Time series forecasting using deep learning
Final Report – Team Singolo – Latha Airodi
GTID: 903469792

1. Introduction – Motivation

In today's highly competitive retail business environment, it is essential to maintain a balance between meeting consumer demands and the cost of inventory. Carrying a bigger inventory enables customer demand to be met, but at the cost of over-stocking of parts and products, causing capital to be tied up and also may result in lower profits. On the other hand, even though lower inventory may decrease inventory expenses, it may result in missed selling opportunities, customer dissatisfaction and other issues. Knowing how much to make and sell, as well as when to sell is key to helping a company increase profits and reduce costs. This information is especially important in manufacturing scheduling to help keep factory costs low by knowing when to increase or decrease temporary labor and other factory expenses in order to control factory costs.

Demand forecasting aids strategic production planning in an industry as it allows managers to anticipate the future and plan joint activities with the functional areas (Veiga, Veiga, Puchalski, Coelho, & Tortato, 2016) [1].Sales forecasting plays an extremely critical role in predicting future sales by examining historical sales for the same time period. It can be as simple as using an excel sheet to track sales and then applying a simple formula to determine future sales or can be as complex as applying machine learning (deep learning) to build ML models based on sales data and then predicting future sales using these models.

2. Problem definition

In this project, we will be analyzing the sales data for a window covering manufacturing company and applying deep learning to predict (or forecast) the sales for a particular product category using the Long Short-term Memory (LSTM)[2] method (a popular recurrent neural network that can learn the order dependence between items in a sequence). We will use Keras v2.0 [3] with the TensorFlow backend to implement LSTM. A benefit of LSTMs in addition to learning long sequences is that they can learn to make a one-shot multi-step forecast which may be useful for time series forecasting.

3. Long Short-Term Memory Networks

Recurrent neural networks are networks with loops in them, allowing information to persist. They can be thought of as multiple copies of the same network, each passing a message to a successor. One of the appeals of RNNs is that they are be able to connect previous information to the present task. For e.g. looking at previous words in a sentence and predicting the next word. In cases where the gap between the relevant information and the place that it's needed is small, RNNs can learn to use the past information. As the gap between current and previous information grows, RNNs are unable to learn to connect the information and will not be as effective.

Long Short Term Memory networks (LSTM) are a special kind of RNN, capable of learning long-term dependencies in sequence prediction problems [4]. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over

arbitrary time intervals and the three gates regulate the flow of information into and out of the cell [5].

4. Data Source

The biggest challenge in this project was identifying and extracting the data required for forecasting. The data was spread across multiple systems and multiple teams. In addition, a majority of the data could not be accessed due to confidentiality. The initial strategy was to forecast based on sales, market promotions, holidays, home sales by region and geographic locations. Unfortunately, access to all of this data could not be provided in time for finishing this project, so the scope of the data was reduced and limited only to total sales figures. The data that was ultimately extracted consisted of the daily sales booked from 2012 through 2020 for one product category.

NOTE: *All sales figures have been hidden from the charts in document to protect confidential information.*

