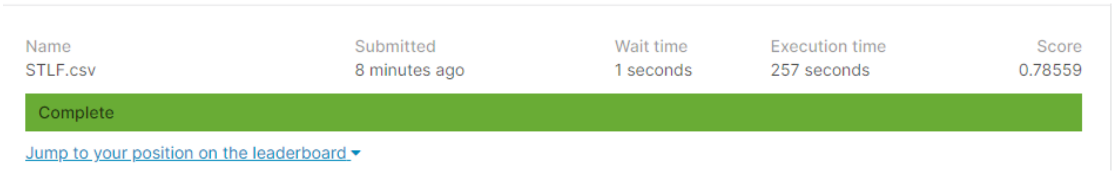
For simplicity, we only show the results of models with optimal order parameters. And because of poor STL results of predicting revenues we stopped further studying of the other models on it.

The results shown as below:

Models	Seasonal	RMSE (Train)	RMSE (Validation)	Kaggle score
EMA	Yes	445006756395	4096894337805	13350202691217
EMA	None	466006756395	4796894337805	
ARIMA	None	405006755397	4099884437826	
SARIMA	Yes	366011756395	3796894337805	
STL	Yes	2.330	2.429	0.780
STL	None	2.920	3.399	0.845

Table 2- *RMSE* of TS

From the table above, we can see EMA, SARIMA models performed extremely bad. Consequently, we select STL model with seasonal = 365 and ARIMA (1,1,1) as forecasting model as the best model among time series models.


Name STLF.csv
Submitted 8 minutes ago
Wait time 1 seconds
Execution time 257 seconds
Score 0.78559

Complete

Jump to your position on the leaderboard

Figure 4 – Submission score of TS

4.2. Extreme Gradient Boosting (XGBoost)

Extreme Gradient Boosting (XGBoost) is an advanced machine learning model to utilize the computation resources deeply, which efficiently implements GBDT algorithm. XGBoost incorporates the greedy algorithm like CART regression tree, traversing all feature partition points for all features, but using a different objective function. In order to constrain the growth depth of regression trees, it put a threshold to limit the improvements of the updated objective function value after splitting from single leaf node. Only when the improvement exceeds this threshold value, it could split successfully. For each time it takes the best result of the previous prediction to further split [2].

XGBoost is widely used in Kaggle competition, as well as many other machine learning projects. In general, it could achieve good results. Hence, we study on XGBoost and compare it with LGBM below.

4.2.1. Strengths of XGBoost

Firstly, whereas traditional GBDT is not designed to handle missing values, XGBoost can automatically apply the missing value handling strategy. In addition, XGBoost supports data sampling and various types of classifiers. It utilizes the first and second derivatives of the cost function at the same time by using second-order Taylor expansion.

Besides, XGBoost optimizes model prediction by avoiding overfitting. There are several parameters which could be adjusted to improve the fitting effect: the setting threshold to limit the increments of objective function after splitting, the maximum depth of trees, number of weak classifiers, learning rate and so on.

4.2.2. Implementation of XGBoost

In order to evaluate the model performance, we split the merged dataset into three parts: the train dataset ('date' before '2016-03-27'), the validation dataset ('date' between '2016-03-27' and '2016-04-25') and the test dataset ('date' after '2016-04-25'). We will train the model using the train dataset and calculate the *RMSE* score of the validation dataset. Then using the parameters with the lowest validation *RMSE* score to do prediction on the test dataset.

The major tuning parameters are 'n_estimators' and 'learning rate', which are described below.

Parameter	Description	Range
n_estimators	Number of weak classifiers in decision tree	[5, 10, 30, 50, 100]
learning_rate	Control the weight reduction coefficient of each weak classifier	[0.01, 0.05, 0.1, 0.2]

Table 3 - Parameter Description of XGBoost

The following table shows the validation *RMSE* score of XGBoost. It is shown that with both number of weak classifiers and learning rate increasing, the *RMSE* scores decrease and finally come to 1.03316. We also study with the cases that learning rate and estimator number increase continuously, but the *RMSE* score shows a rise trend.

Hence, the best XGBoost model in our study is the case when learning rate is 0.2 and number of estimators is 100. Additionally, XGBoost model also leads to memory error or bad allocation error when training on large dataset, because it requires additional space for execution. Hence, we have to drop some features to make sure the memory storage is enough for training.

	n_estimators					
Learning_ rate		5	10	30	50	100
	0.01	1. 18980	1.17648	1.13273	1.10176	1.05982
	0.05	1.14121	1.10051	1.04454	1.03853	1.03640
	0.1	1.09884	1.05771	1.03747	1.03616	1.03411
	0.2	1.05659	1.04008	1.03630	1.03487	1.03316

Table 4 – *RMSE* of XGBoost

The final test score of XGBoost model is 0.86330, which may worsen than LGBM model and Time Series model. The failure reason of XGBoost will be discussed further in next part.

