Prototype Implementations for Forecasting of Network Performance KPIs

Stefan Klas
sk15962@bristol.ac.uk

July 2018

Contents

1 Introduction **1**
1.1 Problem definition, goals and scope . . . 2

2 Background about mobile networks and KPIs **2**

3 Anomaly Detection System - Overall Structure **2**

4 Notation **3**

5 Prediction Algorithms **3**
5.1 Reference Algorithm . . . 3
5.2 Straight ARIMA Algorithm . . . 5
5.2.1 Parameter tuning . . . 6
5.2.2 Implementation . . . 7
5.3 Straight LSTM Neural Network Algorithm . . . 7
5.4 Dependencies . . . 8
5.5 Linear Regression With ARIMA Algorithm . . . 8
5.6 Linear Regression With LSTM Algorithm . . . 9

6 Anomaly Detection **10**

7 Analysing Results **11**

8 Challenges With Implementation **13**

9 Recommendation for next steps **13**

10 Software Implementation **14**

11 Quick User Guide For The Software **14**
11.1 Preparing for the software to run . . . 14
11.2 Running the software . . . 14
11.3 Output of the software . . . 14
11.4 Expected behaviour of the software . . . 14

12 Conclusion **15**

13 Acknowledgement **15**

1 Introduction

Telecommunication companies use KPIs (Key Performance Indicators) to assess the state of the networks at any given time, applied to different types of radio access networks (GSM,3G,4G). KPIs are an important tool used to detect and determine the source of problems in the network which could affect consumers. With the advancement of fields like artificial intelligence (AI) and growing computational power, there has been increasing interest in detecting anomalies in networks automatically.

1.1 Problem definition, goals and scope

The general problem can be summarised as follows: How can we detect anomalous network behaviour after a change to the network. Examples of changes might be: software or hardware updates, introduction of new software versions, changes of software configurations etc.
To study the above problem, we focussed on the following scenario: We assume there are N measurement probes in the network that generate N time series of actual, measured KPI values. So we have N data streams that get continuously updated.
When a change event happens, the anomaly detection system takes a certain amount of historical KPI data and predicts the next 24 hours of KPI values for those N time series.
The system will continue to collect actual KPI values. For every point in time after the event date, the system can compare actual to predicted data. An alert is raised if conclusive evidence exists that an anomaly is present.

A commercial solution to this problem will consist of many different components depending on detailed system requirements. Examples are: Classification techniques to group KPIs based on specific properties, so different algorithms and decision techniques can be used, Different prediction techniques for KPIs, Decision techniques to determine whether an observed outcome shall be flagged as an anomaly, Information feedback techniques to inform engineers of potential issues And others.

The scope of this analysis is to focus on the following goals: to study some possible prediction methods to assess how well these predictions perform, to implement a software that supports the previous goals.

Inputs

- Event data, including timestamp when event occurs. The event could be software update, hardware update etc..
- Historical data of all KPIs

Output

- Forecasted movement of KPI after event

This output would be compared with actual data received after the event and raise an alert if the KPI veered off the trend. An engineer would be made aware and the problem could be diagnosed and fixed. The report used two weeks of historical network KPI data for LTE and a given event timestamp to predict all KPIs for 24 hours.

2 Background about mobile networks and KPIs

Many of the KPI definitions, in the context of LTE, make use of telecommunication acronyms. Understanding the basics serves the purpose to interpret some of the data.
The LTE network structure consists of four sections - UE, EUTRAN, EPC and IMS. The UE (user equipment) is the user's device and this equipment is either in active or idle state. The EUTRAN (radio access network) consists of eNodeBs which are essentially base stations. Each eNodeB consists of multiple cells tasked to receive and send data from one spacial direction. The EPC (Evolved Packet Core) consists of many other components including the home subscriber server, mobility management entity and serving gateway. The PDN gateway is then the link between the mobile device and the services in systems like IMS and external packet networks like the internet. Handover occurs when the UE communication channel is moved and the alternative provides a better signal. Examples include frequency, EUTRAN handovers which can be both intra or inter. When an LTE connection is established, a RAB (radio access bearer) is set up which is a channel for data to flow and which has associated parameters such as maximum delay and packet loss limit to define quality of service. In LTE, different categories of service quality are defined using QoS Class Identifiers (QCI).
This brief description helps us understand the meaning of most KPIs. One illustration is KPI called "E-RAB Stp SR, QCI3". Using the terminology above, this represents the radio access bearer setup success rate for services allocated to QCI 3.

3 Anomaly Detection System - Overall Structure

In order to detect anomalies in the KPIs' measurements after an event, a proposed solutions is shown in Fig. 1, which consists of three stages: 1) classifying the KPI, 2) predicting the KPI and 3) anomalies detection. We will be focusing on the latter two stages.
The classification, stage 1, would involve dissecting each KPI's historical data through methods such as frequency analysis to determine the type of signal it is and from this, selecting the appropriate prediction algorithm best suited.

