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 employs multiple data analysis and machine learning techniques that achieves good result within a short amount of time, competitive with 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 the use of multiple supervised learning methods to make such prediction based on time series dataset of Corporación Favorita, an Ecuador based grocery retailer. Ecuador is a country whose economy is strongly dependent on oil, and fluctuates in energy price has significant impact on consumer behavior.

Our work focused on a dataset obtained from an ongoing Kaggle competition, "Store Sales: Time Series Forecasting" [1]. The dataset is made of multiple sheets of time series data with large number of observations and variables. In this work, we evaluated the different aspects that might impact the sales in a store such as seasonality, oil prices, product categories, and historical sales data across all stores.

We considered several different learning models to predict the sales figures and then evaluate which of those models offer the best performance. 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 considered several promising models including linear regression, random forest, gradient boosting decision trees (XGBoost, LightGBM, and CatBoost), 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 employed ensemble prediction to further improve our results, achieving a notable RMSLE score of 0.405 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 problem. Prior to the modern age of machine learning, Auto-Regressive Moving Average (ARMA), was one of the most well-known techniques, first proposed by [2] in his Ph.D thesis, and later popularized by Box and Jenkins [3], according to Wikipedia. ARMA provides a succinct description of a (weakly) stationary stochastic process in terms of two variables, one for the auto-regression (AR), and one for the moving average (MA). ARIMA and SARIMA – I for Integrated and S 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, such as weekly or quarterly variations.

Another approach we considered was random forest, first proposed by Ho [4] in 2008, a well-established technique that relies on an ensemble learning method

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

for classification or regression based on a collection of decision trees at training time. For 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 [5] (2016) and CatBoost [6] (2017) are two popular, light-weight open-source software libraries that offer cross-platform gradient boosting decision trees (GBDTs) frameworks. Unlike random forests, the decision trees in GBDTs are built additively and dynamically rather than created upfront. LightGBM [7] 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 made LightGBM more popular in recent years. All three models are well-known for their high performance.

Support Vector Regression (SVR) [8] 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 handle non-linear classification and regression analysis. SVM and SVR can often outperform their linear counterparts in domains that are highly non-linear, though they may suffer from long runtime and lack of explainability.

Long Short-Term Memory (LSTM) [9] 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 overcome the vanishing gradient problem. Though now overshadowed by more advanced techniques such as Transformer, it nonetheless is still widely used in time-series prediction due to its relative simplicity in setup 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 were 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 did not exist at the beginning of the data collection, 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, If we could 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) $= 1,688 \times 54 \times 33 = 3,008,016$ 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.

We had to first address the large amount of missing data in certain stores, or certain product families within a store. One approach was to set the sales figure on those days to zero. However, doing so could severely compromise seasonality and long-term trend if we were to use the entire range of data. Figure 2 shows what the aggregate sales look like. It would confuse and degrade the quality of our regressors if they were to be trained on such incomplete data.

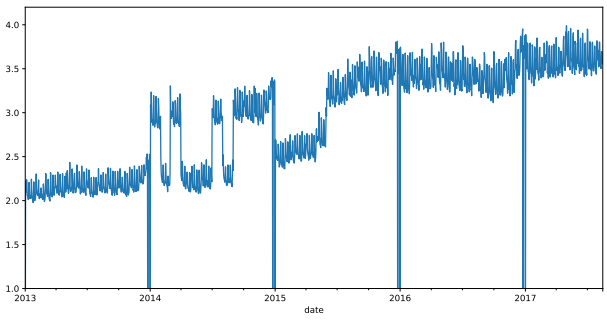

Fig-1: Aggregate sales price over time (original)

Alternatively, one can choose to ignore all data before a certain cut-off date. However, this approach could be too conservative, as some stores only opened recently, and we would be forced to trim the dataset to

the tune of the worst offender. This approach poses serious limit on our training data.

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, but we believe this issue is negligible. Figure 3 shows the aggregate sales over time after the inpainting. Note that the range of sales become much smoother.

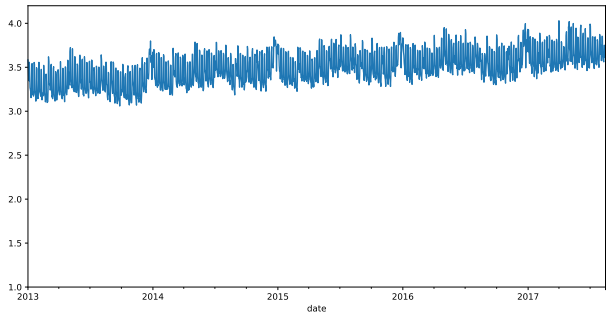

Fig-2: Aggregate sales price over time (inpainted)

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.

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 linear models to optimize on the logarithm of sales rather than sales. Needless to say, we would revert the transformation after prediction.

At this point, we can think of the training data as a list of observations X_i, where $1 \leq i \leq 3,008,016$. 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 Experimental Results

We will now discuss our approaches and experimental results. We have evaluated and will adopt a variety of prediction models that work well with time-series analysis – linear regression, XGBoost [5], CatBoost [6], LightGBM [7], Random Forest [4], SVR [8], and LSTM [9].

We split the dataset, made of 3,008,016 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[1]. The missing data inpainting technique allows the entire collection of observations to be used, without suffering from poor data quality due to missing data. Each model was trained on the training set, but evaluated on the validation set which the models do not train on.

