Forecasting_paper

July 9, 2021

1 FORECASTING PROJECT on order transactions

Paolo Cavadini, July 2021.

You can also find the Notebooks in the GitHub repository: https://github.com/pcavad/Forecasting

- ETL: ETL.ipynb
- EDA: EDA.ipynb
- Forecasting: Forecasting.ipynb
- /support/ETL.py
- /support/Helper.py
- /support/DeepLearning.py

1.1 Business understanding - main objective of the analysis

The business context is to support my customer with forecasting. Their business is seasonal with peak periods for Christmas time and events such as Black Friday. However, being a worldwide distribution different peak seasons may apply (e.g. Chinese New Year, etc). Being the brand less than 5 years the sale channels didn't start at the same time which adds a significant randomness to the data.

1.2 Business goal

I ultimately want to predict business and provide a better insight into which factors can affect sales ahead of the time. In a contemporary scenario the sale channllenge is shifting toward a strong interaction with the sourcing channel where brands experience scarsity of components and increased shipping costs.

1.3 Brief description of the data

The dataset includes orders data from one of my customers. Data have been originally dumped from the ERP. The orders file describes transactions between the brand and wholesalers around the world and it was de-normalized to include headers and lines data. It is composed of 9781 rows and it encompasses transactions along 54 months, from July 2016 until June 2021. The fetaures include billing and shipping information as well as line items. I re-organized the data to obtain a time series with totals and sub-stotals for 4 different sale channels.Having in mind to provide reporting and forecasting I developed different assets:

- ETL .py library: routines to fit the raw data for analysis

- Helper .py library: functions which generate pivot tables and plots
- Deep Learning .py library: functions to make a time series and train a simple RNN and a LSTM model
- ETL notebook: calls ETL functions with parameters
- EDA notebook: the place where using the helper library I plot reports and analyse the time series
- Forecasting notebook: the place where I run different time series forecasts

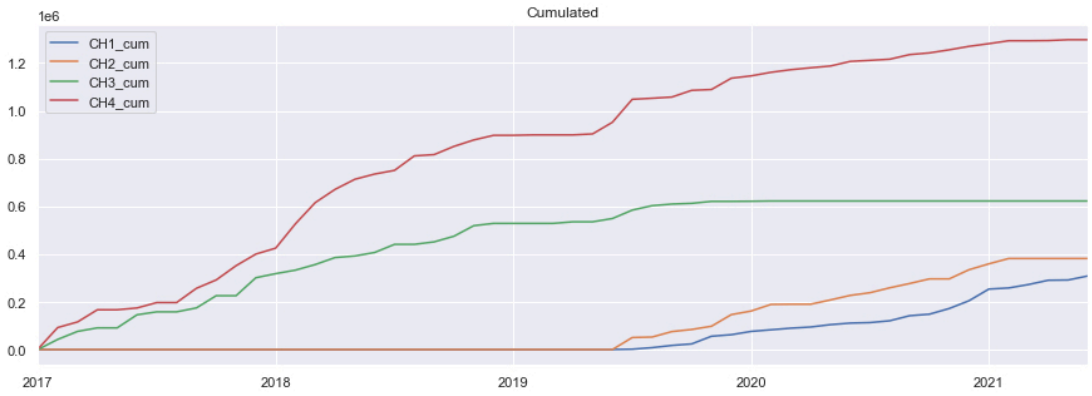
1.4 ETL:

- read and load many raw files in csv format
- filter out (cancelled, etc.)
- drop useless columns
- create new columns for analysis
- creates an order date column datetime64 for resampling
- create an exchange rates file to use only USD
- harmonize data as needed (e.g. wholesaler names)
- save the processed dataframe to a new csv file

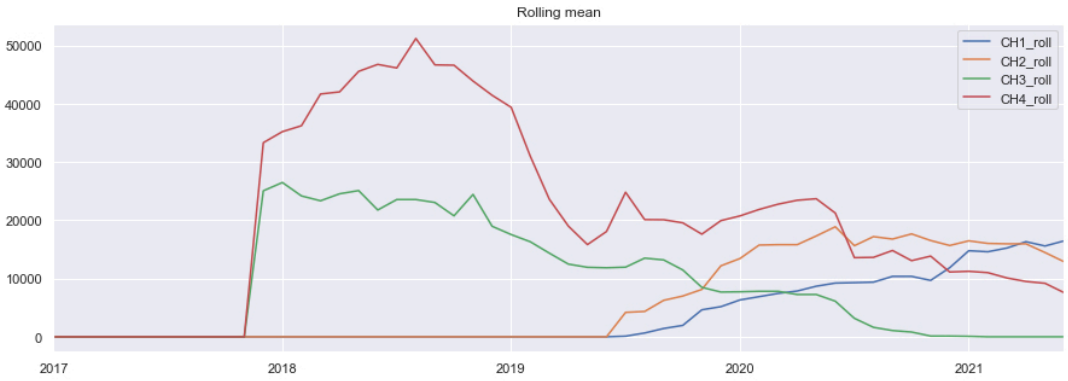
1.5 EDA:

The time series have been originally sampled on a monthly basis to avoid re-sampling and interpolation.

The sale channels didn't start at the same time which became a challenge to obtain a normal distribution:



The rolling mean have been also very different along time:



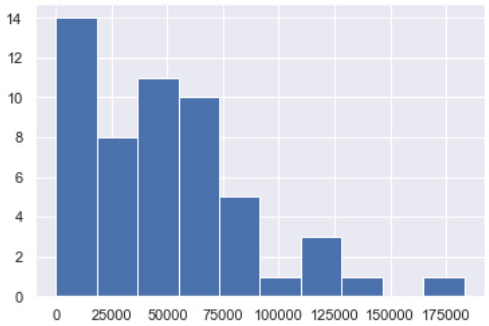
The chunks of the mean and variance values of the total time series show a significant variation:

	mean_vals	var_vals
0	63,813.09	2,854,323,944.82
1	42,679.94	912,891,593.34
2	37,101.22	970,269,765.07
3	89,936.07	700,745,830.06
4	90,396.08	1,177,856,381.27
5	56,690.22	412,861,769.56
6	41,906.68	363,756,995.33
7	52,742.65	306,821,059.68
8	571.79	631,272.78
9	24,478.39	729,339,873.63
10	86,140.70	4,729,699,041.41

