LiBerTY – Store Sales Forcast with Machine Learning

Suhas Anand Balagar* Hardy Leung† Loukya Tammineni‡ Xichang Yu§

Abstract

LiBerTY is an ensemble regression engine to predict the store sales given past sales figures. It employed a variety of robust and general data analysis and machine learning techniques to achieve good result within a short amount of time, competitive to existing best-known result while being robust and generally applicable to other problems.

1 Introduction

The ability to predict the sales of a variety of stores is highly sought out in supply chain logistics, as it finds applications in increasing customer satisfaction and reducing food waste. We are proposing use of multiple supervised learning methods to predict the sales of stores based on time series dataset of Corporación Favorita, an Ecuador based grocery retailer. Ecuador is a country whose economy is strongly dependent on the oil and fluctuates with the price of oil.

Our work focused on a dataset from an ongoing Kaggle competition [1], "Store Sales – Time Series Forecasting". The dataset includes multiple sheets of time series data. In this work, we evaluated the different aspects that might impact the sales in a store such as holiday seasons, oil prices and historical sales data across all stores.

We plan on using different supervised learning models to predict the prices and then evaluate which of those models give out the best results. Supervised learning approach works best in this case, as we have huge time series data of both input and the output parameters mentioned above. We applied several transformation and optimization to improve the data quality in preparation for the optimization. To seek the best store sales prediction, We have evaluated several models including linear regression, gradient boosting algorithms (XGBoost, LightGBM, and CatBoost), random forest, Support Vector Regression (SVR), and Long Short-Term Memory (LSTM). We found XGBoost to be the best-performing individual method, and focused on hyper-parameter tuning via grid search. We further employed ensemble prediction to further improve our results, achieving a notable RMSLE score of 0.425 within our compressed project time-frame.

2 Related Work

The store-sales prediction problem can be directly formulated as a multivariate multiple time-series regression problems. Prior to the modern age of machine learning, Auto-Regressive Moving Average (ARMA) was one of the most well-known technique, first proposed by [2] in his Ph.D thesis, and later popularized by Box and Jenkins [3], according to Wikipedia [4]. ARMA provides a succinct description of a (weakly) stationary stochastic process in terms of two variables, one for the auto-regression (AR), and the second for the moving average (MA). ARIMA, where "I" stands for Integrated and "S" stands for seasonal, are variants of ARMA appropriate for cases when data show evidence of non-stationarity, such as a long-term upward trend, and seasonality.

Random forest, first proposed by Ho [5] in 2008, is a well-established technique that relies on an ensemble learning method for classification or regression based on a collection of decision trees at training time. For

*San José State University, suhasAB@github
†San José State University, ksleung@github
‡San José State University, LoukyaTammineni@github
§San José State University, Codyyu36@github

regression tasks, usually the average prediction of the individual trees is taken, thereby correcting the tendency of individual trees to overfit to their training set.

XGBoost [6] (2016) and CatBoost [7] (2017) are two popular, light-weight open-source software libraries which provide gradient boosting decision trees (GBDTs) framework for a plethora of program languages. They are both well-known for their high performance. LightGBM [8] is another GBDT developed by Microsoft with the explicit goal of speeding up the training process of conventional GBDT by up to 20X while achieving almost the same accuracy. The huge runtime efficiency makes LightGBM a popular GBDT technique in recent years.

Support Vector Regression (SVR) [9] is an extension of the original Support Vector Machine (SVM) algorithm, first proposed in 1962. SVM and SVR are supervised learning models, known for their powerful ability to perform non-linear classification and regression analysis,

Long Short-Term Memory (LSTM) [10] is a recurrent neural network first proposed by Hochreiter and Schmiduber in 1997, and later gained huge popularity in the deep learning community within the last ten years due to its ability to handle the vanishing gradient problem. Though overshadowed by more advanced techniques such as Transformer, it nonetheless is still widely used in time-series prediction due to its relative simplicity and strength in handling long-range time-series analysis.

3 Data Preparation

Our main dataset contained the daily sales figures of 54 stores of Corporación Favorita, from January 1st, 2013 to August 15th, 2017, except Christmas Days when the stores were all closed. We are also given the locations (city and state) of each store. There are a total of 33 different product families, from `Automobile` to `Seafood`. Note that not all stores sell all products. Moreover, there were some stores that opened only after the data collection has begun, and sometimes the sales figures of certain stores were missing over a few months. All stores in the dataset were still in business as of August 15th, 2017, After properly dealing with missing data, Christmas, and stores that were yet to open, we would have a dataset made of (#days) $\times$ (#stores) $\times$ (#families) $= 1688 \times 54 \times 33 = 3008016$ numbers. We were also given the daily oil price over the same period of time, as well as the dates of the regional or national holidays. These information could have a tremendous impact on the accuracy of our prediction.