Due to our large dataset, throughout our experiments 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 could be notably different from the validation error. We should point out that this is the result of overfitting, because we never fit our model to the validation data. Instead, since we are predicting 16 days worth of sales into the future, we would have to make predictions one day at a time. A sales prediction on day N into the unknown would be treated as past sales on day $N + 1, N + 2, \ldots$ and so on. As a result, we expect a certain amount of quality degradation as we predict further and further into the future.

[1] We use a different experimental setup for LSTM.

4.1 Regression Models

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[2]. On the flip side, with default settings, XGBoost tended to offer the best quality output among the regressors, and thus we decided to let XGBoost run longer to achieve the best result. Sadly, we were not able to meaningfully perform hyper-parameter tuning on XGBoost due to its relatively slow runtime. The XGBoost error, more officially the root-mean-square log-error (RMSLE), is as follows[3]:

Model	RMSLE
XGBoost	Training 0.401
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), hinting at 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:

Model	RMSLE
CatBoost	Training 0.338
CatBoost	Validation 0.357

4.2 LSTM

In parallel, we also evaluated LSTM recurrent neural networks to make 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 would look at each day as a datapoint, with the corresponding past sales treated as features. In the case of LSTM, however, we would chronologically first split the dataset into a training set and a validation set at a 90:10 ratio. 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. 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 compared to the other models, due to our inexperience and time constraint, and the omission of significant amount of information that were not captured or modeled by LSTM (or by us). That said, we have made significant improvement with our LSTM results as of late, and we are confident that this approach will prove fruitful given time.

4.3 Ensemble

In recent years, ensemble learning has become an increasingly popular approach in machine learning [10]. 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. In other words, an ensumble approach helps lower the variance of predictions and reduce generalization error.

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 regression [11], random forest, XGBoost, and SVR. Our results are as follows:

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

[3]Our apology for the minimally rendered tables. The given pandoc template had a bug and failed miserably in creating even the most basic Markdown tables.

Model	RMSLE	
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, submit the result to Kaggle, and the results were as follows:

Model	Validation	Kaggle
XGBoost	0.403	0.440
CatBoost	0.357	0.570
Random Forest	0.454	0.432
Ensemble	0.411	0.405
LSTM	*	0.870

We should point out that (1) our LSTM model was developed using a different experimental setup, and hence we were not able to show a comparable validation results, and (2) we did not use Kaggle results as a mean to turn our hyper-parameters.

It is interesting that CatBoost, the best performer in our experiments, failed to do well in the Kaggle testset. One explanation is that that we were asked to project 16 days into the future, and hence the accuracy of the prediction is subject to other factors such as noise, and how aggressive the estimators are. It is highly speculative, but possible, that the ensemble model, due to the element of a consensus, may take a more conservative approach to prediction, and ended up reducing the variance in its performance.

5 Discussion

We believe we have achieved encouraging results, ranking close to the top of the competition (as of the writing of this paper, the top Kaggle RMSLE score was 0.378). As this is an unranked competition, there are a significant amount of code sharing, and we saw the top-100 board littered with submissions built on top of the same codebase (some even used as-is). Moreover, we observed a significant amount of cherry picking, since multiple submissions were allowed and the best-performing 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 robust approaches. We merely saw Kaggle competition as a data point of reference, rather than an end goal (our real goal is to learn about machine learning).

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

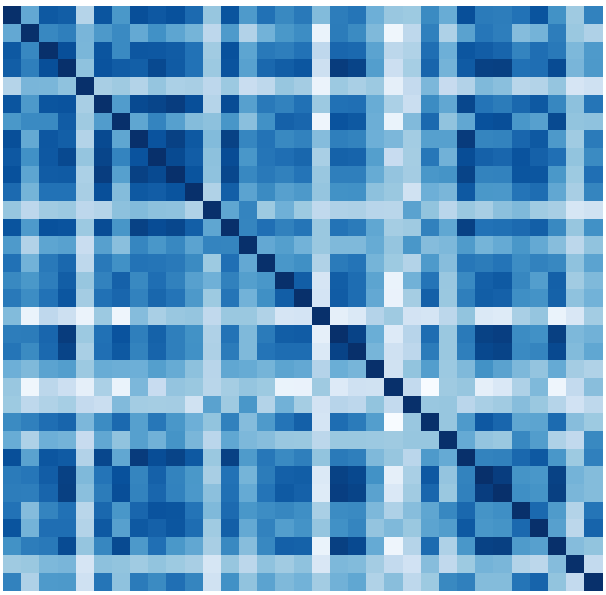

Fig-3: Correlation between product families.

First of all, we were aware of, but did not deal with, the correlations between product categories shown in Figure 3, where evidently certain product families are highly correlated to each other. We could have created additional features such as `average-sales-across-family`, `average-sales-across-store` that are far smoother than just sales figures. Each of our regression model was 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 across stores and across products. Is there merit to the multi-model approach?

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? If we had time to investigate model reduction techniques such as principle component analysis, would that in turn speed up our models and allow for more tuning? That said, we are still very happy that our model performed very well. Most importantly, it is robust and what we learned is generalizable to other problems that we may encounter in the future.

5 Discussion

In this paper, we presented LiBerTY, an ensemble regression engine which incorporated a variety of data engineering and machine learning techniques to successfully predicts store sales.

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] 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.

[5] 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.

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

[7] 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.

[8] 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.

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

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

[11] G. C. McDonald, "Ridge regression," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 1, no. 1, pp. 93–100, 2009.