Homework 1: M5 Forecasting - Accuracy (Kaggle Competition)

Group 7
LSDA_WKS

Shahriyar Mahdi Robbani
Faculty of SCIENCE
University of Copenhagen
(XQR418)

Stefano Pellegrini
Faculty of SCIENCE
University of Copenhagen
(MLQ211)

Jean-Baptiste Van Den Broucke
Faculty of SCIENCE
University of Copenhagen
(DSH872)

Mads Porse Pedersen
Faculty of SCIENCE
University of Copenhagen
(XKS203)

I. Introduction

Time series forecasting is a very important topic within a plethora of fields; from stock trading, product sales in a store and the change of the price of fuel over time, to birth rates, unemployment rates and changes in population sizes. How time series data develops over time can be split into 4 components: *Level* (baseline value of the series, so essentially a straight line), *Trend* (increasing or decreasing behavior over time), *Seasonality* (repeating patterns in the data over time), and *Noise* (unexplained variability in the data), of which Trend and Seasonality are not always observed. [3]

The purpose of this report is to describe our approach to the Kaggle "M5 Forecasting - Accuracy" challenge. The competition involves hierarchical sales data from Walmart (spanning 1913 days into the past) with the goal of predicting the next 28 days of sales. The competition uses the Weighted Root Mean Squared Scaled Error (RMSSE) to evaluate the results.

Lastly, it should be mentioned that when we initially submitted a score to the Kaggle competition on May 31^{st} - we were advised to do so by the course instructors, because the final data (last 28 days of sales) would be made available shortly after that date. This score was based on a notebook that we originally intended to improve upon, but, later on, we ended up using another notebook as a starting point. Thus, our highest score on the Kaggle public leaderboard should therefore not be taken into account when evaluating our submission for Homework 1. We thought we would be able to select among all our Kaggle submissions which scores we wanted to keep, however that was not the case, as we found out later on.
In short: Please disregard our highest public score on Kaggle, and consider as our highest the score of our final model reported in the Results section.

II. Background and Data

The data for the Kaggle challenge was composed of 5 datasets: a `calendar.csv` file containing dates information about product sales, a `sales_train_validation.csv` file containing the historical daily sales per product and store, a `sell_prices.csv` file containing the products prices per date and store, and a `sample_submission.csv` file showing the correct format for the submission.

As starting point for approaching the Kaggle competition, we used the "M5 First Public Notebook Under 0.50" public notebook by kkiller [2]. This model uses a tree based approach, namely Light Gradient Boosting Machine (lgbm) [1], to predict time series data.

During the preprocessing step, the notebook combines prices, sales and dates information into a unique dataset. Then, it performs features engineering to add, in each row, sales information from previous rows. The main idea behind such approach is to use historic sales data to predict future sales. In order to predict the sales for day x, the starting model used sales data with a lag of 7 (sales from day $x-7$) and 28 (sales from day $x-28$). Sales throughout the week can fluctuate greatly, so in order to stabilize the sales value, this method also used a rolling mean of 7 days and 28 days for each lag value used. Therefore, the features created to predict the sales for day x were sales on day $x-7$ (*lag_7*), the stabilized sales of day $x-7$ using a window of 7 days (*rmean_7_7*), the stabilized sales of day $x-7$ using a window of 28 days (*rmean_7_28*), sales on day $x-28$ (*lag_28*), sales on day $x-28$ using a window of 7 (*rmean_28_7*) and sales on day $x-28$ using a window of 28 (*rmean_28_28*).

The other main approach used by this method is to multiply each prediction by a "magic number" in order to improve the Kaggle public score. The justification for this increase is that there is an upward trend in sales throughout the years, and a tree based model will never be able to predict sales larger than the training data, therefore this magic number is used to model the upward trend of sales figures.

It is also worth mentioning that this model uses 2,000,000 randomly selected data points to generate a validation dataset and monitor the training process. A seed value was chosen to fix which data points were used in the validation dataset in order to make the comparison between models consistent. We used a seed value of 42.

III. Approach

A. Magic Numbers

We believe the "magic numbers" used by kkiller is bad practice. While the justification of using it to model an upward trend does have some merit, multiplying each prediction with an arbitrary number will only lead to overfitting. In order to verify if this is truly the case, we extracted the last 28 days from the training data (day 1886 to day 1913) and attempted to predict the sales for these 28 days with and without the magic numbers multiplication. If the magic numbers approach is generalizable, it should perform well on this dataset ("custom" test dataset) as well. We used the standard RMSE measure

(from the Scikit-learn package) to determine how well the predictions fit the true values.