Our job is to predict the sales figures for each store between August 16th, 2017 and August 31st, 2017, inclusive.

Due to the large amount of missing data in certain stores, or certain product families within a store, we could decide to treat those days as zero sales, in which the seasonality and trend could be severely compromised if we were to use the entire range of data. Figure 2 shows what the aggregate sales look like. It would confuse and possibly severely degrade the quality of our regressors if they were to be trained on such undesirable data.

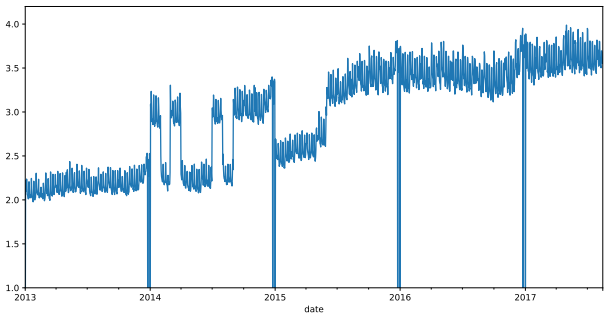

Fig-1: Aggregate sales price over time (original)

Alternatively, one can choose to ignore data that were too old. However, this approach is too conservative, because some stores may not have complete data until late, and we would be forced to trim the dataset to the tune of the worst offender. This approach seriously limit our datasize.

Instead, we propose to "inpaint" the sales figure using patches of data from either a year ago or a year later

(take the average if both are available). Note that this does create a theoretical possibility of a leakage problem since the missing data in the training set may be inpainted from dates in the validation set. We believe this issue is negligible, if at all. Figure 3 shows the aggregate sales over time after the inpainting. Note that the range of sales become much smoother.

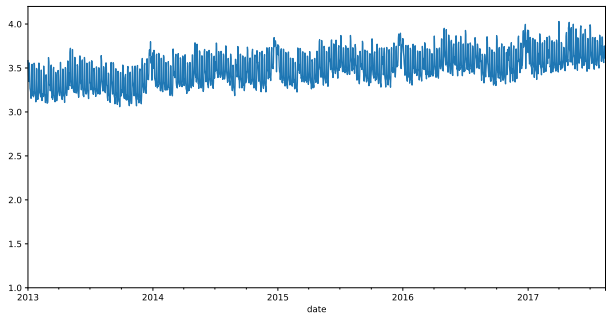

Fig-2: Aggregate sales price over time (inpainted)

In addition, we have performed a few standard data engineering techniques, including one-hot encoding of categorical attributes, standardization of certain attributes such as oil price, as well as generation of derived features such as `day-of-year`, `month-of-year`, `week-of-year`, `day-of-week`, `is-holiday`, and so on.

We have also checked for frequency distribution of data elements as part of data exploration. We have calculated and plotted correlation mapping to establish the correlations between different input and output parameters such as holiday events, oil prices, transactions per store type, dates of salary, natural calamities etc.

Furthermore, we transformed the sales values logarithmically via `log1p()`, i.e. $f(x) = \log(x + 1)$. This is due to the highly non-linear nature of sales (similar to how stock price movement is better modeled geometrically), and as such we believe it is easier for the models to optimize on the logarithm of sales rather than sales. Of course, we shall invert the transformation after prediction.

At this point, we can think of the training data as a list of observations X_i, where $1 \leq i \leq 3008016$. Each observation is keyed by [`store`, `family`], and is accompanied by a total of K features, $X_{i,1}, X_{i,1}, \ldots, X_{i,K}$. Note that we include among these features the most recent S (defaulted to 20) past sales, which we called lagging sales. This transformed and cleaned dataset is then passed to the regressors to obtain the prediction.

4 Methods

In this section we'll briefly discuss our approach. First, we have considered several different prediction engines that work well with time-series analysis – linear regression, LightGBM [8], XGBoost [6], Random Forest, [5], SVR, and LSTM.

We split the dataset, made of 3008016 observations, into a training set $X_{\texttt{train}}, y_{\texttt{train}}$, and a validation set $X_{\texttt{val}}, y_{\texttt{val}}$ at a 75:25 split after randomization. The missing data inpainting allows the entire collection of observations to be used, without suffering from poor data quality due to missing data. Each model is trained on the training set, but evaluated on the validation set which the models do not train on.