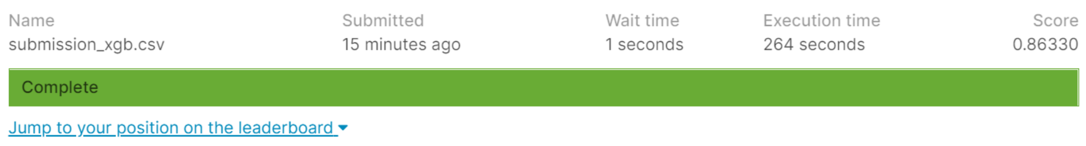

Name submission_xgb.csv
Submitted 15 minutes ago
Wait time 1 seconds
Execution time 264 seconds
Score 0.86330
Complete
Jump to your position on the leaderboard

Figure 5 – Submission score of XGBoost

4.3. Light Gradient Boosting Machine Model (LGBM)

Gradient Boosting Decision Tree (GBDT) is a machine learning model that never fails. Its primary idea is to get the best model by using weak classifier (Decision Tree) iterative training, which has the advantages of good training effect and difficulties in over-fitting. GBDT is not only widely utilized in industry, but also employed in multi-classification, CTR prediction, search sorting and other jobs. It's also a deadly weapon in various data-mining competitions, with GBDT accounting for more than half of Kaggle's competition winners. LightGBM (Light Gradient Boosting Machine) is a framework for implementing the GBDT method, which provides efficient parallel training and has the benefits of faster training speed, lower memory usage, improved accuracy, distributed support, and rapid data processing.

The major goal of LightGBM is to alleviate the challenges that GBDT has in dealing with large amounts of data, so that GBDT may be used more effectively and quickly in industrial settings.

4.3.1. The disadvantages of XGBoost algorithm

XGBoost, a decision tree algorithm based on the pre-sorting method, was the most well-known GBDT tool prior to the advent of LightGBM. feature segmentation point.

The advantage of this pre-sorting algorithm is that it can find the segmentation point accurately. But the disadvantages are also obvious: first, space consumption. Such algorithms need to store the eigenvalues of the data, as well as the results of feature sorting (for example, the sorted index is saved for quick subsequent calculation of segmentation points), which consumes twice as much memory as the training data. Secondly, there is also a large cost in time. When traversing each segmentation point, it is necessary to calculate the splitting gain, which costs a lot of consumption.

4.3.2. The optimization of LightGBM

- Decision tree algorithm based on Histogram.
- Gradient-based One-side Sampling (GOSS): Using GOSS can reduce a large number of data instances with only small gradients, so that only the remaining data with high gradients can be used in the calculation of information gain, which saves a lot of time and space overhead compared with XGBoost traversing all eigenvalues.
- Exclusive Feature Bundling (EFB): Using EFB, you can combine many mutually Exclusive features into one Feature for dimensionality reduction.
- Leaf-wise Leaf growth strategy with Depth limitation: Most GBDT tools use an inefficient Level-wise decision tree growth strategy because it treats leaves at the same level indiscriminately, incurs a lot of unnecessary overhead. In fact, many leaves have low splitting gain, so there is no need to search and split. LightGBM uses a leaf-wise algorithm with depth constraints.

First, we create train (d<1914), validity (1914≤d<1942) and test (d≥1942) datasets.

In the project, we used *RMSE* to find optimal components. Then chose the parameter which had the minimum *RMSE* every time to building the model and then predict the sales.

Parameter	Description	Range
max_depth	Maximum depth of each tree to prevent overfitting.	[3,4,6]

Table 5 – Parameter Description of LGBM

The *RMSE* of each model is shown below:

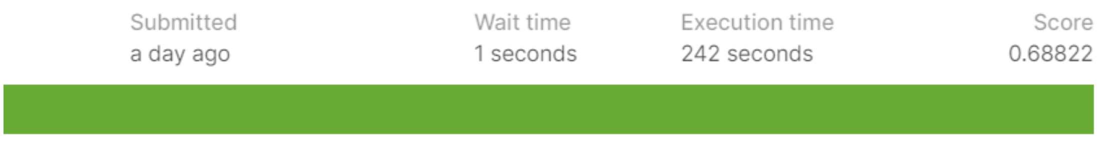
max_depth	Category	RMSE(train)	RMSE(validation)	
3	HOBBIES	1.70545	1.69749	
	HOUSEHOLD	1.64616	1.78818	
	FOODS	4.10085	3.83612	
		7.45246	7.32179	SUM
4	HOBBIES	1.68301	1.67881	
	HOUSEHOLD	1.54367	1.66867	
	FOODS	3.77333	3.50963	
		7.00001	6.85711	SUM

6	HOBBIES	1.65119	1.6589	
	HOUSEHOLD	1.41275	1.53633	
	FOODS	3.309	3.10543	
		6.37294	6.30066	SUM

Table 6 – *RMSE* of LGBM

So, we chose max_depth=6, which had the minimum *RMSE* of validation dataset, for further model prediction.