The second stage is the prediction stage, which uses historical data of one or more KPIs and the event date to create a model that best represents the data. This historical data normally is a sliding window, which builds a model based on the last n weeks of data. Given this model and the date, it is then possible to forecast how this KPI is expected to evolve in the future. The third stage receives the new data after the event, and determines whether an alert flag should be raised based on whether the readings lie within the upper and low bounds of the forecast made.

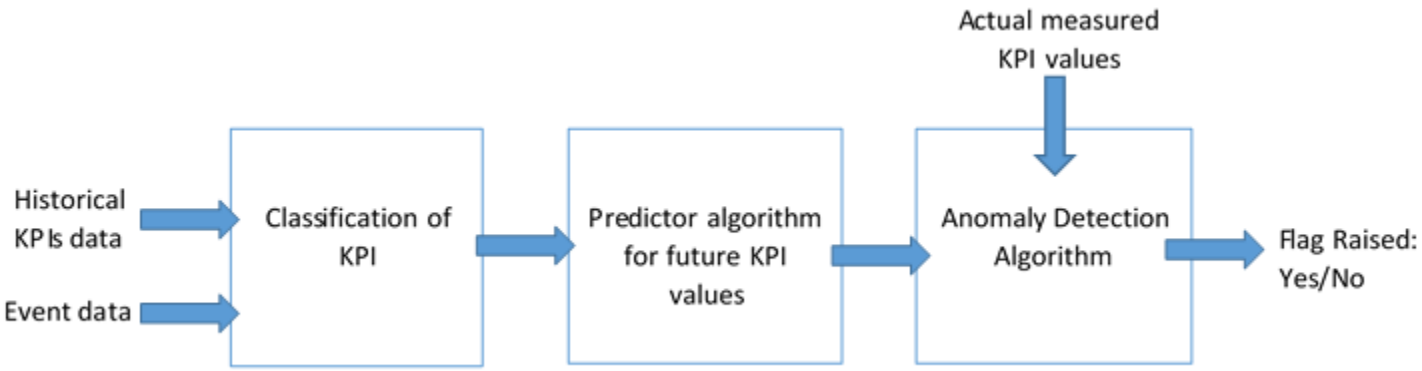

Figure 1: Structure of anomaly detection system

4 Notation

To help describe some of the algorithms, we use the following notation.

H:	The historical data of a KPI
P:	The predicted future data of a KPI
i:	A datatime index to index the KPI values
H[i]:	Historical KPI value at datetime i
DW[i]:	Day of the week of datetime (e.g. Monday)
L_H:	Length of the historical data of a KPI. FOr example: 2 weeks of data: 24 * 7 * 2

5 Prediction Algorithms

Five prediction algorithms have been trialed in making KPI forecasts. We start with a reference algorithm, which is simple and intuitive to implement, as the baseline to compare all the other proposed algorithms against and see whether greater accuracy is obtained.

5.1 Reference Algorithm

The reference algorithm predicts the value of a KPI at time point i (whereby i is a dateTime) from two previous time points in the historical data of this KPI. This has been chosen purely from observation:
First, for many of the KPIs, work day and weekend measurements differ substantially. For example, Mondays differ more from Saturdays than Tuesdays.
Second, looking only at a particular day in the week, the hourly pattern within the day is consistently similar over time. For example: The hourly KPI data for Monday 6 Aug is similar to the one for Monday the previous week. This is demonstrated by Fig 2.

Based on above, the formula is as follows:

For each datetime i we want to predict a future value P[i]:

if DW[i] = Monday or Saturday then
    P[i] = (H[i - 6 days] + H[i - 7 days]) / 2
else
    P[i] = (H[i - 1 days] + H[i - 7 days]) / 2

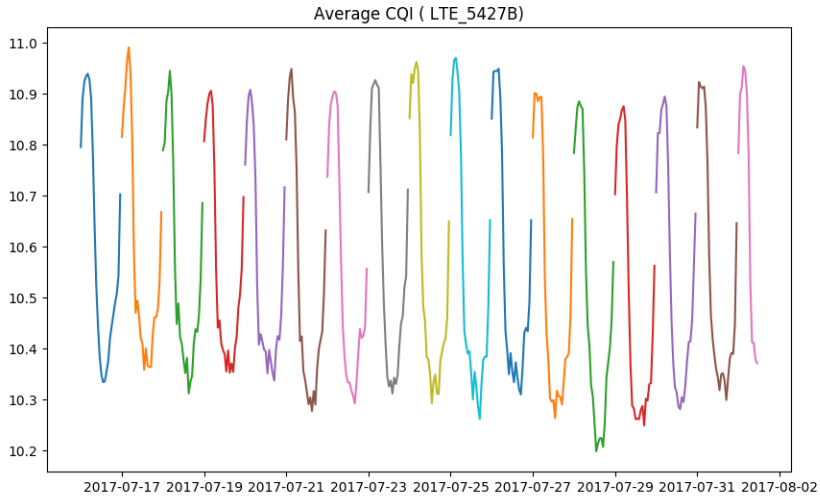

Figure 2: Historical Daily Data - Average CQI

This means, that for most days of the week, by averaging between the two days, we are capturing both the very short (within the week) and longer (outside the week) trend. Specifically for predicting time points in Saturday or Monday, the algorithm uses the previous Sunday or Tuesday respectively, instead of Friday or Sunday as the previous day's measurement.