5. Methodology

5.1 Data Exploration

Missing and invalid data

The dataset contained rows which had zero or negative (returns) for the sales, so daily sales was aggregated at the weekly level. The final dataset consisted totals sales by week from 2012 through 2020. A subset of this data (sales from Nov-2017 through Nov 2020) was selected to be used be used for applying the LSTM algorithm and predicting future sales. This subset consisted of 163 rows.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 163 entries, 0 to 162
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   date      163 non-null    datetime64[ns]
 1   salesamt  163 non-null    int64
dtypes: datetime64[ns](1), int64(1)
memory usage: 2.7 KB
```

Figure 5-1 - Data description

Trends and anomalies

A line plot of the data clearly shows an increasing trend in sales for this particular product category. Other interesting observations on this data that would affect the performance of the machine learning model are as follows,

- The three spikes in the data reflect the black-Friday promotions in 2018, 2019 and 2020. This was the only information related to promotions that was available, any other promotion data for other years was not accessible
- The country wide coronavirus shutdown caused the factories to be shut down for two months in 2020. Due to this there is a decrease in the sales booked during this period compared to the same period in the previous two years. This is an extreme anomaly

(hopefully we will never experience this in a long time). This can also be clearly seen in the dip in sales around that time period. The data was **not** adjusted (for e.g. removing this data or replacing it with average sales around the same time period) to accommodate for the factory shut downs

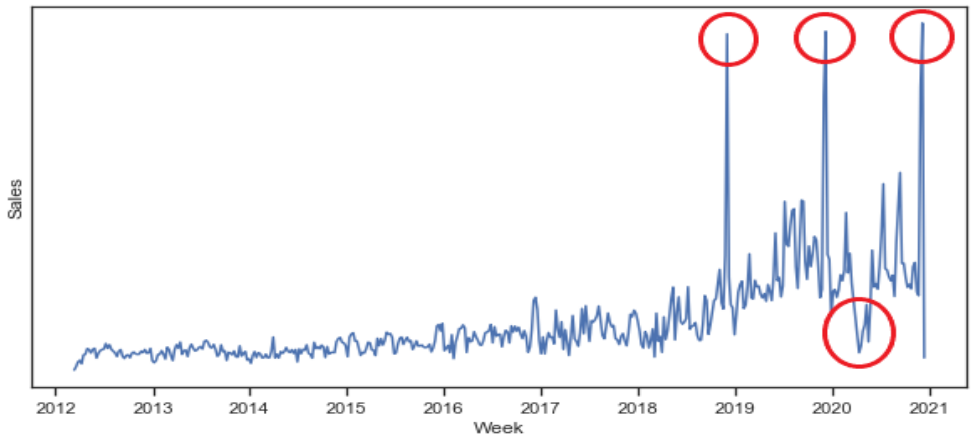

Figure 5-2 - Data trends and anomalies

Data distribution and KDE plot

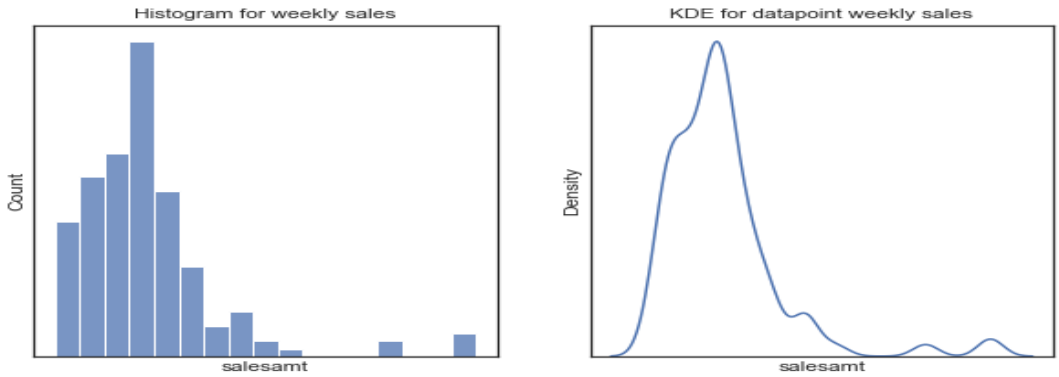

Figure 5-3 - Data histogram & KDE plot

5.2 Data Preparation

5.2.1. Prepare data for multi-step time series forecasting

The first step is to check if the data is stationary and convert it if it is not. A stationary time series [3] is one whose statistical properties such as mean, variance, autocorrelation, etc. are all constant over time. Looking at the weekly sales chart below it is obvious that the data is not stationary. The data has an increasing trend up to around 04/2020, when it drops and then starts increasing again. One method for converting the data is to get the difference in sales compared to the previous month and build the model on it. Figure 5-4 and 5-5 shows the before and after transformations.

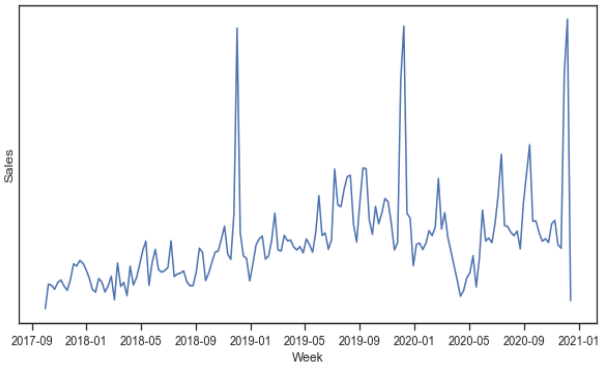

Figure 5-5 – Un-stationary data

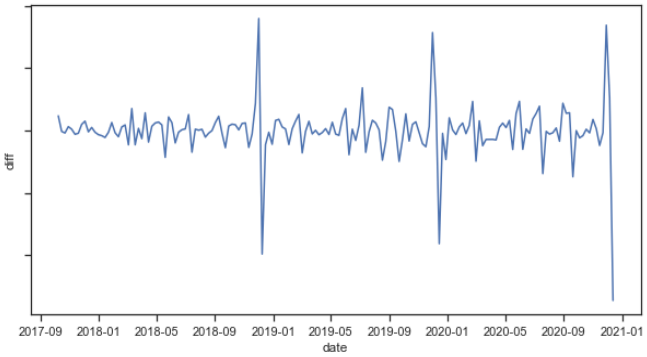

Figure 5-4 - Stationary data

5.2.2. Transform data into a supervised learning problem

Next the data has to be transformed from a series into a supervised learning problem. This involves taking a list of numbers and convert them into a list of input and output patterns so that the algorithm can learn how to predict the output patterns from the input patterns.

5.2.3. Determine lag or lookback period

Next step in data preparation is to determine the lag or the look-back period. We will need to use previous weekly sales data to forecast the data and the look-back period varies based on the model. A lag of 12 weeks was chosen for this dataset (adjusted R-squared of 0.30). This resulted in a dataset with 150 rows and 15 columns