On this basis, we tested the performance of all our new models by using both, the RMSE on our "custom" test dataset, and the Kaggle RMSSE evaluating the prediction of the next 28 days (day 1914 to day 1941).

B. Lag features

The basis of a tree based approach is that we will use features of a day x to predict its sales value y, and performance of this model will depend on how related these features are to the sales value y. Normally time series data for a day x depends on previous days and this can cause seasonality. We believe patterns in sales figures are caused by weekly, monthly, and yearly patterns. In order to model this, we use the sales from 7 days ago, 28 days ago and 364 days ago. The values 28 and 364 were chosen to preserve the structure of the week as some items may only be sold on certain days of the week. Additionally, we also used a 364 day rolling mean window along to get an average of sales value of that item for the whole year, as this can be used to determine the overall sales trend for the item. Therefore, the novel lag features added to the model were sales on day $x-364$ (*lag_364*), the stabilized sales of day $x-364$ using a window of 7 days (*rmean_364_7*), and a window of 28 days (*rmean_364_28*). Also, the annual sales average starting from day $x-7$ (*rmean_7_364*), day $x-28$ (*rmean_28_364*), and day $x-364$ (*rmean_364_364*).

The original model was unable to use all the training data due to RAM constraints, therefore the addition of the 364 lag and window features allowed us to use the entire dataset by "remembering" sales values from 1 year ago.

Related to the RAM issue, it is worth mentioning that for all tests performed to improve the models performance, we used standard free Google Colab accounts with limited computational resources.

C. Parameter Tuning

As for all other changes we applied to our models, for the parameter tuning we changed only one parameter at a time, to make sure that we would be able to tell which parameter change would have a positive effect on the accuracy of the model. Also, it should be mentioned that for all runs of training, the models were trained for 1200 iterations.

In Table 1 are listed all the parameters contained in the model from the original notebook.

Parameter	Value	Parameter	Value
objective	"poisson"	lambda_l2	0.1
metric	"rmse"	verbosity	1
force_row_wise	True	num_iterations	1200
learning_rate	0.075	num_leaves	128
sub_row	0.75	min_data_in_leaf	100
bagging_freq	1		

TABLE I
ORIGINAL MODEL PARAMETERS.

As mentioned, the parameters (and their corresponding values) above were all passed to the LightGBM method in the original notebook. Thus, looking at these parameters, as well as reading up on the LightGBM documentation [1], we got inspired to test out different parameter values to see if it would be possible to improve the accuracy of the model that way. We started testing new parameter values after obtaining the most optimized model with the addition of lag features.

The changes in the parameters values that improved the model performance and led to our final model were the reduction of the number of leaves (*num_leaves*) from 128 to 112, and the reduction of the learning rate (*learning_rate*) from 0.075 to 0.065. These two parameters are considered as "core" ones for this algorithm, and must be tuned carefully (https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html). Since LightGBM can easily overfit, it is important to set an appropriate number of leaves. Reducing it decreases the model complexity and this risk, at the expense of a good accuracy. We found out that 112 was an appropriate trade-off. Moreover, the decrease of learning rate appeared to yield the best increase performance-wise. Also, since we work with a relatively large number of iterations (1200, 100 is default), it is important to select a small learning rate to get a good accuracy. [1]

D. Round up

As final method to improve our Kaggle score, we rounded up all predicted sales whose fractional part was above a certain threshold (.65). This approach follows the logic that a predicted sales of 1.65, for example, is more likely to be 2 than 1. Also, we decided to only round up, and not to round down, by following the original notebook's author idea, that a tree based model can not predict sales larger than the training data. So, that it can not model the upward trend in sales throughout the years. This method, as the "magic numbers" multiplication used by the author, can lead to overfitting. But, we believe that, while our method is justified by a logical nature, the "magic number" multiplication is not. The threshold of 0.65 was chosen by iterating through the possible threshold supported from our logical reasoning (from 0.5 to 0.9), and we simply chose the threshold that obtained the best result.

As for all our other methods, the "round up" method was tested by using both, Kaggle's RMSSE on the original target, and the RMSE on our "custom" test dataset.

IV. RESULTS

Table 2. shows the novel features and performance of the models that led to an improvement in the final score, except for the *Baseline* model, and its *magic* feature, which relate to the original notebook that we used as a baseline. RMSE refers to the loss obtained on our "custom" test dataset (day 1886 to day 1913) using RMSE as metric, while RMSSE refers to the Kaggle score obtained on the prediction of the next 28 days (day 1914 to day 1941). *Magic* refers to the "magic numbers" multiplication, *Lag364* refers to the novel lag features mentioned in subsection B of Background and Data, *Leaves* refers to the reduction of the tree leaves number from 128 to 112, *Rate* refers to the reduction of the model

learning rate from 0.065 to 0.055, and *Round* refers to the "round up" method mentioned in the previous section.