Figure 3, with actual (orange) and forecast (blue) measurements of Average CQI, shows that this algorithm does a very good job for this KPI. This can easily be understood from Figure 2, which shows just how repetitive this KPI is on a daily basis. Figure 4, measuring E-RAB DR Activ QCI RNI is an example where it does not perform as well. This is generally due to either having a significant outlier in the two historical measurements used or if the KPI is random, showing no clear pattern and repetition.

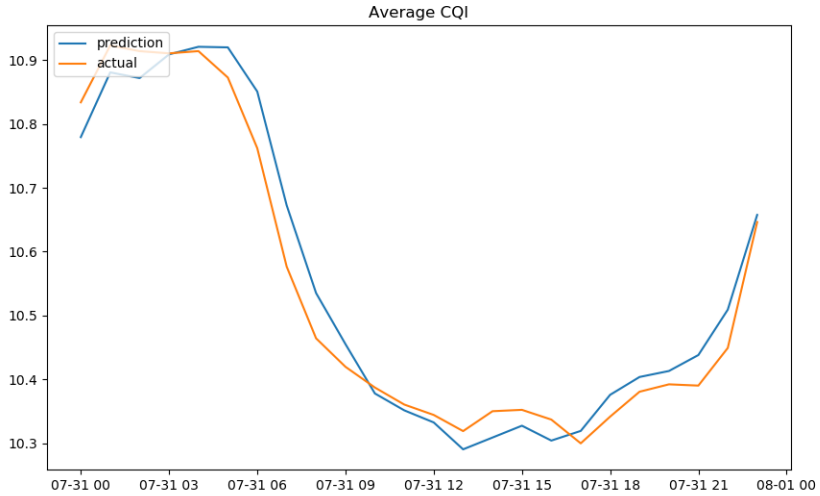

Figure 3: Forecast - Average CQI

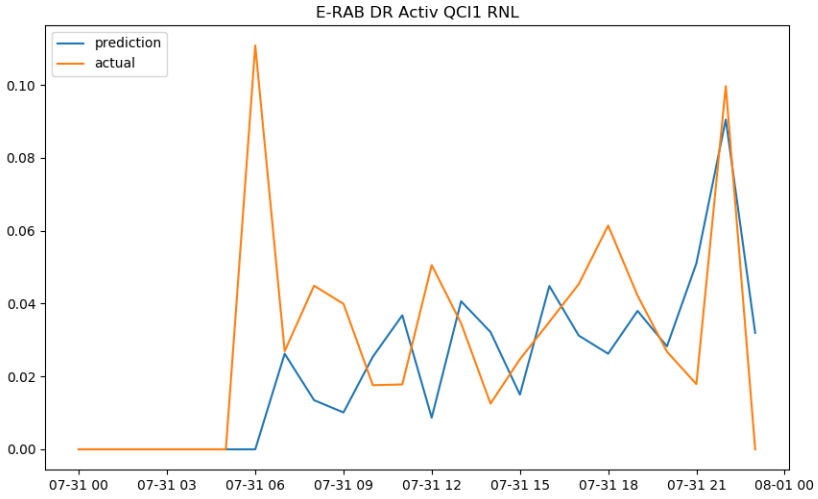

Figure 4: Forecast - E-RAB DR Activ QCI RNL

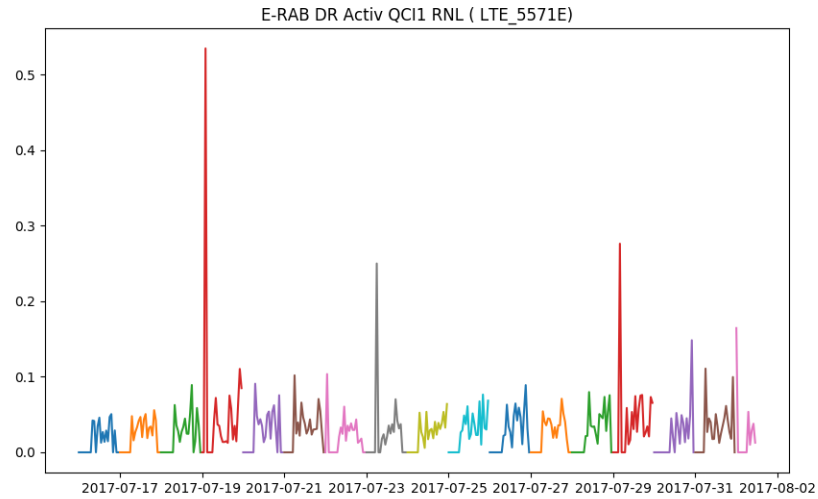

Figure 5: Historical Daily Data - E-RAB DR Activ QCI RNL

5.2 Straight ARIMA Algorithm

The ARIMA (autoregressive integrated moving average) model is a common technique used to understand data and make time series forecasts. It takes three parameters which can be tweaked to create the model that best represents the data. By feeding historical data in, the model learns the underlying coefficients that will be used for future predictions.