The rolling means of the total confirm a trend and the standard deviation changes along time, while the boxplot shows few outliers:

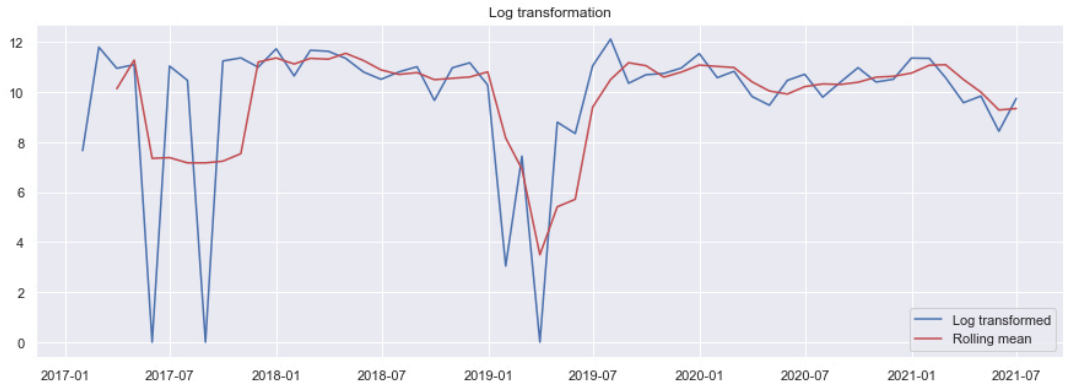
The histogram and normal test show a non-normal distribution, while the Dickey-Fuller test is stationary:

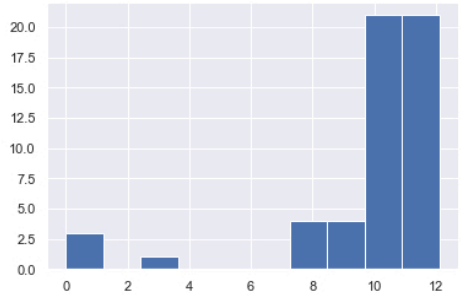
```
NormaltestResult(statistic=14.118261621909, pvalue=0.0008595248589791014)

Dickey-Fuller test:
Index: -5.802923
p-value: 4.586308720283926e-07
Number of observations: 53
Critical values: {'1%': -3.560242358792829, '5%': -2.9178502070837, '10%': -2.5967964150943397}
```



Reduce trend and variance

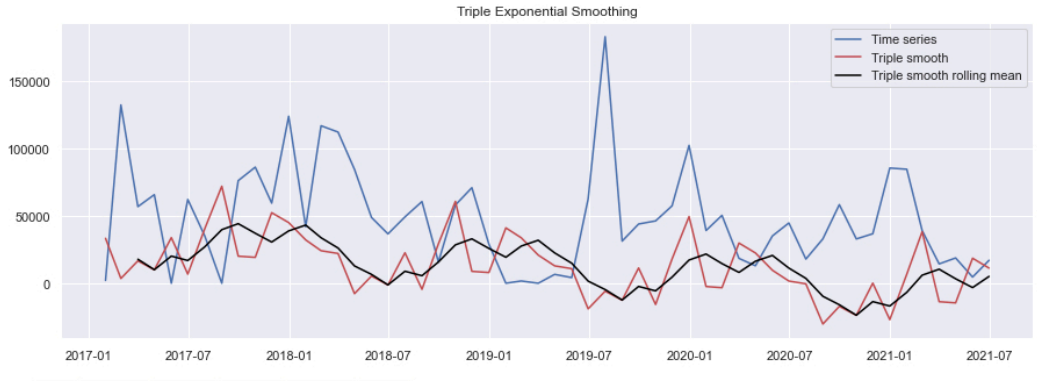
I run a Log transformation to reduce variance: the result wasn't normally distributed nor stationary:

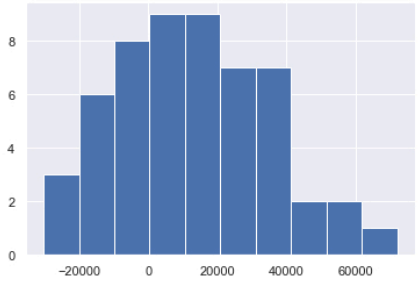




```
NormaltestResult(statistic=49.24243314483155, pvalue=2.028343500881349e-11)
Dickey-Fuller test:
Index: -2.956629
p-value: 0.039148001424211445
Number of observations: 50
Critical values: {'1%': -3.568485864, '5%': -2.92135992, '10%': -2.5986616}
```

I applied Triple Exponential Smoothing: the result was normal but not stationary:

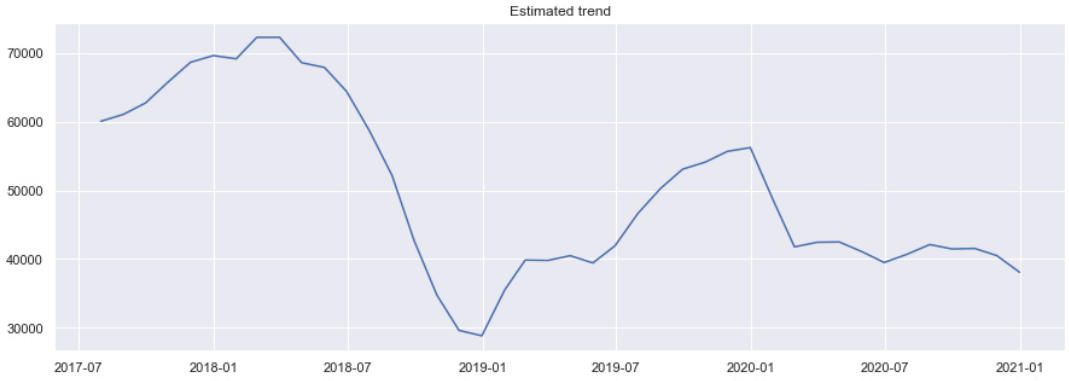


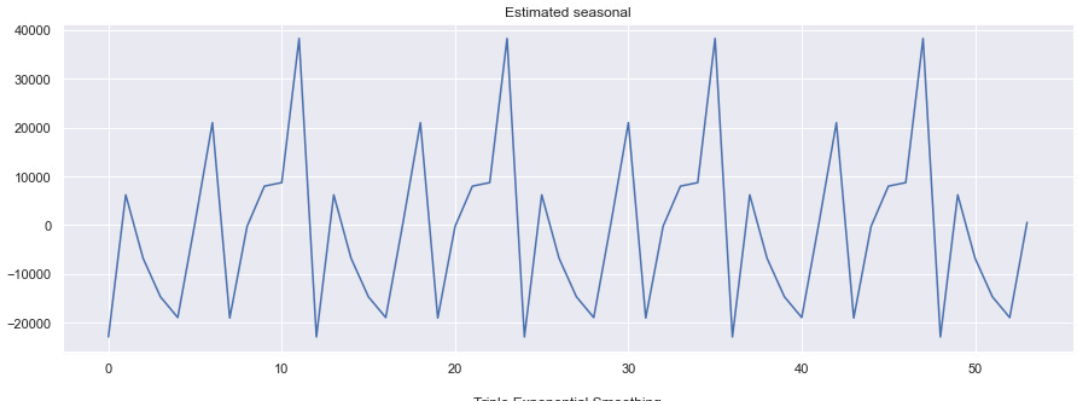


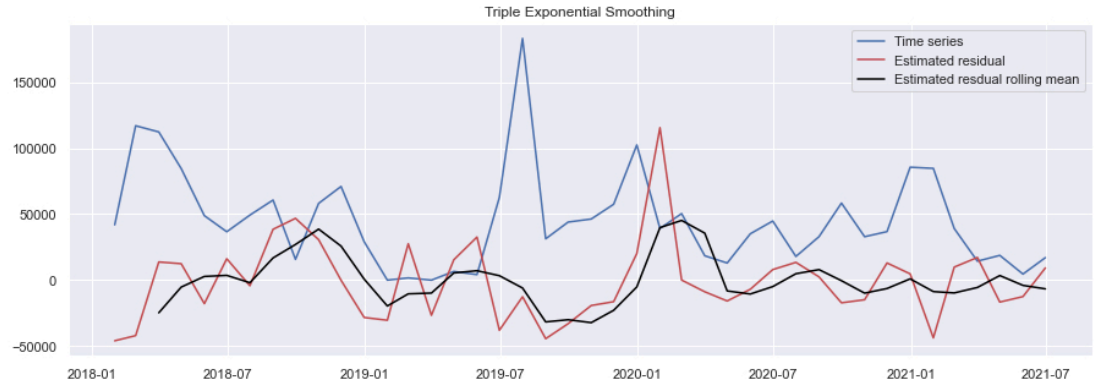

```
NormaltestResult(statistic=1.0162976539369806, pvalue=0.6016082295701024)
Dickey-Fuller test:
Index: -2.238077
p-value: 0.19271704903125286
Number of observations: 51
Critical values: {'1%': -3.5656240522121956, '5%': -2.920142229157715, '10%': -2.598014675124952}
```

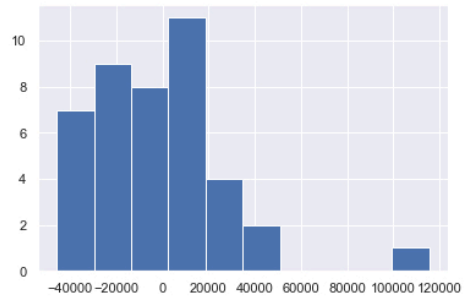
Reduce seasonality

The decomposition of the time series show a trend, seasonality in both 6 and 12 months, and a residual with a spike around July 2019. The residual is not normal but it is stationary:



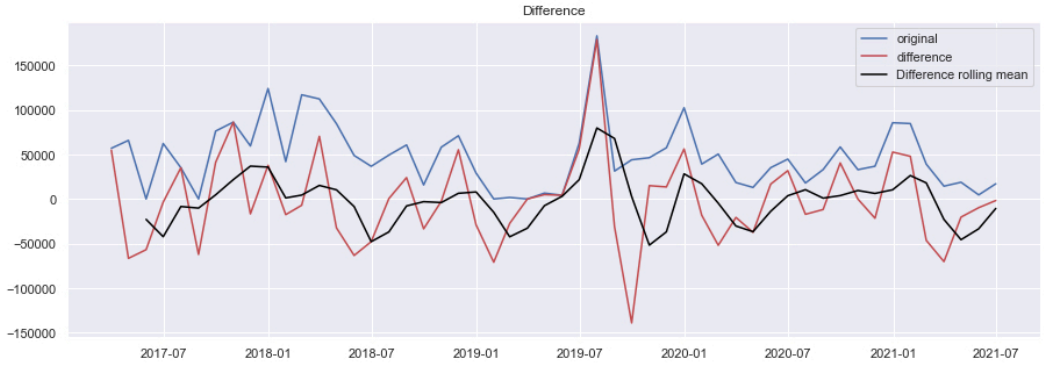


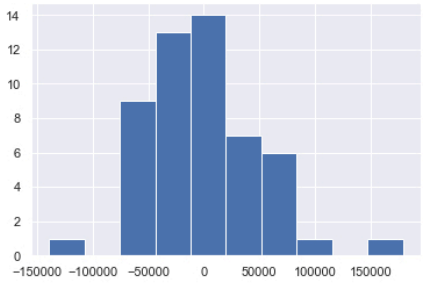





```
NormaltestResult(statistic=19.703088020229323, pvalue=5.266581365742102e-05)
Dickey-Fuller test:
Index: -5.249585
p-value: 6.958079976510259e-06
Number of observations: 41
Critical values: {'1%': -3.60098336718852, '5%': -2.9351348158036012, '10%': -2.6059629803688282}
```

I went to a differentiated time series and found a normal distribution (below the plot with 2 months diff):






```
NormaltestResult(statistic=9.659864617973982, pvalue=0.007987061899968239)
Dickey-Fuller test:
Index: -4.532701
p-value: 0.0001715491616437357
Number of observations: 46
Critical values: {'1%': -3.5812576580093696, '5%': -2.9267849124681518, '10%': -2.6015409829867675}
```

I didn't apply any transformation during the next section in which I run different forecasting models. The original time seris works for the sake of the exercise without adding complexity.

