Project Report

Title: Seeloz C2A Modeling Team

**Abstract:**

The goal of this project is to predict the probability of sales for different products given the data of past sales. We designed an LSTM RNN for this purpose that takes as input sales data for the past 7 weeks and predict the sales for the 8th week. Some of the challenges faced in this project were mostly with the feature engineering since most of the data are encoded and normalized. After careful feature engineering and model selections, we arrived at a model that gives a satisfactory performance.

**DL Problem Specification**

Seeloz provided the raw data and our task was to develop a model to generate predictions for sales. Most of the data were either normalized or hashed, so little insight could be gained from the raw data itself. The final dataset created after feature engineering is a matrix of weekly orders, that contain the aggregated order information for each product per warehouse. The data set has the following features:

|  |  |
| --- | --- |
| **Feature** | **Description** |
| Order\_date | Week date |
| Product\_id | The product |
| Qty\_ordered | Total quantity of this product ordered in this week |
| Qty\_delivered | Total quantity delivered |
| Delivery\_time | Average time it takes to deliver the order |
| Processing\_time | Average time it takes to process the order before shipment |
| Warehouse\_id | List of warehouses in which the orders for this product in this week were placed |
| Site\_id | List of sites at the orders of this week for this product were delivered |
| Invoice\_item\_id | List of invoices for each order of this product in this week |

Table 1: weekly dataset.

This dataset represents the time series of sales across weeks. Our task was then to predict sales for new weeks given this data of past sales. The predictor column is **Qty\_ordered.**

**Design and Milestones**

On Google Cloud platform, we used BigQuery and Cloud storage to pre-process the raw data and generate a CSV file containing the following features:

Product\_id, order\_date, quantity\_ordered, quantity\_delivered, processing\_time, delivery\_time, warehouse\_id, site\_id, quantity\_returned,

Pandas was then used for further pre-processing with Google Colab notebook. This pre-processed data was grouped by weeks and the total quantity of each product sold for a week computed. Because of the unknown normalization strategy used on the quantity of products, we decided not to fill in the missing weeks. At first, we tried filling in the missing weeks with 0 values but since 0 falls within the normalization range (-1 to 1) we thought it might affect the ability of our model to learn properly. The new weekly data as described in table 1 above was saved as a CSV file.

This weekly data is the final dataset used to train and evaluate our model. The data was split according to the following ratio: 80% train, 10% validation and 10% test. Because this is a time series data and the task involve predicting future trends, splitting was done without shuffling the data.

To simplify the training and model generation we decided to use only the following features for training.

Product\_id , Qty\_ordered, Delivery\_time and Processing\_time

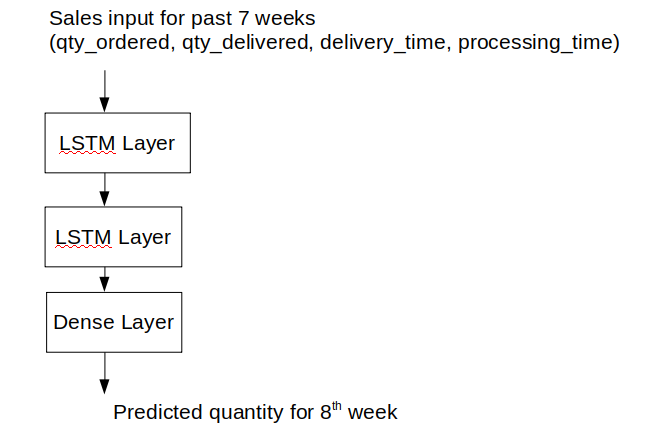
This is because the list of warehouse\_ids and site\_ids were mostly sparse. So we decided to train a general model using all the data and this general model can be used to predict overall sales across all warehouses for a particular product.

Another pre-processing step we did was to generate embeddings for the product\_id. Similar to Stock2Vec, we created an embedding model, product2vec, that transforms product\_ids into a vector space. The objective here was to see whether there are any relationships between products that can contribute to a better learning in the model. Our embedding model takes in a product\_id and generates a vector of size 50 as the embedding for the product.