Parameters

- P - auto regressive (AR) component, representing the number of previous consecutive time points involved in determining the next time point
- D - difference component, used to ensure that the data is stationary with trends removed for better fitting
- Q - moving average (MA) component, representing how wide the moving average should be to influence the model.

Extensions of the ARIMA models exist, such as seasonal ARIMA models that specialise in including seasonal change information in the prediction. These both come packaged up in the *statsmodels* python module, which was used in our implementation.

5.2.1 Parameter tuning

To determine the ARIMA parameters for each KPI, one proposed method involved analysis of the autocorrelation function (ACF) and partial autocorrelation function (PACF). Figure 6 shows these two functions for E-UTRAN Init E-RAB acc over different time lags. Analyzing this example with the assumption that the data is stationary, the significant cutoff in the

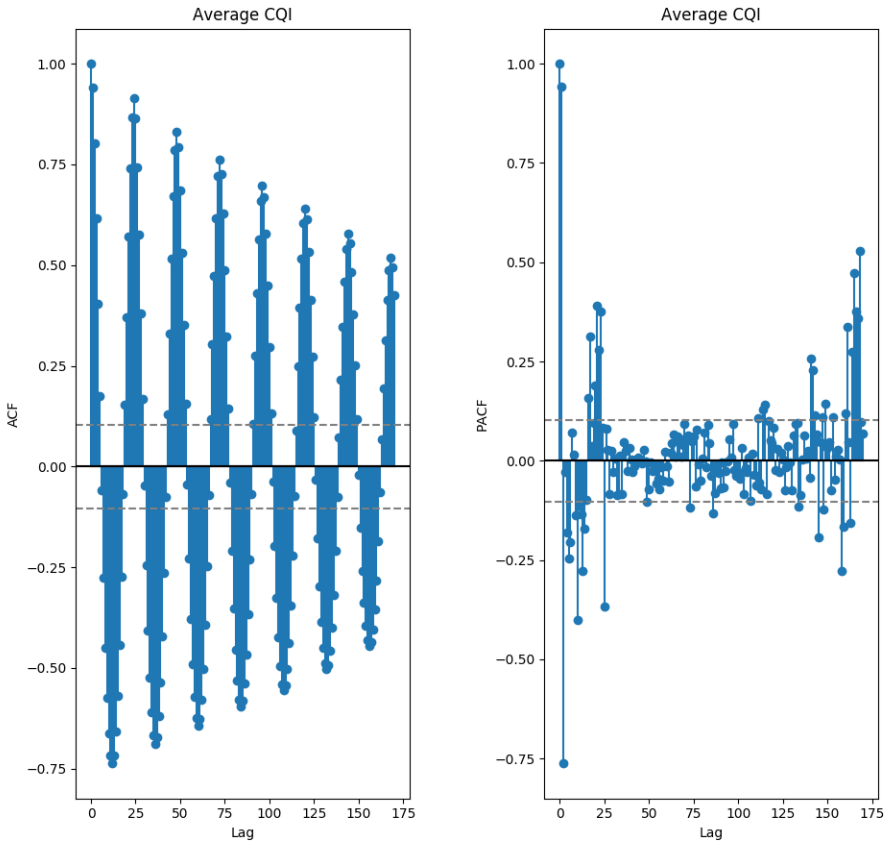

Figure 6: Left: ACF, Right: PCF for E-UTRAN Init E-RAB acc

PACF as opposed to ACF suggests it is an "AR signature" and therefore can be explained more easily by adding AR terms instead of MA terms. Also, the second highest value in the PACF occurs at lag 168 (the previous week) which could be used as the seasonality period in the seasonal ARIMA model. Due to the laborious nature of this approach and implementation memory issues with large seasonal period values, an alternative approach was used as described next.
A grid search method was used to find the perfect set of parameters for each KPI. This is a costly process having to create an ARIMA model for each possible parameter combination, however, it would have to be done only once for a given set of KPIs.
To numerically assess the quality of the candidate ARIMA models, a few possible metrics are available and could be used.

- Akaike Information Criterion
- Bayesian information Criterion
- Standard deviation of residual error
- Squared residual error
- Mean residual error
- A weighted combination of any of above.