We found that XGBoost offers superior performance over other engines. We then perform a grid search to further fine-tune the model for even better performance. We were resource-limited and therefore not able to conduct a more complex hyper-parameter tuning that may give us further benefit.

Finally, we adopt the ensemble learning approach [11] to further improve the performance of our model. Our results will be detailed in the next section.

Throughout our experiments, we observed that, due to the large dataset, we have consistently observed only a small difference between the testing and validation error. However, when we submitted our solutions to be judged by Kaggle, the true error is often higher than the validation error. We should point out that this is not because of overfitting, because we never fit to the validation data. Instead, since we are predicting 16 days worth of sales into the future, some of the past sales numbers were merely projected as they were generated. Therefore, the more further out the date, the less accurate the prediction is.

Experimental Results

5.1 Regressions

We first focused on the four main regression models. Among them, linear regression, LightGBM, and CatBoost were relatively efficient, whereas XGBoost took much longer to run[1]. On the flip side, XGBoost tend to offer the best quality output among the regressors, though still reasonable considering the size of our training data. As a result, we decided to perform a grid search on the faster models, while at the same time we let XGBoost run longer to achieve the best result.

We first focused on XGBoost due to its good performance. However, we were not able to meaningfully perform hyper-parameter tuning due to its relatively slow runtime. The error, more officially the root-mean-square log-error (RMSLE), is as follows:

```
XGBoost | Training 0.401[2]
XGBoost | Validation 0.403
```

We were encouraged to see that the validation error closely tracked the training error.

Next, we did grid search with CatBoost, varying the maximum depth between 6 and 8, and the number of estimators between 1000 and 1800. It turned out the best performing parameters were (8, 1800), suggesting the possibility that the the best hyper-parameters may be outside of the search box. With the hyper-parameter tuning, CatBoost was able to achieve a better result than XGBoost in its default setting:

```
CatBoost | Training 0.338
CatBoost | Validation 0.357
```

5.2 LSTM

We then went from regression to time-series prediction. The pipeline for the LSTM flow was different, because it was set up to be a time-series predictor. Earlier, each regressor treated each day as a datapoint, with the corresponding past sales treated as features. In the case of LSTM, we would chronologically first split the dataset into a training set and a validation set. Then, we would create a time series of data for $X_{\texttt{train}}$ that includes the past 16 days for each row, and a time series of data for $y_{\texttt{train}}$ which includes the future 16 days of data for every row.

We built both a simple one-layer LSTM model, and a three-layer LSTM model, and trained both models for 800 epocs each. Unfortunately, due to the nature of the model we built, we were mostly treating this as a univariate time-series prediction rather than a regression with large number of engineered features. Unfortunately, we were not able to get good results, observing RMSLEs that exceed 2.

5.3 Ensemble

In recent years, ensemble learning has become an increasingly popular approach in machine learning [11]. Ensemble methods are able to take advantage of the strength of a diverse class of models to improve accuracy, and to alleviate individual shortcomings and tendencies to overfit.

Typically, the ensemble process is made of three steps: (1) generation, (2) pruning, and (3) integration. First we would independently generate models based on the specification; then we would prune away redundant models, but caution must be taken to ensure that pruning does not happen at the expense of accuracy; finally we would integrate and engage the models to increase the accuracy of prediction.

In our experiment, our ensemble includes Ridge, random forest, XGBoost, and SVR. Our results are as follows:

```
Random Forest | Training 0.445
Random Forest | Validation 0.454

Ensemble | Training 0.421
Ensemble | Validation 0.411
```

All in all, we believe we did a reasonable job with all our models. We did, after all experiments were done,

[1] Unfortunately, we did not consistently record the experimental runtime, and hence were only able to offer qualitative commentaries.

[2] Our apology for the poorly rendered tables. The pandoc template had a bug and failed miserably in creating even basic Markdown tables.

submit the result to Kaggle, and the results are as follows:

	Validation	Kaggle
XGBoost	0.403	0.440
CatBoost	0.357	0.570
Random Forest	0.454	0.432
Ensemble	0.411	0.405

It is interesting that CatBoost, which performed the best, failed to perform in the Kaggle testset. We believe this can be partially explained by the fact that we were to project 16 days into the future. Therefore, a more aggressive estimator, despite the success in training and validation, faired so poorly when tested with Kaggle data.