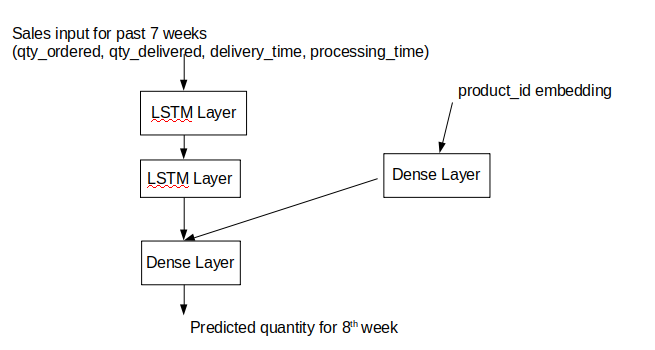
We then decided to build two different LSTM RNN models, one with the product embedding as input and another without the embeddings.

Our model was designed based on the following model (<https://www.kaggle.com/frlemarchand/covid-19-forecasting-with-an-rnn>). The architecture of our models follows

LSTM only model



LSTM + Embedding model



Both models use the data for the past seven weeks to generate predictions for the 8th. The LSTM + embedding model takes the product\_id embedding as an additional input.

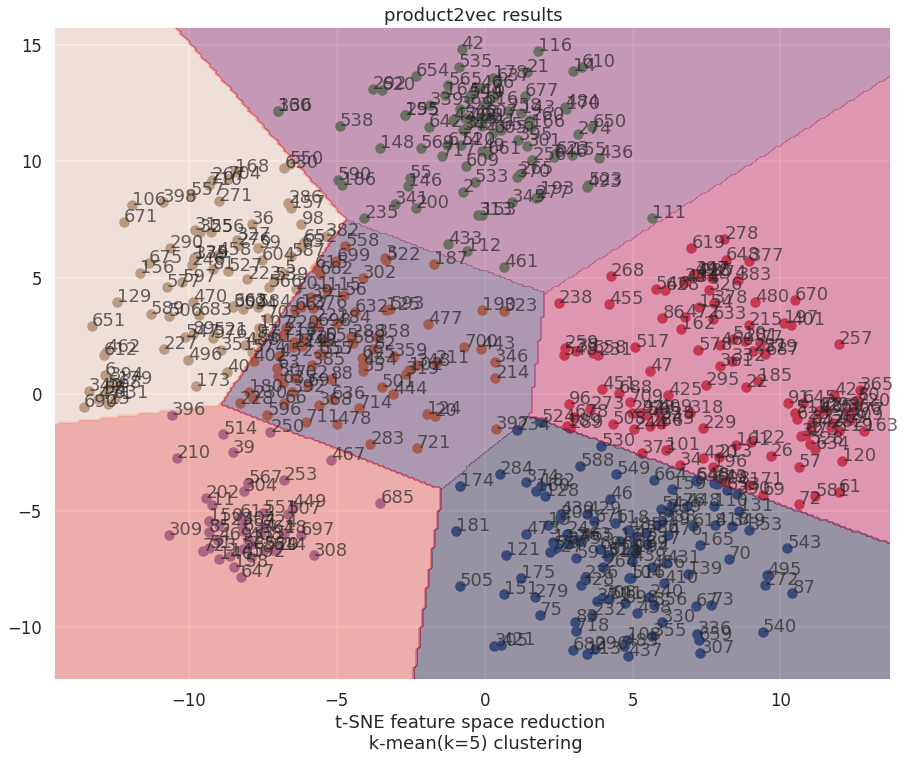
For both models,

The first LSTM layers has 64 units, the second LSTM layer 32 units and the final dense layer has 128 neurons. The output layer is a dense layer with 1 neuron for predicting the expected quantity. The dense layer branch in the LSTM+ model has 50 neurons. Adam optimizer was used for both models and Mean Absolute percentage error use as loss function.

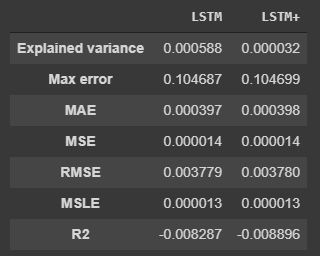
Because we spent most of the time engineering the features for the learning, we did not experiment much with the hyperparameters on our model. We trained each model for 500 epochs with a batch size of 16 and implemented early stopping and the reduce learning rate on plateau strategies.