5.2.2 Implementation

In the final implementation, the lowest standard deviation of residual error was used, representing the best fitting model. As opposed to the squared residual error, it formed a model that better represents the shape of the historical data. In contrast, taking just the squared residual error could create a more zig-zag forecast. The alternative criteria such as the AIC and BIC were also trialed but these did not produce consistently good results.
In our implementation, the input to our ARIMA model was pre-processed historical KPI data. For example, if the forecast period is 24 hours on a Monday, we include into the training data only all previous Mondays. This would mean, that if the forecast period of 24 hours stretches across two days, two ARIMA models were used. We call this in short "weekday data slicing".
Fig. 7 shows a good forecast for the stated KPI for the next 24 hours. Fig. 9 shows the training and test data for this case. On the other hand, there are cases where the straight ARIMA model doesn't perform well, as shown in Fig. 8. The corresponding hourly training and test data is shown in Fig. 10.
One reason the latter prediction may not have performed well could be due to a lack of training data. Note that we decided to apply "weekday data slicing". This means, that if our original training data is only two weeks long, we have only two full days of training data.

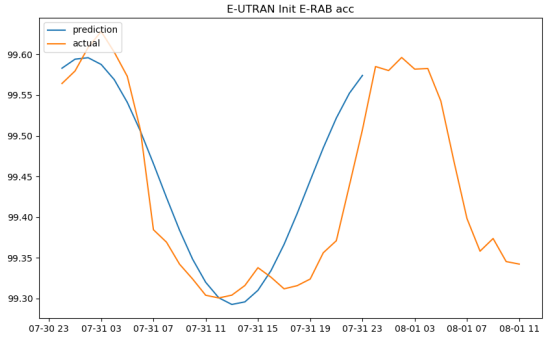

Figure 7: Straight ARIMA Forecast - E-UTRAN Init E-RAB acc

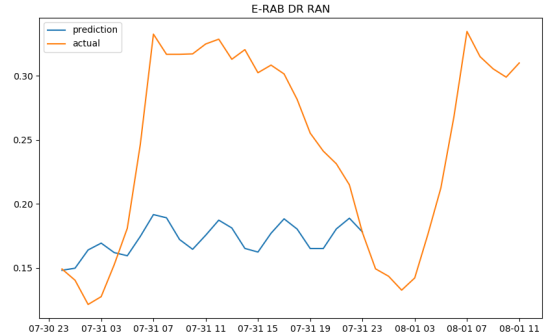

Figure 8: Straight ARIMA Forecast - E-RAB DR RAN

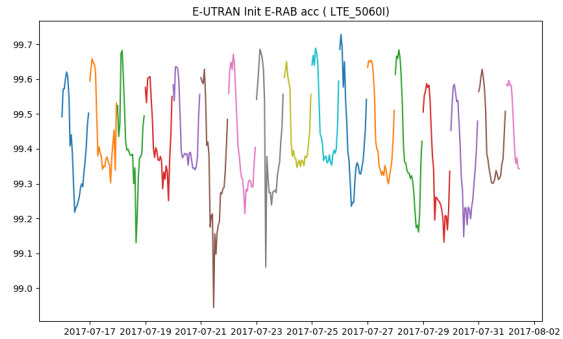

Figure 9: Historical Daily Data - E-UTRAN Init E-RAB acc

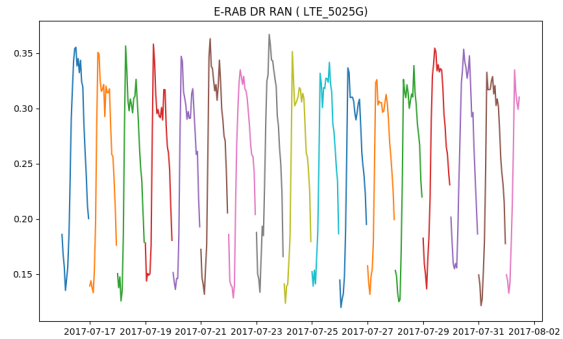

Figure 10: Historical Daily Data - E-RAB DR RAN

5.3 Straight LSTM Neural Network Algorithm

Long short term memory neural networks is another well known technique that has been used to forecast time series. Each neuron within the hidden layers contains its own state, which contains memory of what was previously fed into the network and uses this in order to determine what the output should be. Therefore given large amounts of data, it should be able to recognize patterns in both weekly trends and monthly trends in the KPIs.
The implementation of this is a n+1 forecaster, trained with back propagation by taking each day as input and the next day as output. The day of the week is also fed in such that it would be able to learn that variation in KPIs can occur during the week.
It was implemented using Keras, with TensorFlow as backend underneath. More specifically the LSTM recurrent network package was used to model the hidden layer neurons.

The model has 25 input nodes, namely the 24 hours of KPI data and a label which refers to the specific 24 hour period (e.g. a Monday, or a 24 hour period across two days). Data then flows through one hidden layer with 50 nodes, before data flows to 25 output nodes.

Thirty epochs were used (the neural network goes through the entire training data set in 1 epoch). Further testing would be required to guarantee that the best number of epochs is used to both avoid under= and overfitting To have confidence that the neural network is general enough and not overfitted, more historical data is required.
Figure 11 shows that a nearly perfect forecast could be achieved for this particular KPI.
In case of Figure 12, the model didn't perform as well, though this is most highly likely due to the greater variation in the KPI. However, there is little chance that this algorithm would be able to forecast perfectly in this case.