1.6 Key findings

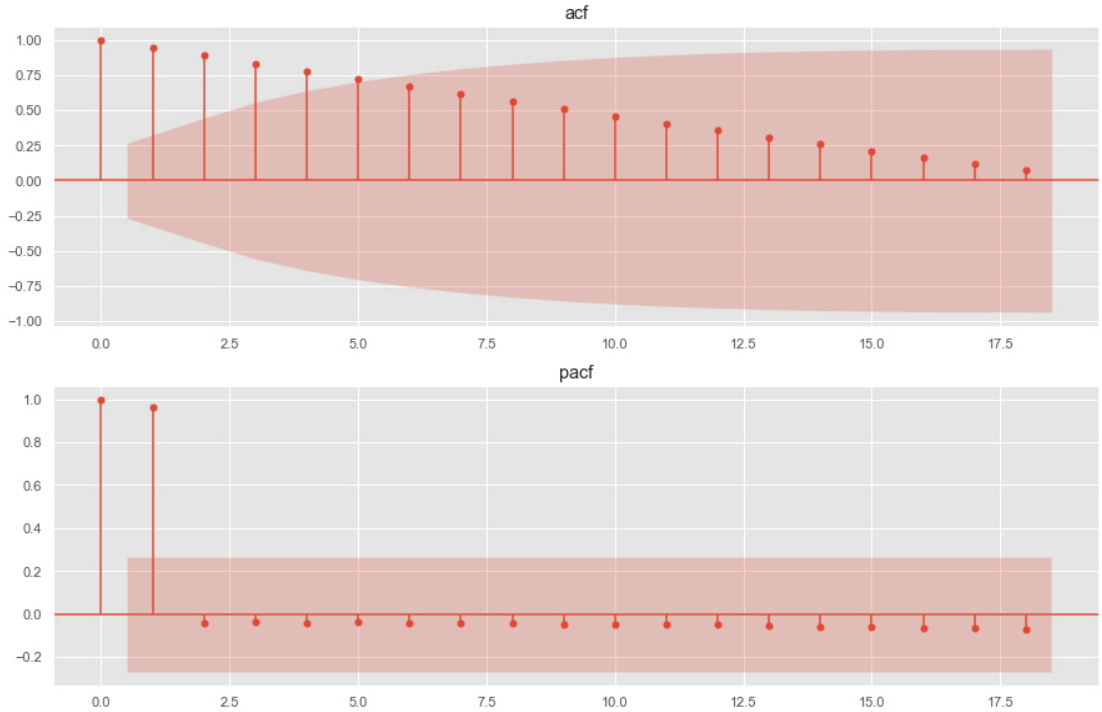
I have implemented 4 different approaches which can deal with both trend and seasonality:

- ARMA
- SARIMA (with different optimizations)
- Deep Learning with RNN
- Deep Learning with LSTM and LSTM stacked

I used the **Mean Squared Error** as a metric to compare the performance.

1st step: ARMA

The acf and pacf plots suggest an autocorrelation model of second order.

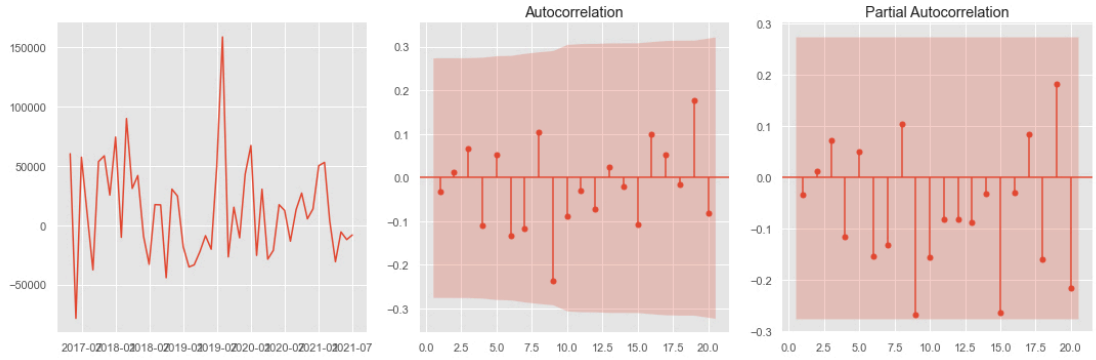


Optimization of the pq parameters: [(2,0), (2,1),(2,2)]