Finally, we got the evaluation results and create the submission file. The score is 0.68822.



Submitted
a day ago

Wait time
1 seconds

Execution time
242 seconds

Score
0.68822

Figure 6 – Submission score of LGBM

4.4. Neural Network

The final nonlinear method that we analyze is the artificial neural networks. A neural network is an interconnected group of natural or artificial neurons that uses a mathematical or computational model for information processing based on a connectionistic approach to computation. Arguably the most powerful modeling device in machine learning, neural networks have theoretical underpinnings as "universal approximators" for any smooth predictive association.

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. Generally, Neural Network has one input layer, one output layer and hidden layers.

In our project, we choose a simple neural network with 2 hidden layers, "ReLU" as active function, "adam" as optimizer and *RMSE* as loss function.

The loss goes down to almost 2 after 50 epochs.

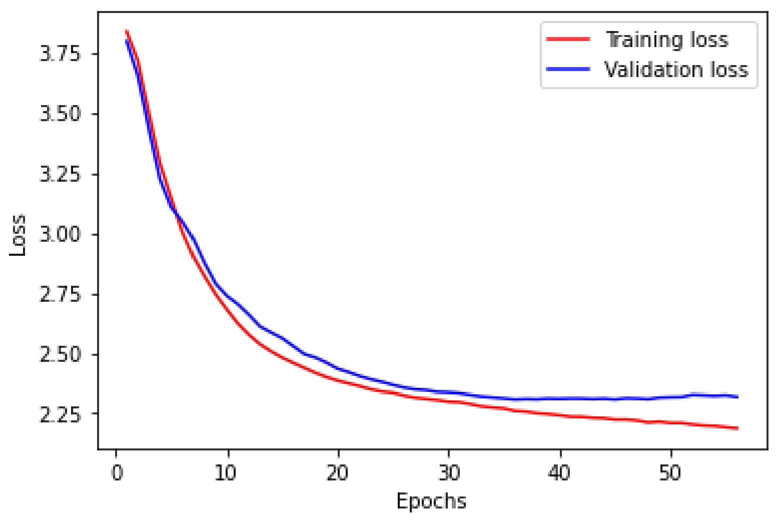


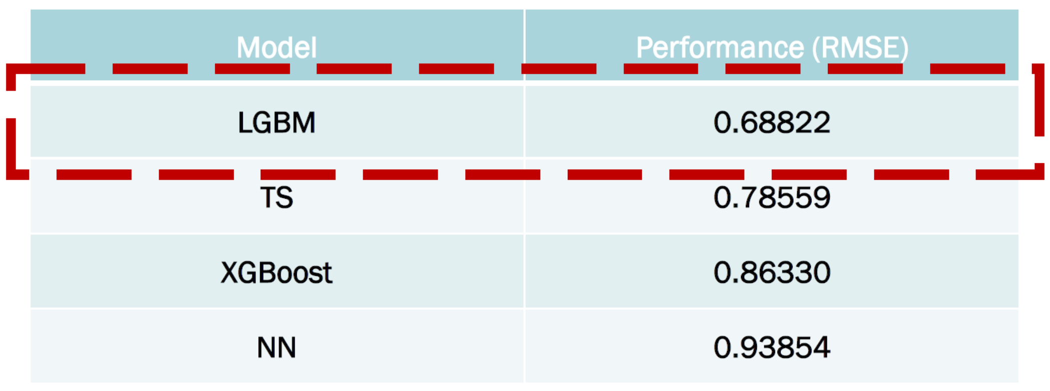
Figure 7- Loss of NN

	75%	66.7%
d_1 – d_1941	2.09361	2.05081
d_1602 – d_1941	0.96589	0.93854

Table 7- *RMSE* of NN

When we train our NN model with smaller dataset, the model has better performance.

5. Model Comparison & Conclusion



Model	Performance (RMSE)
LGBM	0.68822
TS	0.78559
XGBoost	0.86330
NN	0.93854

Table 8- Model comparison

LGBM achieves the highest test score over all models. Compared with other models, LGBM has a high computation speed and low execution memory usage, which is suitable for training large dataset and doing prediction.

In the future work, we may try to improve other models by further feature engineering and data preprocessing.

6. Presentation Video Link

Video title: mafs6010z_Zhen_Liu_Lu_Lai (project 3)
Source: YouTube
Available from: https://youtu.be/ae7z58X2qnI

7. References

[1] M5 Forecasting – Accuracy Estimate the unit sales of Walmart retail goods. Available from: https://www.kaggle.com/c/m5-forecasting-accuracy

[2] 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).