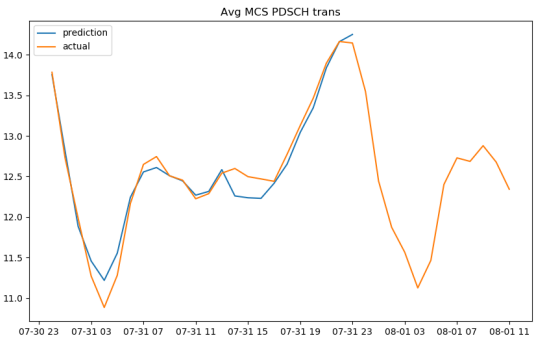

Figure 11: Straight LSTM Forecast - Avg MCS PDSCH trans

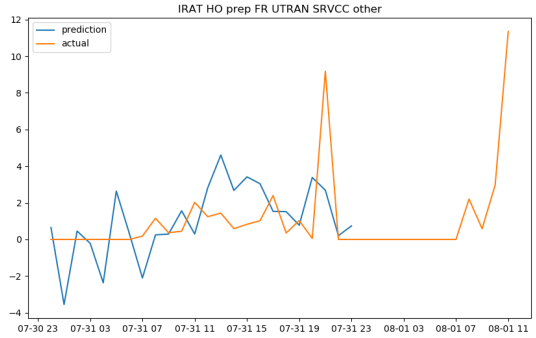

Figure 12: Straight LSTM Forecast - IRAT HO prep FR UTRAN SRVCC other

5.4 Dependencies

Some of the KPIs may depend on other KPIs. For example, high level KPIs such as "Avg act UEs DL" may be driven by many other KPIs. Therefore, it might make sense to establish those driving KPIs and consider them in the prediction of a target KPI.
In this case two challenges arise:
How to identify the KPIs that are driving a target KPI,
How to consider the KPIs in the prediction of a target KPI.

Regarding the first challenge, in order to try to find these driving forces, a correlation matrix was created where there was a strong correlation (0.9) between dependent KPIs. The belief was that by determining the traffic KPIs that were driving the higher level KPIs, it would be possible to more accurately model them.
One strategy was to apply a hard threshold for correlation. All KPIs with a correlation above the threshold value were considered. A graphical illustration of this is the correlation matrix of all KPIs shown in Fig. 13.
The second strategy was to take the three KPIs with the highest correlations with the target KPI.

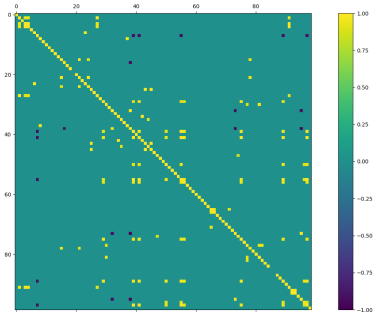

Figure 13: KPI Correlation Matrix - Threshold 0.9

The second strategy was used for the later implementations for the following reason. A hard same threshold over all KPIs led to some KPIs having little correlated KPIs and therefore the result would have been be equal to the one achieved with the the straight algorithms.

Regarding the second challenge, linear regression was the only method applied to incorporate driving KPIs to predict a target KPI. This was based on the belief, that having found driving traffic KPIs, a linear combination of them would be sufficient to forecast the target KPI.

5.5 Linear Regression With ARIMA Algorithm

Having determined the dependencies on KPIs, the next step forward was to use linear regression in order to combine them. First, we applied linear regression on the historical data for the dependent KPIs and the target KPI (input was dependent KPIs, target was target KPI). This established the coefficients of the linear regression model.

Second, we used the previous Straight ARIMA model for each dependent KPI to forecast its behaviour.
Third, by combining these forecasts through the linear regression model, we determine the predicted values for the target KPI.

This was implemented by making use of the linear regression model with the Scikit-learn Python package.

Figure 14 shows a good performing forecast, whilst Figure 15 shows a situation where this approach of using dependent KPIs to forecast a target KPI makes the prediction worse than using a single ARIMA algorithm. The reason in this specific case has little to do with low correlation. Instead, the lack of accuracy in forecasting the dependent KPIs themselves can be blamed for the poor result.

