Model Report on Rossmann Dataset

Liqi Gu, Shuai Jiang, Shengtao Zhong, Runying Jiang
lglu20@soton.ac.uk, sj4n20@soton.ac.uk, sz3g20@soton.ac.uk, rj1u20@soton.ac.uk

Abstract: This report discusses a selection of models to solve the **Rossmann Store Sales Prediction** problem, which is based on a time series dataset. First, we analysed the problem and defined the objective. In order to achieve this goal, we went through a pipeline of data preprocessing, including data cleaning, feature engineering/selection and feature scaling, then several dirty models were made to be the baseline. Afterward, more complicated models (XGBoost, LSTM, Informer) were implemented and fine-tuned to make more precise predictions. Finally, we simply averaged the results of different models and achieved a good performance.

1. Problem Analysis

The Rossmann Store Sales training dataset comprises some attributes of 1115 different stores in Germany, along with totally 1017209 sales records from 2013-01-01 to 2015-07-31, while the objective of this problem is to predict the sales of these stores during the period from 2015-08-01 to 2015-09-17. A rational assupmtion of sales is that they are affected by the seasonality and periodicity, thus, we define the problem as a time-series regression task. The performance is evaluated by Root Mean Square Percentage Error according to kaggle:

$$\text{RMSPE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{y_i}\right)^2}$$

In the next few sections, we will do data preprocessing and train several models to minimise the error on the test dataset.

2. Date Preparation

2.1. Data cleaning

In the aspect of data cleaning, we deal with the missing values, and then drop the outliers according to the distribution structure of features.

2.1.1. Missing Value

The missing data is statistically analysed and shows in **Table I** and the list below

- train.csv - historical data (2013-01-01 to 2015-07-31) including Sales.
  The training dataset has 1017209 rows and 9 colomns with no missing data.
- test.csv - historical data (2015-08-01 to 2015-09-17) excluding Sales
  The testing dataset has 41088 rows and 8 columns with only 11 missing data in *Open* column.
- sample_submission.csv - a sample submission file in the correct format
- store.csv - supplemental information about the stores
  The store dataset has 1115 rows with 10 columns and there are some missing values

Name	Dataset	Type	Missing Number
Open	test	float64	11
CompetitionDistance	store	Int	3
CompetitionOpenSinceMonth	store	float64	354
CompetitionOpenSinceYear	store	float64	354
Promo2SinceWeek	store	float64	544
Promo2SinceYear	store	float64	544
PromoInterval	store	object	544

Table I: The Number Of Missing Data

We notice that the missing value in *CompetitionOpenSinceMonth*, *CompetitionOpenSinceYear* indicates the store faced no competition with other stores. In the **Section 2.2.2**, we will compute the number of months that the competitor had opened, and fill the missing number as zero. We do the same processing to the missing value in *Promo2SinceWeek* and *Promo2SinceYear*. Then we use the median value to impute *CompetitionDistance* since the distribution is skewed. As for *PromoInterval*, we fill the missing value with a new category, indicating the store had no promotion. Also, in the testing dataset, we assume the store opened if the opening status is not given.

2.1.2. Outliers

An outlier is an observation that lies an abnormal distance from other values in a random sample from a population. To avoid the influence of those extreme points, we set up a Quantile filter to remove them. We only keep the data between upper and lower quartiles, which is the data have a difference between 75th and 25th percentiles.

2.2. Feature Engineering

Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data.

2.2.1. Decompose features

- Split String value: Original *PromoInterval* feature describes the consecutive intervals *Promo2* is started, naming the months the promotion is started anew. E.g. "Feb,May,Aug,Nov" means each round starts in February, May, August, November of any given year for that store. We split the string and leave it for further processing.
- Mapping String value : Since the model cannot handle strings, we digitally encode the]*Storetype*, *Assortment*, and *Stateholiday* features which is mapping the string value from a∼d to 0∼4. Besides, We map the processed *PromoInterval* to numbers from 1 to 12 and add a new feature *IsPromoMonth* which shows that whether the store participate in the promotion in the current month.