6 Discussion

We believe we have achieved encouraging results, despite not ranking among the top 100 in the Kaggle competition. Due to the fact that this is an unranked competition, there are a significant amount of code sharing. In fact, 16 out of the top 25 submissions were *identical*, since the complete source code was publicly shared and easily reproduced. Moreover, we observed a significant amount of cherry picking, which goes against as multiple submissions were allowed and the top submission was used to rank the entries.

In addition, we observed that many of the entries used Kaggle submissions as a validation criteria. Many took on huge undertakings of data spelunking and feature engineering. We did not engage in either activities, but were instead more interested in learning and exploration.

That said, I believe if we have time, there is a great deal of opportunity we have not explored yet. Here we shall name a few.

First of all, we were aware of, but did not deal with, the correlations between product categories. Each of our regression model is built to predict the future of $54 \times 33 = 1782$ time series (one per store per product family). Another school of thought would be to create 1782 models, so each is focused on a single time-series, but lose track of the cross-correlations between stores and between product families. Should we take advange of that?

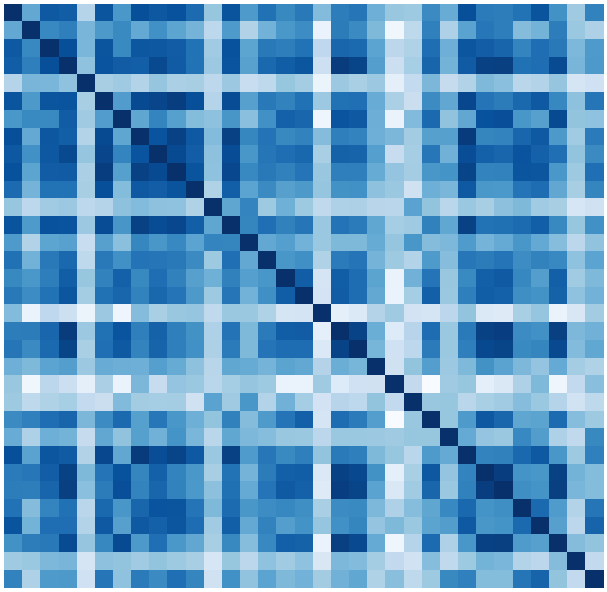

Fig-3: Correlation between product families.

Second, we could be leaving a lot of performance on the table by not performing more exhaustive grid searches. Would the ensemble benefit from better tuned individual models? Being able to manage the size of the dataset and training time could be crucial. We did not consider whether we should perform a principle component analysis (PCA) to reduce the dimensionality of our dataset, and we should definitely consider it if we were to continue the work.

8 Conclusions

In this work, we have presented LiBerTY, an ensemble regression engine that successfully predicts the store sales, which successfully employed a variety of data engineering and machine learning techniques.

References

[1] "Store sales - time series forecasting," *Kaggle*. [Online]. Available: https://www.kaggle.com/competitions/store-sales-time-series-forecasting.

[2] P. Whittle, *Hypothesis testing in time series analysis*, vol. 4. Almqvist & Wiksells boktr., 1951.

[3] G. T. Wilson, "Time Series Analysis: Forecasting and Control, 5th Edition , by George E. P. Box , Gwilym M. Jenkins , Gregory C. Reinsel and Greta M. Ljung , 2015 . Published by John Wiley and Sons Inc. , Hoboken, N," *Journal of Time Series Analysis*, vol. 37, no. 5, pp. 709–711, 2016, [Online]. Available: https://ideas.repec.org/a/bla/jtsera/v37y2016i5p709-711.html.

[4] Wikipedia contributors, "Autoregressive–moving-average model — Wikipedia, the free encyclopedia." 2022, [Online]. Available: https://en.wikipedia.org/w/index.php?title=Autoregressive%E2%80%93moving-average_model&oldid=1108230487.

[5] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, 1995, vol. 1, pp. 278–282 vol.1, doi: 10.1109/ICDAR.1995.598994.

[6] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[7] A. V. Dorogush, V. Ershov, and A. Gulin, "CatBoost: Gradient boosting with categorical features support," *arXiv preprint arXiv:1810.11363*, 2018.

[8] G. Ke *et al.*, "LightGBM: A highly efficient gradient boosting decision tree," in *Advances in neural information processing systems*, 2017, vol. 30, [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[9] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," *Advances in neural information processing systems*, vol. 9, 1996.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] L. Rokach, "Ensemble-based classifiers," *Artificial intelligence review*, vol. 33, no. 1, pp. 1–39, 2010.