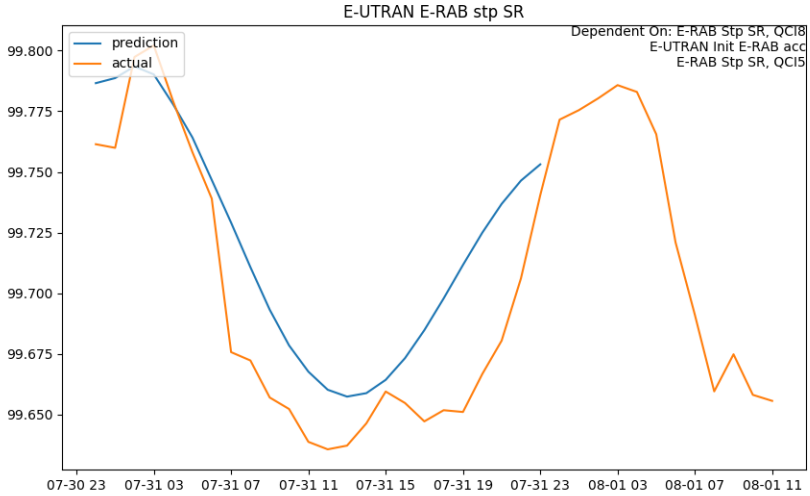

Figure 14: Linear Regression with ARIMA - E-UTRANE-RABstpSR

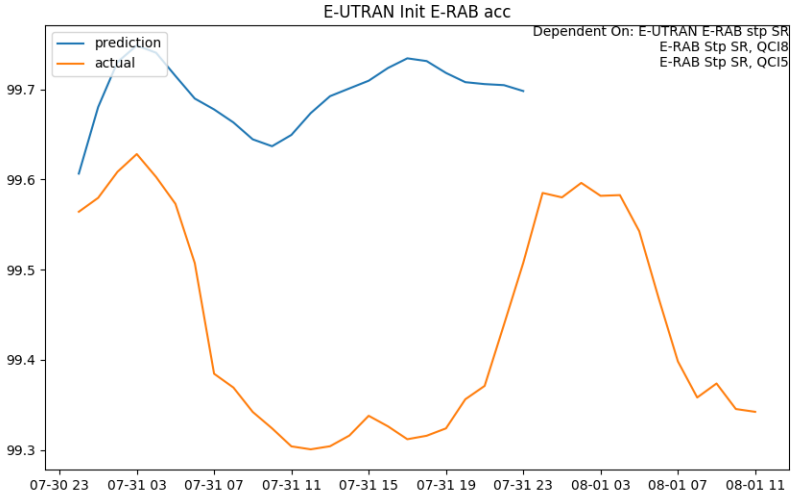

Figure 15: Linear Regression with ARIMA -E-UTRANInitE-RABacc

5.6 Linear Regression With LSTM Algorithm

Similar to the approach suggested in the previous section, another approach was to use a linear combination of highly dependent KPIs that had been forcast individually using the LSTM method instead of the ARIMA method.

Figure 16 compares predicted values (blue line) against actual test values (orange line) for a KPI called "Tot RRC conn re-estab att". The prediction has been made based on the top three most highly correlated other KPIs. These had very high

correlation of 0.97 or higher. This will explain, why the forecast is pretty good.

Figure 17 shows a situation where the target KPI has little correlation to any of the other KPIs which might be considered as potential drivers of the target KPI.

The question arises, why this algorithm would perform better in some instances compared to the Straight LSTM algorithm. One hypothesis can be the following: In the Straight LSTM algorithm, any anomalies included in the historical training data will have a strong influence on the predicted data. If however, the dependent KPIs show fewer or no anomalies in the historical data, the combined use of those KPIs can lead to a cleaner prediction for the target KPI.