2.2.2. Transform features

- Transform datetime value: The date format cannot be processed in the model so we split the datetime into three features: *year*, *month* and *day*. As we have already stated in the data exploration report, there are repeated patterns of the week of the year and day of the week, we also add them as a feature.
- Transform competition and promo start data: In the original data set, *CompetitionOpenSinceYear* and *CompetitionOpenSinceMonth* features gives the approximate year and month of the time the nearest competitor was opened, which can not better express its internal meaning and the specific relationship between sales. Therefore We build the feature *CompetitionOpen*, which describes the number of months the store's current date is away from the competition opening. In the same way, *Promo2SinceYear* and *Promo2SinceYear* features show that the year and calendar week when the store started participating in *Promo2*. According to these two features, we calculate the number of months from the store's current date to participating in promo2 as *PromoOpen*.

2.2.3. Create new Features

We also join the store dataset and train dataset to create more features:*SalesPerDay*, *CustomersPerDay*, *SalesPerCustomersPerDay*, *StoreGroup'* and from **1**, it can be seen that they work really well.

- Add *StoreGroup*:Based on common sense that store sales are usually relatively stable, we divide the stores into four groups by Quantile filter based on sales to help with the prediction. And the counts and Sales mean of each group are listed in the following table.
- Add *SalesPerDay*,*CustomersPerDay*: We add the average customer and average sales of the store as the new features*SalesPerDay*,*CustomersPerDay* .
- Add *SalesPerCustomersPerDay*: Since we have *Customers* feature which indicates the customer flow in train set but we don't have it in test set, to fully use all the features given, we take the average sales of user purchases as a new feature *SalesPerCustomersPerDay*.

Group	count	Sales mean
1	210160	4516
2	207646	5934
3	212232	7210
4	213187	10052

2.3. Feature selection

Feature selection refers to the process of reducing the features for processing and analysis. the **Figure1**, we can find even the least relative feature *StateHoliday*, *IsPromoMonth*(not showing in the figure),is still useful to a certain extent. Besides, after doing some experiments of choosing only the top importance features and comparing the performance, we decide to keep all the related feature s. In addition, Some features such as *CompetitionOpenSinceYear*, *CompetitionOpenSinceMonth*,*Promo2SinceYear* and *Promo2SinceYear* has been transformed in the Feature Engineering section, so we delete them. MoreOver, the feature *Customer* does not show in test dataset, we delete it as well.

2.4. Feature Scaling

Traditionally, all of the numerical data should be normalized to ensure a better model performance.

- *CompetitionDistance*: We apply the standard scaler to this numerical feature though it has alomost no effect on the performance of the models.
- *Customers*: it has been deleted while training.
- *Sales*: The target of our prediction. It is not reasonable to apply the standard scaler on target value, instead the log transformation is used to make it less skewed.

3. Model Training

3.1. Train many quick and dirty models

Table II shows the performance of the quick models with dirty data. We use the Nested Cross-Validation instead of the traditional one to split the time series data without causing data leakage. We divide the training set into different subsets by month. The first subset is data from January to March, and we add April to the second one and so on , add month one by one to prevent the dependency in time series data. The training of Random forest regression is relatively slow and it behaves a little bit worse than the other two models. The nonlinear characteristics of random forest regression have more advantages than linear algorithm but random forest regression has its own limitations in the time series problem, which is that the prediction range of random forest is constrained by the highest and lowest label values in the training data. Because SVR model is sensitive to missing data, the performance of the model is not as good as random forest.