```
# Define the regression formula
model = smf.ols(formula=field_names, data=df_supervised)
# Fit the regression
model_fit = model.fit()
# Extract the adjusted r-squared
regression_adj_rsq = model_fit.rsquared_adj
print (field_names)
print(regression_adj_rsq)
print(df_supervised.info())
```

```
diff ~ lag_1 + lag_2 + lag_3 + lag_4 + lag_5 + lag_6 + lag_7 + lag_8 + lag_9 + lag_10 + lag_11 + lag_12
0.29855752397762225
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 15 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   date      150 non-null    datetime64[ns]
 1   salesamt  150 non-null    int64
 2   diff      150 non-null    float64
 3   lag_1     150 non-null    float64
 4   lag_2     150 non-null    float64
 5   lag_3     150 non-null    float64
 6   lag_4     150 non-null    float64
 7   lag_5     150 non-null    float64
 8   lag_6     150 non-null    float64
 9   lag_7     150 non-null    float64
 10  lag_8     150 non-null    float64
 11  lag_9     150 non-null    float64
 12  lag_10    150 non-null    float64
 13  lag_11    150 non-null    float64
 14  lag_12    150 non-null    float64
dtypes: datetime64[ns](1), float64(13), int64(1)
memory usage: 17.7 KB
None
```

Figure 5-6 - Data with the lag variables

5.2.4. Split and scale the data

The data is then split into training and test data and scaled. The test set consisted of the sales for the last six weeks.

```
print(df_model.shape)
print(train_set.shape)
print(test_set.shape)
print(test_set)
```

```
(150, 13)
(144, 13)
(6, 13)
[[ 1.618300e+04  9.031500e+04 -1.785200e+04  1.161600e+04 -3.990600e+04
  -5.731100e+04  3.530000e+02 -3.693640e+05  1.435520e+05  1.395870e+05
   2.199770e+05 -8.495300e+04  2.165300e+04]
 [-1.185250e+05  1.618300e+04  9.031500e+04 -1.785200e+04  1.161600e+04
  -3.990600e+04 -5.731100e+04  3.530000e+02 -3.693640e+05  1.435520e+05
   1.395870e+05  2.199770e+05 -8.495300e+04]
 [-1.559800e+04 -1.185250e+05  1.618300e+04  9.031500e+04 -1.785200e+04
   1.161600e+04 -3.990600e+04 -5.731100e+04  3.530000e+02 -3.693640e+05
   1.435520e+05  1.395870e+05  2.199770e+05]
 [ 8.455980e+05 -1.559800e+04 -1.185250e+05  1.618300e+04  9.031500e+04
  -1.785200e+04  1.161600e+04 -3.990600e+04 -5.731100e+04  3.530000e+02
  -3.693640e+05  1.435520e+05  1.395870e+05]
 [ 2.622370e+05  8.455980e+05 -1.559800e+04 -1.185250e+05  1.618300e+04
   9.031500e+04 -1.785200e+04  1.161600e+04 -3.990600e+04 -5.731100e+04
   3.530000e+02 -3.693640e+05  1.435520e+05]
 [-1.361005e+06  2.622370e+05  8.455980e+05 -1.559800e+04 -1.185250e+05
   1.618300e+04  9.031500e+04 -1.785200e+04  1.161600e+04 -3.990600e+04
  -5.731100e+04  3.530000e+02 -3.693640e+05]]
```

Figure 5-7 - Split and scaled data

5.3 Model Fitting

Once the data has been prepared, the data will be fit to the LSTM model. The following libraries and the corresponding parameters were used. As it is observed, the model improves itself and reduces the error at each epoch.