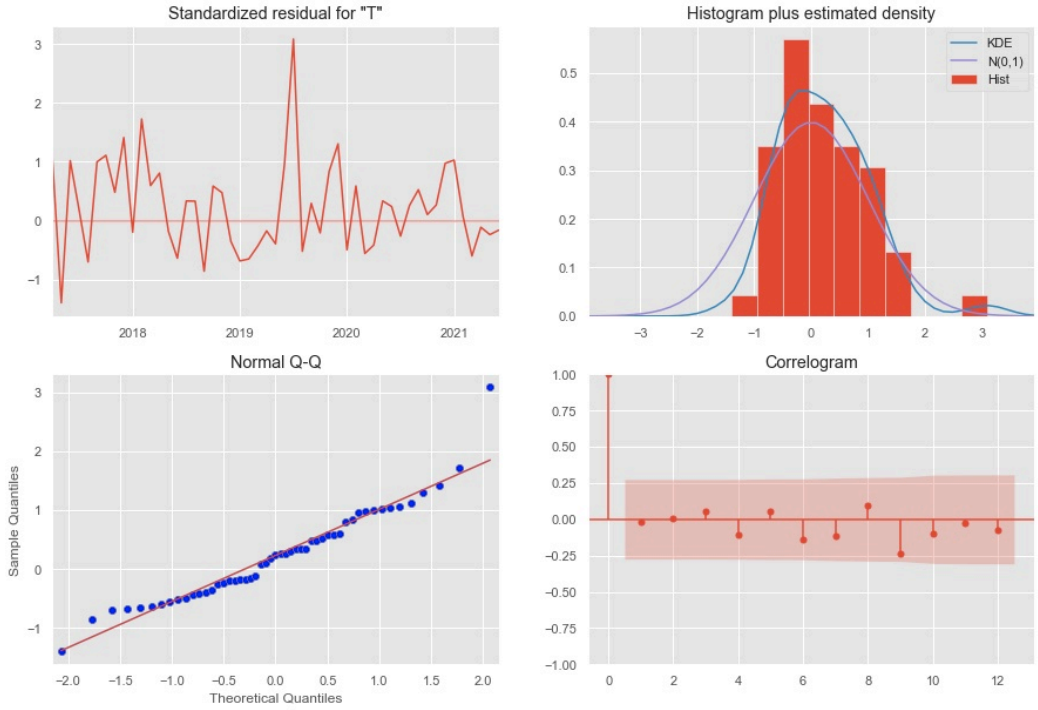
pqs	mse
(2, 0)	1.214048e+09
(2, 1)	6.834230e+08
(2, 2)	2.345307e+09

3nd step: SARIMA

I run the model:



The diagnostic plots show a barely normal behavior:



I run additional SARIMA tests:

- Normality: the Null hypothesis is normally distributed residuals
- Ljung-Box: the Null hypothesis is no serial correlation in residuals
- Heteroskedasticity: the null hypothesis is no heteroskedasticity (tests for change in variance between residuals)
- Durbin-Watson: we want between 1-3, 2 is ideal (tests autocorrelation of residuals)

The results for the residuals have been not ambitious:

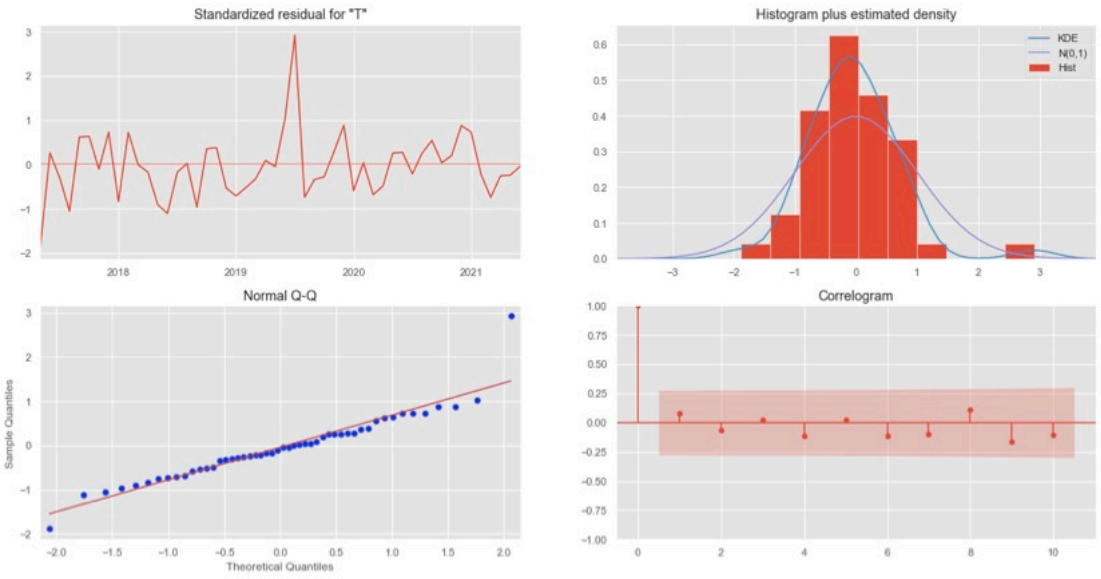
Test	Result	Interpretation
Normality	val=18.104, p=0.000	no normal distribution
Ljung-Box	val=1.951, p=0.924	serial correlation
Heteroskedasticity	val=0.296, p=0.016	heteroskedasticity

Test	Result	Interpretation
Durbin-Watson	d=1.98	no autocorrelation

I run the auto_feat optimization and came up with parameters used for a second round of modeling:

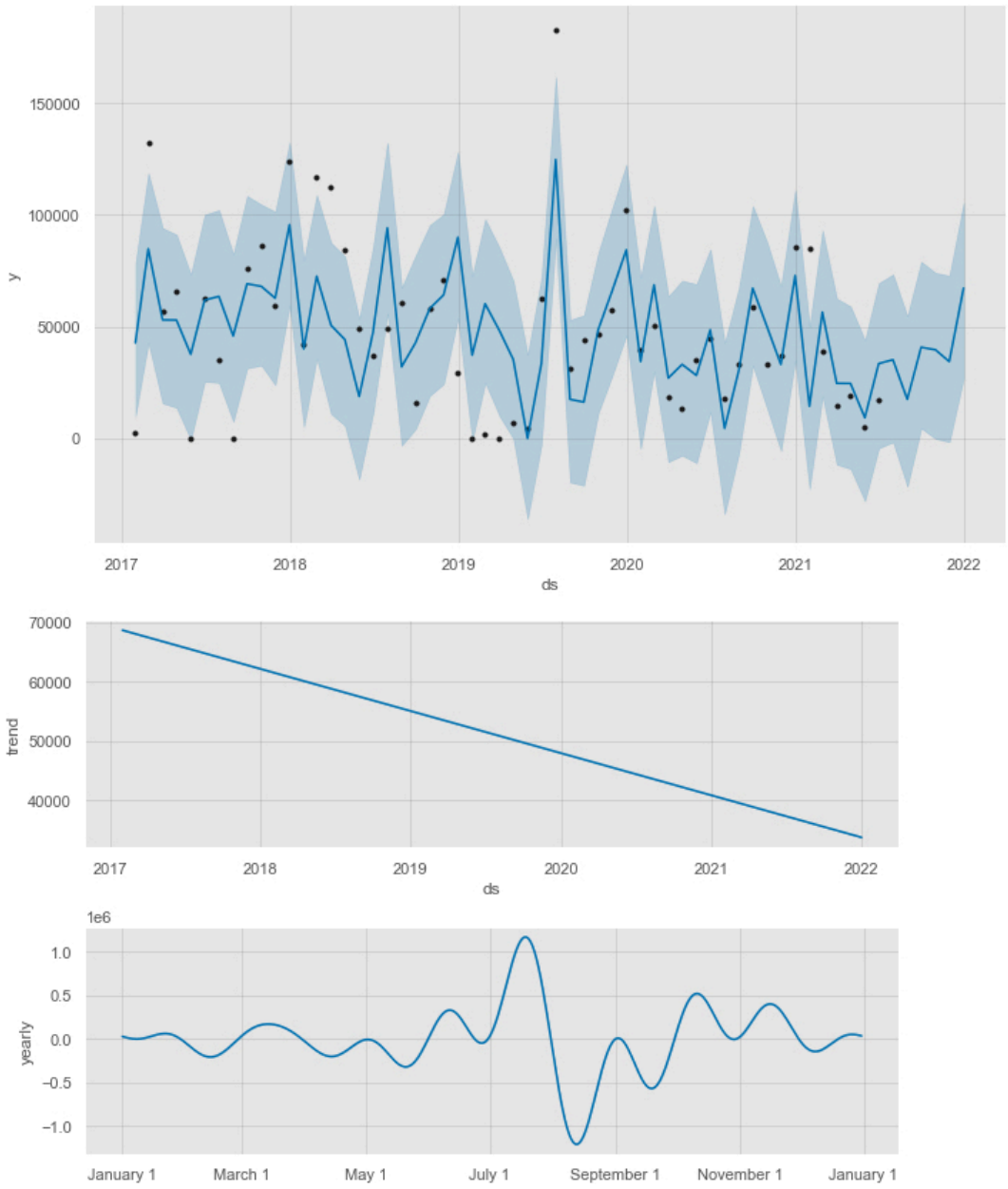
```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,1,0)[3]             : AIC=1255.323, Time=0.01 sec
 ARIMA(1,1,0)(1,1,0)[3]             : AIC=1240.884, Time=0.05 sec
 ARIMA(0,1,1)(0,1,1)[3]             : AIC=1224.291, Time=0.06 sec
 ARIMA(0,1,1)(0,1,0)[3]             : AIC=1239.961, Time=0.03 sec
 ARIMA(0,1,1)(1,1,1)[3]             : AIC=1226.266, Time=0.09 sec
 ARIMA(0,1,1)(0,1,2)[3]             : AIC=inf, Time=0.04 sec
 ARIMA(0,1,1)(1,1,0)[3]             : AIC=1233.966, Time=0.06 sec
 ARIMA(0,1,1)(1,1,2)[3]             : AIC=1228.272, Time=0.08 sec
 ARIMA(0,1,0)(0,1,1)[3]             : AIC=1227.907, Time=0.02 sec
 ARIMA(1,1,1)(0,1,1)[3]             : AIC=1225.280, Time=0.06 sec
 ARIMA(0,1,2)(0,1,1)[3]             : AIC=1225.762, Time=0.04 sec
 ARIMA(1,1,0)(0,1,1)[3]             : AIC=1227.526, Time=0.03 sec
 ARIMA(1,1,2)(0,1,1)[3]             : AIC=1227.681, Time=0.07 sec
 ARIMA(0,1,1)(0,1,1)[3] intercept   : AIC=1225.551, Time=0.15 sec

Best model:  ARIMA(0,1,1)(0,1,1)[3]
Total fit time: 0.800 seconds
1224.2912246619696
ARIMA(order=(0, 1, 1), scoring_args={}, seasonal_order=(0, 1, 1, 3),
      suppress_warnings=True, with_intercept=False)
```

The diagnostic plots show a normal behavior but the MSE didn't improve in the end:



4th step: FB Prophet

The model was able to pick up trend and seasonality without much optimization:



5th step: Deep Learning

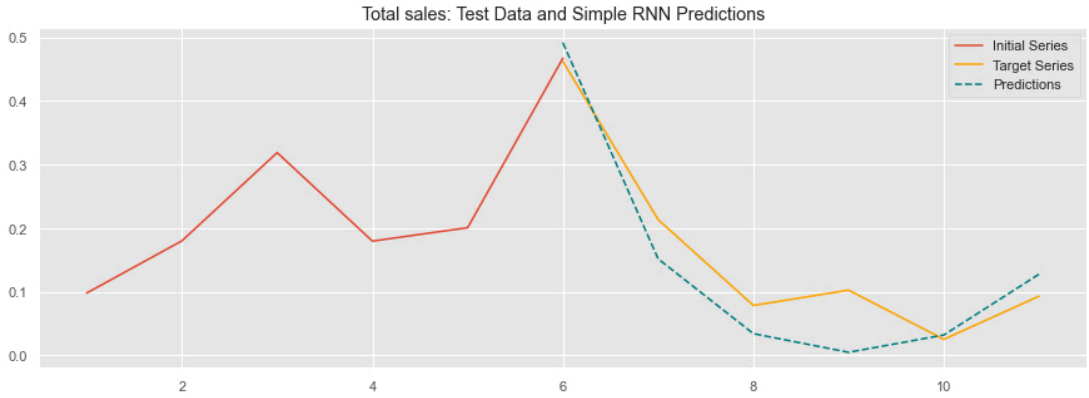
I used the support functions to prepare the time series for the Keras processing, and to generate the models. I trained a simple RNN with dense layer and two variations of LSTM.

RNN

Fitting and predicting with 70 cells and 1500 epochs. I experimented with different activation functions and different optimizers and I didn't find any striking difference. However, the model picked up the trend fairly well.