Model	mean of validation	std	RMSPE of test
Random forest Regressor	0.45740	4.7485e-05	0.48810
Linear regressor	0.46951	5.8273e-05	0.45215
SVR	0.45740	4.7485e-05	0.45215

Table II: Quick models

3.2. XGBoost

The common regression model has a high bias (too simple, the fitting ability of the algorithm itself is poor) or large variance (too complex, the stability and robustness are not strong). Ensemble learning can combine multiple weak regression models to get a better one. XGBoost is a model of ensemble learning, which is widely used in data science competitions and industry. XGBoosting uses CPU multithreading, introduces regularization term, and adds pruning to control the complexity of the model based on the traditional boosting. It has the characteristics of parallel processing, high flexibility and built-in cross-validation. It also has a good performance in solving time series problem in the past few years.

3.2.1. Important features

Figure 1 shows that *Day* is the most important which is what we expected, since it is a time series dataset. Moreover, all of the features related to time such as *CompetitionOpen WeekofYear* and *Day of Week* ranks high. As the sales of each store vary greatly the *store* is also highly related to the sales. We also found that whether it is the promotion month and whether the store has a continuous sales policy seems to have little effect on sales.

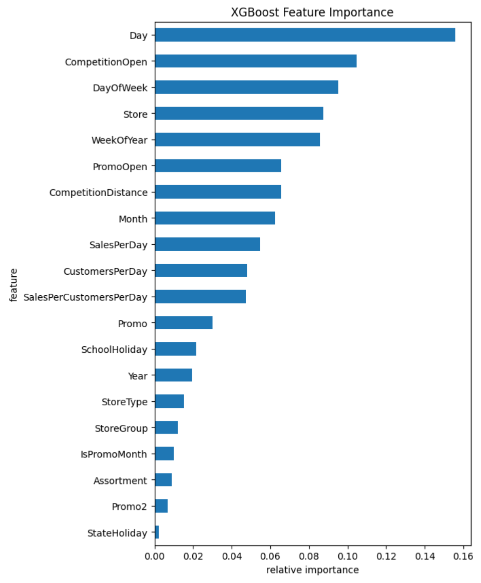

Figure 1: Important features by XGBoost

3.3. LSTM

A popular sequence modeling method in deep learning is the LSTM (Long short-term memory) neural network. Unlike the machine learning method, the neural network architecture is able to extract features automatically. Specifically, the LSTM or other RNN-based model can learn the dependencies within a sequence of data. In this sales prediction problem, the sales can be affected by some factors in the previous days or in the next few days. For example, if the store will start a promotion tomorrow, then it is more likely to have low sales today. However, there are much more dependencies that we cannot perceive intuitively, so we come to the neural network.

Figure 2 shows the structure of our proposed neural network. The model contains a forward LSTM cell and a backward LSTM cell to extract the time series features, as well as a fully connected layer to deal with the non time series features. The time series features, which keep changing to a store and are not deterministic, include *Open, Promo, StateHoliday, SchoolHoliday, Events* The non time series features are fixed attributes to a store, or the changing is deterministic and have no time dependencies, including *Store, DayOfWeek, StoreType, Assortment, CompetitionDistance, Promo2, etc.* The features are either categorical or numerical while the neural network can only accept numerical inputs, thus, we turn the categorical features into learnable embeddings[1].

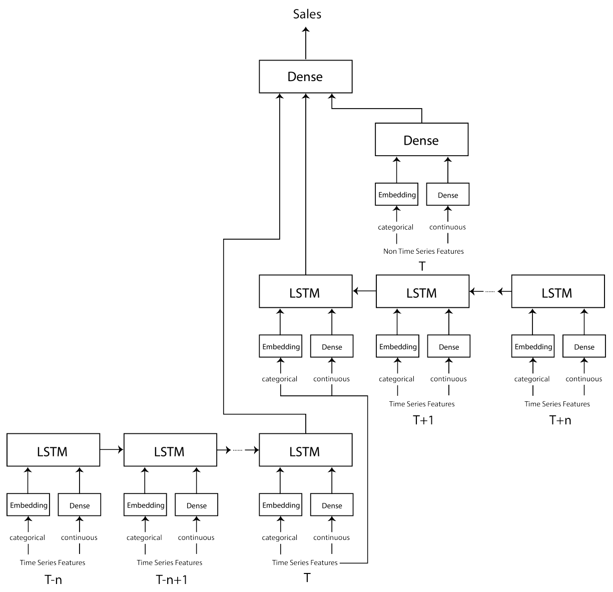

Figure 2: The structure of Rossmann LSTM neural network

3.4. Informer

To further improve the prediction accuracy, we observe the recently introduced state-of-the-art method on time series forecasting named Informer.