```
In [37]: from keras.models import Sequential
         from keras.layers import LSTM
         from keras.layers import Dense

         model = Sequential()
         model.add(LSTM(4, batch_input_shape=(1, X_train.shape[1], X_train.shape[2]), stateful=True))
         model.add(Dense(1))
         model.compile(loss='mean_squared_error', optimizer='adam')
         model.fit(X_train, y_train, epochs=1500, batch_size=1, verbose=1, shuffle=False)
         y_pred = model.predict(X_test,batch_size=1)
```

```
Epoch 1491/1500
144/144 [==============================] - 0s 2ms/step - loss: 0.0012
Epoch 1492/1500
144/144 [==============================] - 0s 3ms/step - loss: 0.0011
Epoch 1493/1500
144/144 [==============================] - 0s 2ms/step - loss: 0.0011
Epoch 1494/1500
144/144 [==============================] - 0s 2ms/step - loss: 0.0011
Epoch 1495/1500
144/144 [==============================] - 0s 3ms/step - loss: 0.0011
Epoch 1496/1500
144/144 [==============================] - 0s 2ms/step - loss: 0.0011
Epoch 1497/1500
144/144 [==============================] - 0s 2ms/step - loss: 0.0011
Epoch 1498/1500
144/144 [==============================] - 0s 2ms/step - loss: 0.0011
Epoch 1499/1500
144/144 [==============================] - 0s 2ms/step - loss: 0.0011
Epoch 1500/1500
144/144 [==============================] - 0s 2ms/step - loss: 0.0011
```

Figure 5-8 - Model fitting & results

6. Results and Conclusion

6.1 Results

Below are results of the prediction from the model (inverse transforming the data for scaling). It appears that the model has an accuracy rate of 76%.

```
In [53]: model.evaluate(X_test, y_test)
         1/1 [==============================] - 0s 1000us/step - loss: 0.7576
Out[53]: 0.7576349377632141
```

Figure 6-2 - Model Accuracy

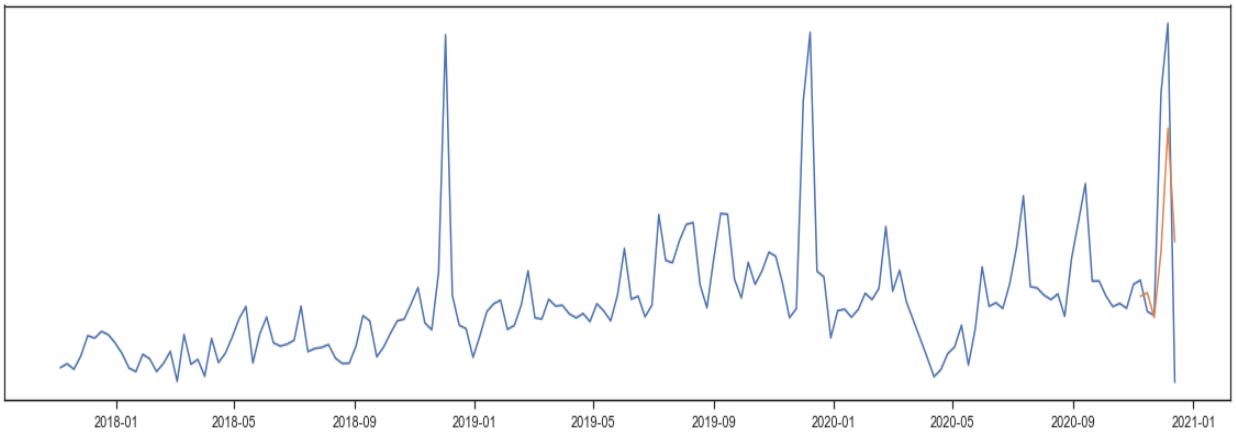

Figure 6-1 - Model prediction vs actual

6.2 Conclusion

A 76% accuracy rate is not the greatest score for any machine learning model. What this tells us is that just the sales amount is not enough to do sales forecasting, but is a good first step towards building an accurate forecasting model. Several additional factors, like promotions, holidays, home sales in a region, geographic locations, etc. will need to be considered and incorporated into the forecasting model to get accurate forecasting data.

The analysis definitely does not stop here, the plan is to continue building the model by exploring additional model parameters and data with the end goal of being able to provide a valuable forecasting tool to the company.

7. References

[1] Agostino, IRS, da Silva, WV, Pereira da Veiga, C, Souza, AM. Forecasting models in the manufacturing processes and operations management: Systematic literature review. Journal of Forecasting. 2020; 39: 1043– 1056. https://doi.org/10.1002/for.2674

[2] Yu Q., Wang K., Strandhagen J.O., Wang Y. (2018) Application of Long Short-Term Memory Neural Network to Sales Forecasting in Retail—A Case Study. In: Wang K., Wang Y., Strandhagen J., Yu T. (eds) Advanced Manufacturing and Automation VII. IWAMA 2017. Lecture Notes in Electrical Engineering, vol 451. Springer, Singapore. https://doi.org/10.1007/978-981-10-5768-7_2

[3] Keras v2.0 - https://keras.io/

[4] Wikipedia - https://en.wikipedia.org/wiki/Long_short-term_memory

[5] LSTM blog - https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[6] Machine learning master https://machinelearningmastery.com/deep-learning-for-time-series-forecasting/