Model	Magic	Lag364	Leaves	Rate	Round	RMSE	RMSSE
Baseline	X					2.1033	0.4985
Model1						2.0875	0.5564
Model2		X				2.0651	0.5473
Model3		X	X			2.0591	0.5473
Model4		X		X		2.0573	0.5473
Model5		X	X	X		2.0568	0.5398
Model6		X	X	X	X	2.0605	0.5046

TABLE II
PERFORMANCE OF THE TESTED MODELS. THE COLUMNS CORRESPOND TO THE DIFFERENT PARAMETERS TESTED, WITH "MAGIC" AS THE USE OF "MAGIC NUMBERS", "LAG364" AS A YEARLY LAG IMPLEMENTATION, "LEAVES" AS A NUMBER OF LEAVES OF 112, "RATE" AS A LEARNING RATE OF 0.065 AND "ROUND" AS A ROUNDING OF THE PREDICTED VALUES.

The *Baseline* model, shown in Table 2, is the original model from the kkiller notebook. It has an RMSSE score of 0.4985, however this is larger than his stated score of 0.4889 because we used a different seed as mentioned in the Background section. While this seed value does affect the score, since we used the same seed for all the models we tested, all our models are comparable. While his RMSSE score is great, it doesn't perform equally well on our "custom" test set with an RMSE score of only 2.1033. As mentioned in the Approach section, we believe that the use of the magic number multiplier was overfitting the Kaggle validation dataset which led to such a low score.

For clarity, from now on, we will only mention RMSE and RMSSE scores, omitting the datasets and assuming that the former is the metric used to evaluate the performance on our "custom" test dataset (day 1886 to day 1913), and the latter is the metric used to evaluate the predictions of the next 28 days (day 1914 to day 1941), on the Kaggle leaderboard.

Model1 is simply the same model without the "magic number" multiplier. The RMSE decreased to 2.0875, while the RMSSE increased to 0.5564 compared to the baseline model scores. This result agrees with our hypothesis since if the magic number multipler was indeed generalizable, it would result in a worse RMSE. While we do acknowledge that our RMSE is not directly comparable to the RMSSE used by the Kaggle evaluation, it still an objective measure of the model performance and can still be used to make a comparison.

Model2 is the model obtained by adding the lag features mentioned in Lag features subsection of Background and Data. By adding information of the yearly sales patterns, the model performance was improved on both datasets, obtaining a RMSE of 2.0651 and a RMSSE of 0.5473.

Model3 to *Model5*, are the models obtained by parameter tuning. It is interesting to observe that on their own, neither the reduction of the number of leaves (*Model3*) nor the reduction of the learning rate (*Model4*) results in a improved RMSSE, although they did increase the RMSE score. When both these parameters are changed together, only then do they decrease the RMSSE score to 0.5398.

Model6 is our final model, with the yearly lag features, the tuned parameters, and the "round up" method mentioned in the Approach section. This method decreased the RMSSE value to 0.5046, while the RMSE was increased to 2.0605. It can be observed that the addition of the "round up" method increased the RMSE only slightly when compared to the "magic number" multiplier in the *Baseline* model.

In fact, Table 2 shows that the addition of the "magic numbers" (*Model1* to *Baseline*) increased the RMSE ($+0.0158$) 4.2 times more than the addition of the "round up" method in *Model6* ($+0.0037$). So, the RMSE and RMSSE scores in Table 2, show that our final model (*Model6*) performs well in both datasets, evaluated with the two different metrics. Therefore we believe that it has a greater generalization ability than the model we used as starting point (*Baseline*), and, so, that it can better adapt to new, previously unseen data.

While it may appear that our final model is not an improvement over kkiller's model, this is only due to the magic number multiplier, which we believe is bad practice. In order to show that our added features are indeed beneficial to model performance, we used them in combination with the magic number and the round up method (0.8 threshold) to obtained an RMSSE of 0.4935 and an RMSE of 2.0672, both of which were improvements with respect to the baseline model.

Figure 1 shows the feature importance of our final model. We can see that the most important feature is, as expected, the item ID. The most important novel feature we added to the model is the sales value one year (364 days) in the past, stabilized over a window of one week (*rmean_364_7*), followed by the the sales one year in the past, stabilized over a window of 4 weeks (*rmean_364_28*). This is followed by the average sales of the previous year, starting from the past 7 days (*rmean_7_364*), the sales one year in the past (*lag_364*) and the average sales of the previous year, starting from one year in the past (*rmean_364_364*), which is placed just before the sell price feature. The least important novel feature is the average sales of the previous year, starting from 4 weeks in the past (rmean_28_364), which is placed between the department ID and the year data feature.