Informer is a new transformer based model for long sequence time series forecasting[2]. It leverages the self-attention Transformer architecture to extend predictions to much farther temporal horizons compare to the other time series forecasting method.

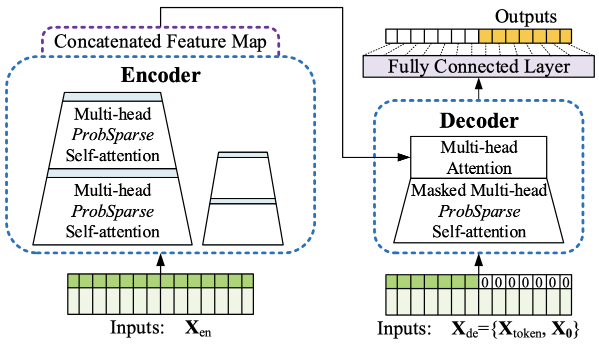

Figure 3: Architecture of Informer

Informer aims to improve the self attention mechanism, reduce memory usage, and accelerate inference speed. It utilizes both a transformer encoder and transformer decoder layers. The decoder can effectively forecast long sequences in a single forward pass. This feature helps accelerate inference speed when forecasting long sequences. The Informer model employs a probabilistic attention mechanism to forecast long sequence. It also includes learned embedding of relevant temporal features. This allows the model to generate an effective task based representation of time. Finally, the Informer likewise can stack n levels of encoder and decoders based on the complexity of the task.

3.5. Measure and compare their performance

After we each train the model, we submit it to Kaggle and get the result of the model **Figure : III** . Then we ensemble the outcome of each base model and results in the final prediction. Our final prediction with RMSPE of **0.11073** ranked 373 on the Kaggle LeaderBoard.

Model	RMSPE of test
XGBoost	0.12953
LSTM	0.12645
Informer	0.11981
Ensemble	0.11073

Table III: Model performance on Kaggle test dataset

3.6. Conclusion

The most interesting part is the XGBoost importance. It reveals that in the long run, the promotion has no effect on sales quality, which is contrary to the common knowledge of the economics of most people. But this can be explained in some respects. The service and quality of the store are always the most important determinants of sales. Another point worth mentioning is that we plan to add external features like *State* and *Weather* at the beginning. Then due to too much data and too noisy, the LSTM training speed is very slow and does not help the improvement of the model. We abandon this method.
Time series forecasting is one of the most widely used data science techniques in business. However, processing and training time-series data requires more skills to ensure time dependence. Generally speaking, the model we trained has a relatively good performance in predicting time series data.

References

[1] C. Guo and F. Berkhahn, “Entity embeddings of categorical variables,” *arXiv preprint arXiv:1604.06737*, 2016.

[2] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, “Informer: Beyond efficient transformer for long sequence time-series forecasting,” *arXiv preprint arXiv:2012.07436*, 2020.

4. Contribution Statement

Due to some unreasonable parts in the division of labor, we redistributed the contribution ratio. The contribution assigned are as follow and we finally reach an agreement on it.

- Shuai Jiang (sj4n20@soton.ac.uk) : 10%
- Shengtao Zhong (sz3g20@soton.ac.uk) : 30%
- RunYing Jiang (rj1u20@soton.ac.uk) : 30%
- Liqi Gu (lglu20@soton.ac.uk) : 30%

This is the contribution ratio for all of the work (including the last report). In the first report, we did not mention the issue of task assignment because the tasks has not all be completed.