Layer (type)	Output Shape	Param #
simple_rnn_1 (SimpleRNN)	(None, 70)	5040
activation_2 (Activation)	(None, 70)	0
dense_2 (Dense)	(None, 1)	71

Total params: 5,111
Trainable params: 5,111
Non-trainable params: 0

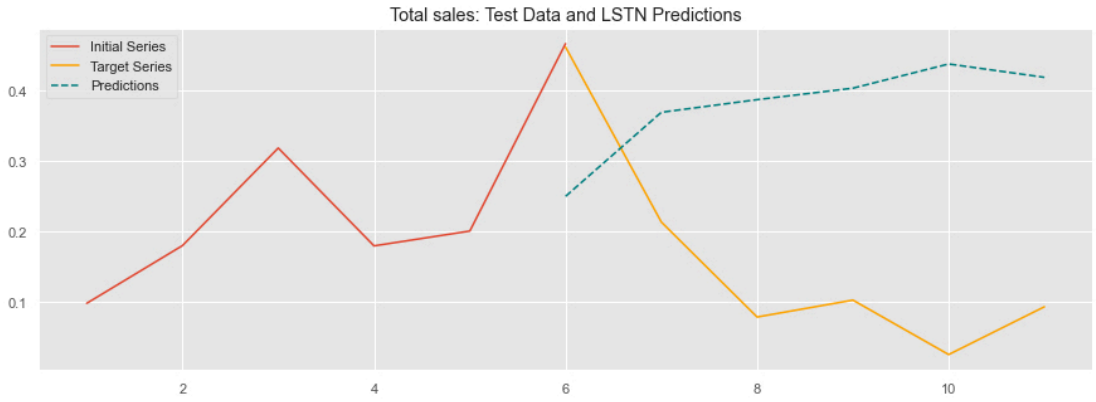


LSTM

Fitting and predicting with 70 cells and 1500 epochs:

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 70)	20160
activation_1 (Activation)	(None, 70)	0
dense_1 (Dense)	(None, 1)	71

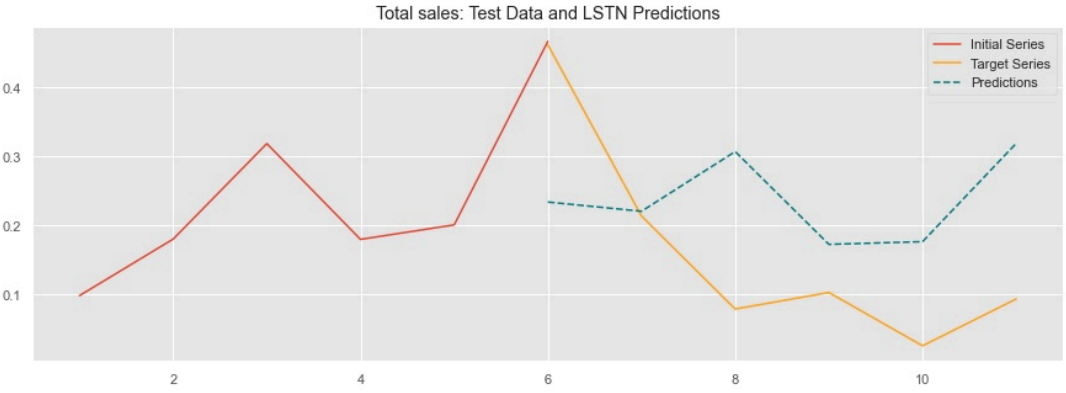
Total params: 20,231
Trainable params: 20,231
Non-trainable params: 0



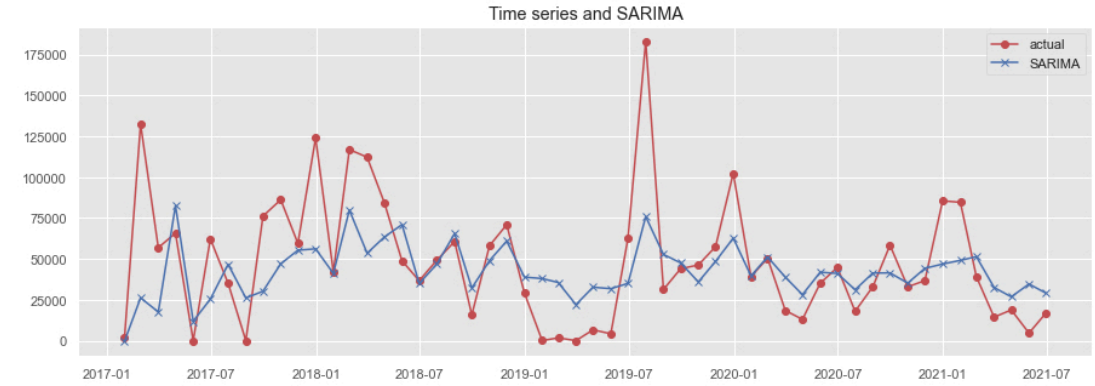
LSTM stacked

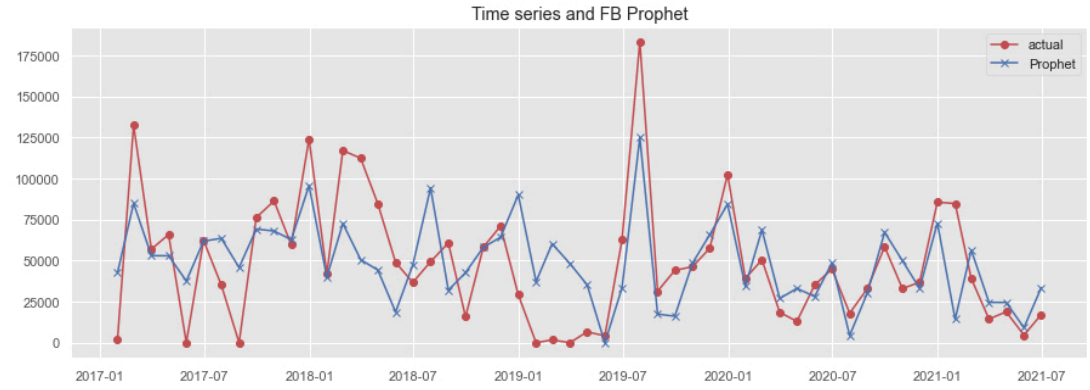
I modeled and trained a second version with 2 LSTM layers stacked, which picked up the trend slightly better:

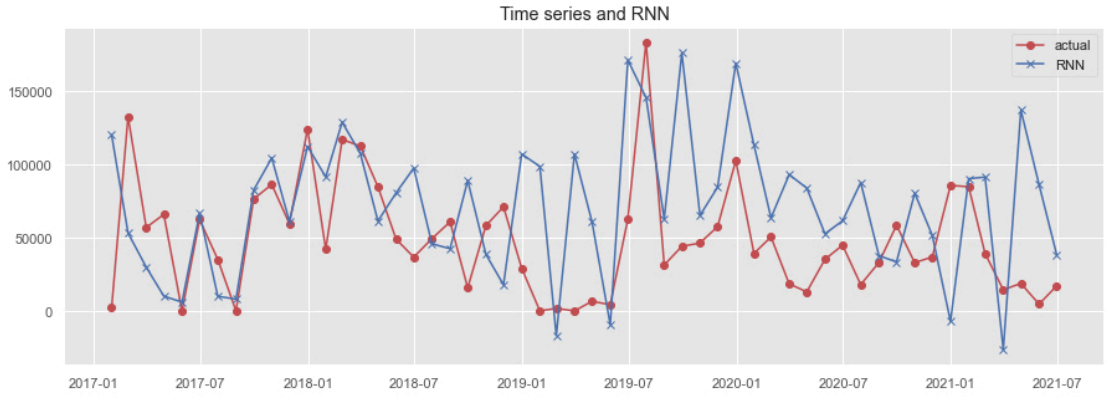
```
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_8 (LSTM)                (None, 6, 70)             20160
_________________________________________________________________
lstm_9 (LSTM)                (None, 70)                39480
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 71
=================================================================
Total params: 59,711
Trainable params: 59,711
Non-trainable params: 0
_________________________________________________________________
```

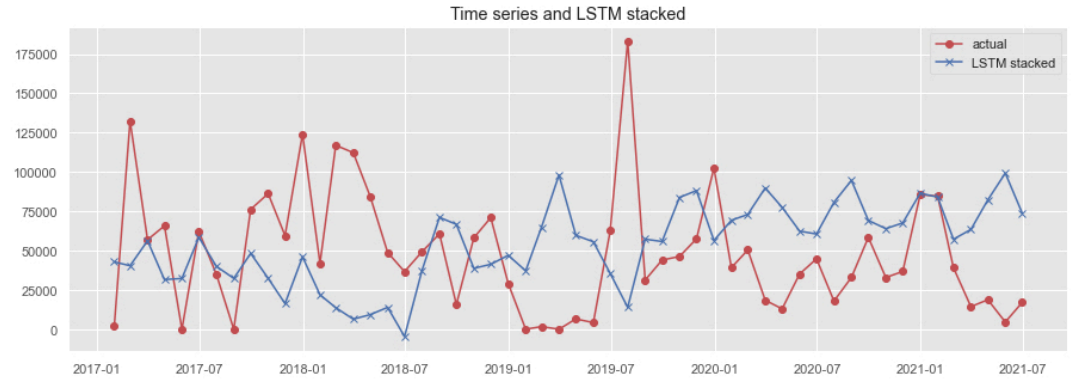


1.6.1 Plotting actual series vs predictions

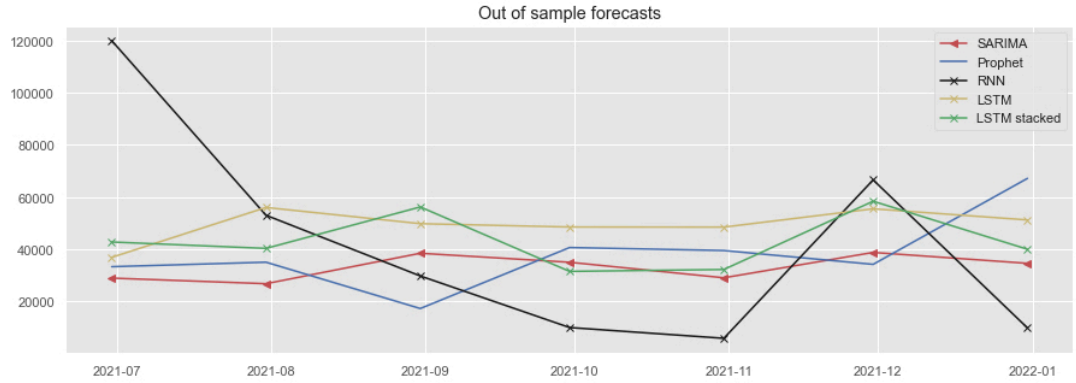








1.6.2 Plotting out-of-sample forecasts



1.7 Summary

FB Prophet ranks well and it offers options to gain insight into the data, more than the RNN which also picked up the trend very well without much of optimization. I found ARIMA and SARIMA more difficult to fine tune, while the LSTMs have been the most challenging. I didn't reach a high degree of optimization which is in the radar for the future. All the neural networks need more work to prepare the data and obtain results. For instance both the RNN and the LSTMs wouldn't pick up any trend without scaling the data.

Model	MSE
ARMA	6.83423e+08
SARIMA	2.77195e+08
FB Prophet	1.06666e+08
RNN	9.75036e+07
LSTM	1.35419e+09
LSTM stacked	1.0253e+09

1.8 Possible flaws in the model and possible improvements

My analysis can improve dramatically with more ground work to prepare the time series, and the bottom line is to obtain a stationary series and with the forecast in hands reverse the transformations to obtain real numbers. It is also possible that more processing power can make the neural networks dramatically more efficient, although they remain difficult to interprtate.

2 THANK YOU!