**Results**

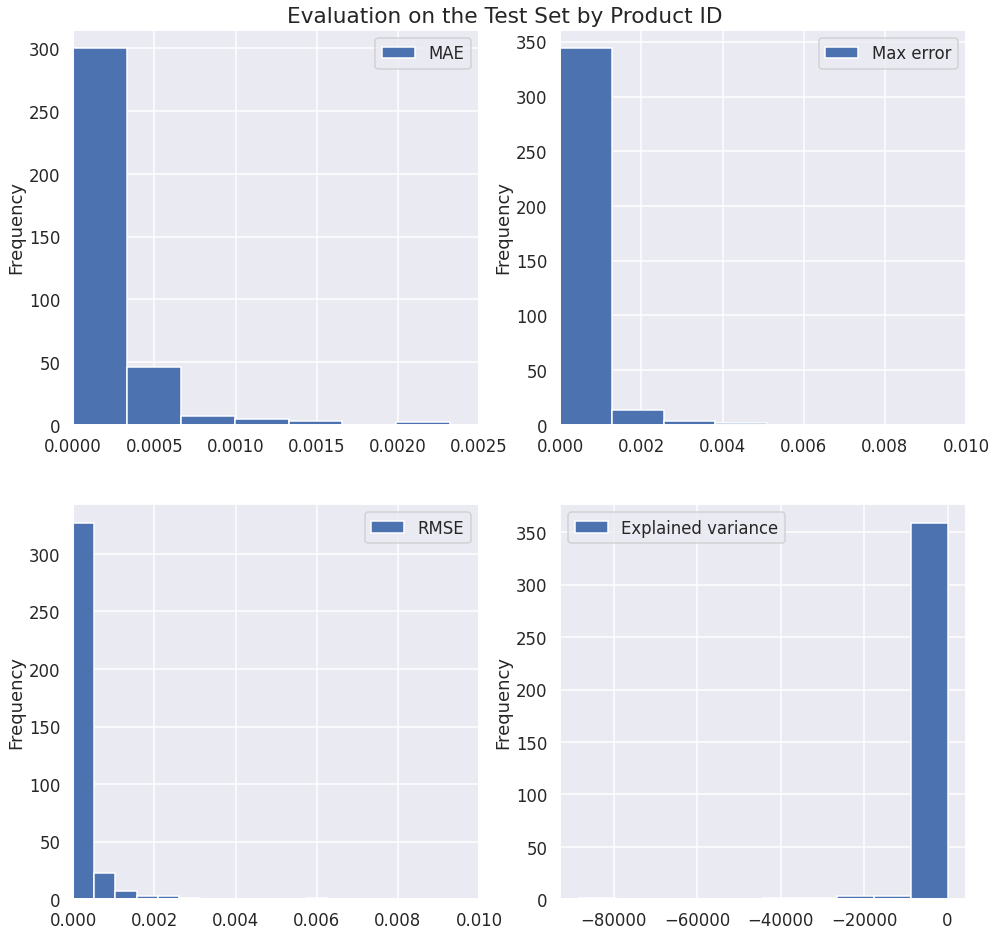
A T-SNE plot is shown in the figure below which is a 2D representation of the products2vec embedding output for 500 of the products in the data set. In addition, we identified 5 clusters with the k-means algorithm and plotted their decision boundaries. The results from product2vec are encouraging because we can see that there are some distinct clusters even in 2D space, indicating that certain products have been identified as related.



For both models, we withheld 10% of the data for testing and evaluated the predictive power of the LSTM and LSTM+ models as shown in the table below. From the results, it is not clear that there is any significant difference in the performance of the two models. Both models converged to approximately 14% error on the validation set which matches the results on the test set, indicating that there was not overfitting.



Next, we calculated the error for each product in the test, because we hypothesized that some products might have better predictions than others. We plot the histogram of the results in the figure below. The figure shows that many of the products have low error rates, but some products have errors that are several orders of magnitude higher than the median.



**Conclusions**

We created a product2vec embedding for the Seeloz products and applied this embedding to LSTM model to predict the weekly total of products ordered based on the past seven weeks of data. This was a good hands-on experience with a real-world dataset. We found that the major challenge was the feature engineering part of the project. As mentioned above we did not know how the data was scaled or encoded and we had to make several assumptions to prepare the data for modelling. Although we were encouraged by the products2vec results our LSTM+ model did not perform significantly better than a standard LSTM model. This may be due to the unknown scaling issue and the error it introduced, specifically preventing our ability to impute missing values. If we had more insight into the way the data was generated, we may be able to further improve the models.