V. CONCLUSIONS

In order to improve kkiller's model, our approach involved engineering lag and window features, parameter tuning of the lgbm model and rounding up the predicted sales values beyond a certain threshold. We removed his magic number multiplier because we believe it leads to overfitting and consider it bad practice. Although our final model has a larger RMSSE score than kkiller's model, this is only due to the magic number multiplier. In order to show this, we combine the magic number multiplier with our improvements, which resulted in a improvement in both scores for both datasets.

It is important to mention that neither the RMSE score on our custom dataset, nor the RMSSE score is an unbiased estimation of model predictive ability since we used both datasets to evaluate each change to our model. Therefore the

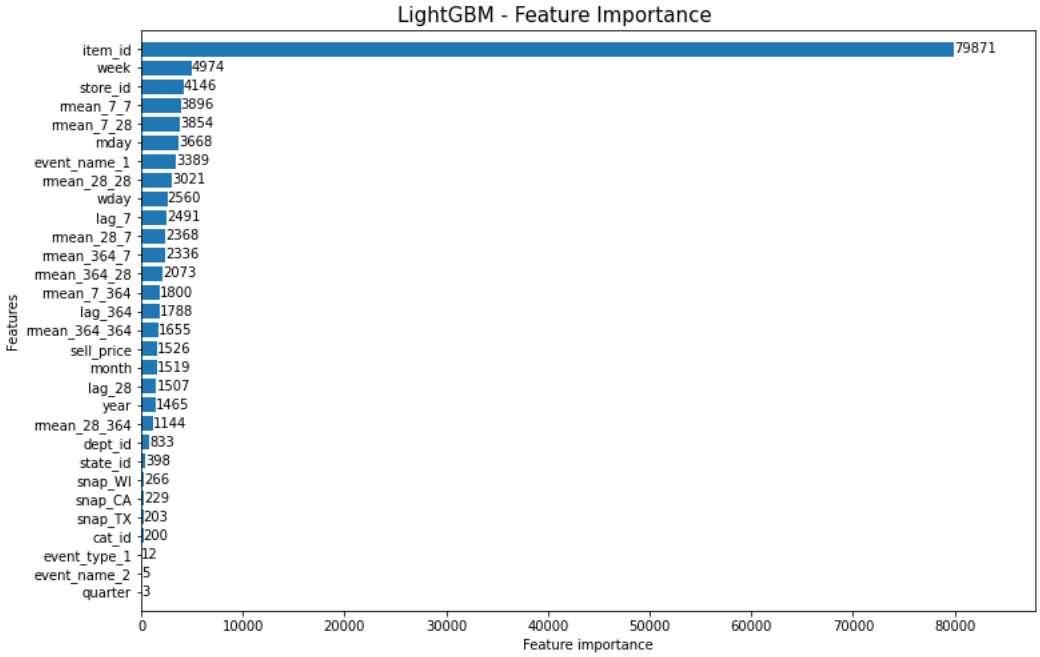

Fig. 1. LightGBM feature importance of the best model (update with the one obtained after reducing leaves number).

true performance of the model can only be determined once the final private leaderboard scores are revealed.

We appended the file `csv_and_notebooks.zip` containining three jupyter notebooks and two submission.csv files:

1) `customtest_RMSE_model6.ipynb`, which contains the code to produce the custom dataset and to calculate the RMSE score obtained by our final model (*Model6*).
2) `kaggle_RMSSE_model6.ipynb`, which contains the code for our final model (*Model6*) and produces the csv file `sub_v_kaggle_RMSSE_model6.csv` which was tested on Kaggle. This notebook and csv file are our main results.
3) `kaggle_RMSSE_magic_model6.ipynb`, which contains the code for our final model with the addition of the magic number and produces the csv file `sub_v_kaggle_RMSSE_magic_model6.csv` which was tested on Kaggle.

VI. Individual Contributions

All members of the team contributed equally.

References

[1] M. Corporation. Lightgbm's documentation. https://lightgbm.readthedocs.io/en/latest/index.html, 2020.
[2] kkiller. M5 first public notebook under 0.50. https://www.kaggle.com/kneroma/m5-first-public-notebook-under-0-50, 2020.
[3] A. V. Metcalfe and P. S. Cowpertwait. *Introductory time series with R*. Springer, 2009.