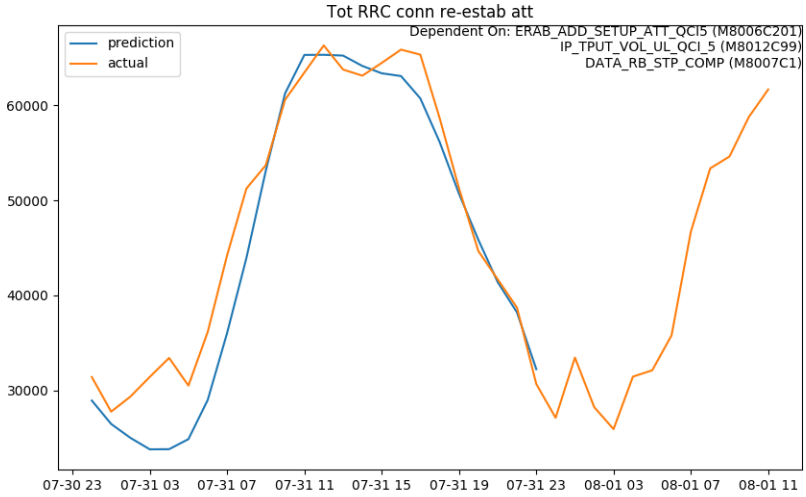

Figure 16: Linear Regression with LSTM Forecast - Tot RRC conn re-estab att

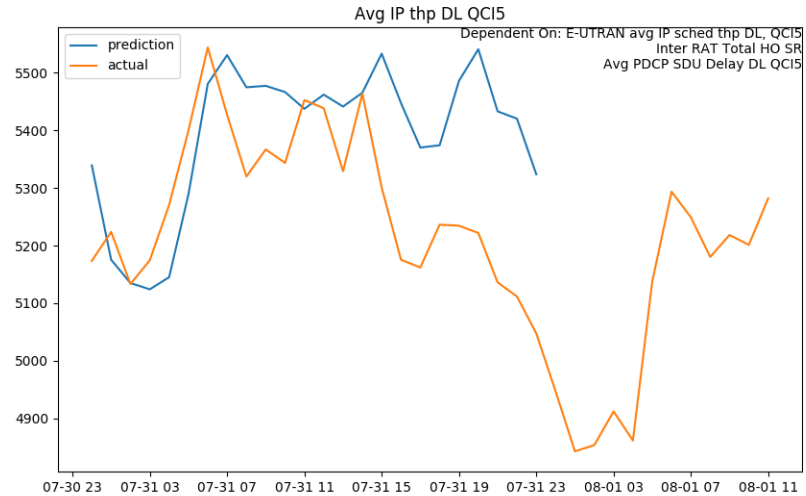

Figure 17: Linear Regression with LSTM Forecastt - Avg IP thp DL QCI5

6 Anomaly Detection

The anomaly detection step is used in order to determine if the real data provided follows the forecast and to raise an alert if it deviates too much.
In this simple implementation, the amount it must deviate at each time step is 3 times the average standard deviation at that time point in the historical data over a certain sliding time window. This for example means that if previous Mondays

at 9am have been very stable, then little leeway for up/down movement would be given to future time points of this KPI at this time.
Figure 18 shows several time series: The blue curve represents the predicted KPI values. The red curve shows the actual KPI values used for testing. The orange upper curve and the green, dotted lower curve are a variable width two-sided boundary around the predicted data series.
The chart shows that no alert needs to be raised, as the actual data measured stays within the tunnel boundaries.
Figure 19 in contrast shows a situation, where the red curve representing the newly measured actual KPI values raises too much above the organge upper tunnel boundary. In this case, an alert could or should be raised.

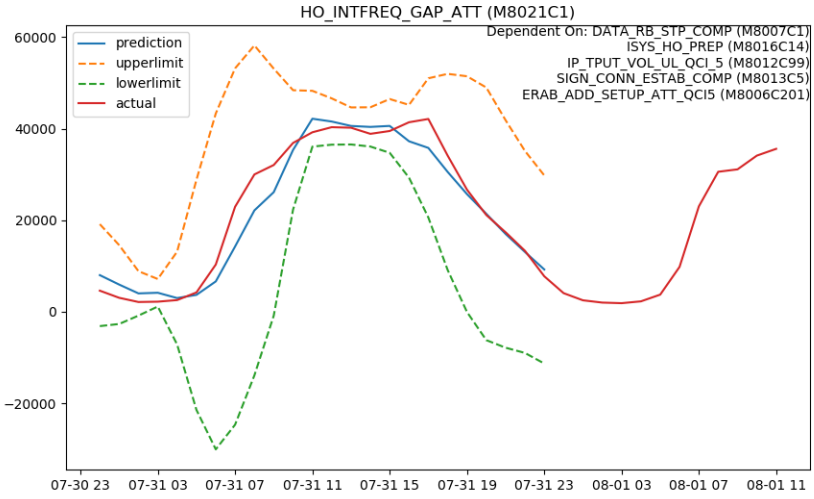

Figure 18: Anomaly detection using variable bounds - No alert raised

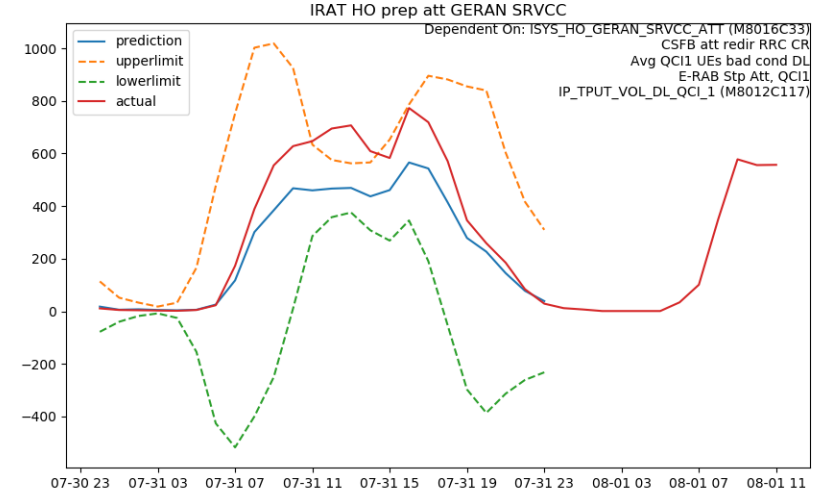

Figure 19: Anomaly detection using variable bounds - Alert raised

7 Analysing Results

In order to assess the relative performance of the different prediction methods, we calculated the Normalised Mean Squared Error (NMSE) between the prediction and the actual data. From this, we can make a few observations.

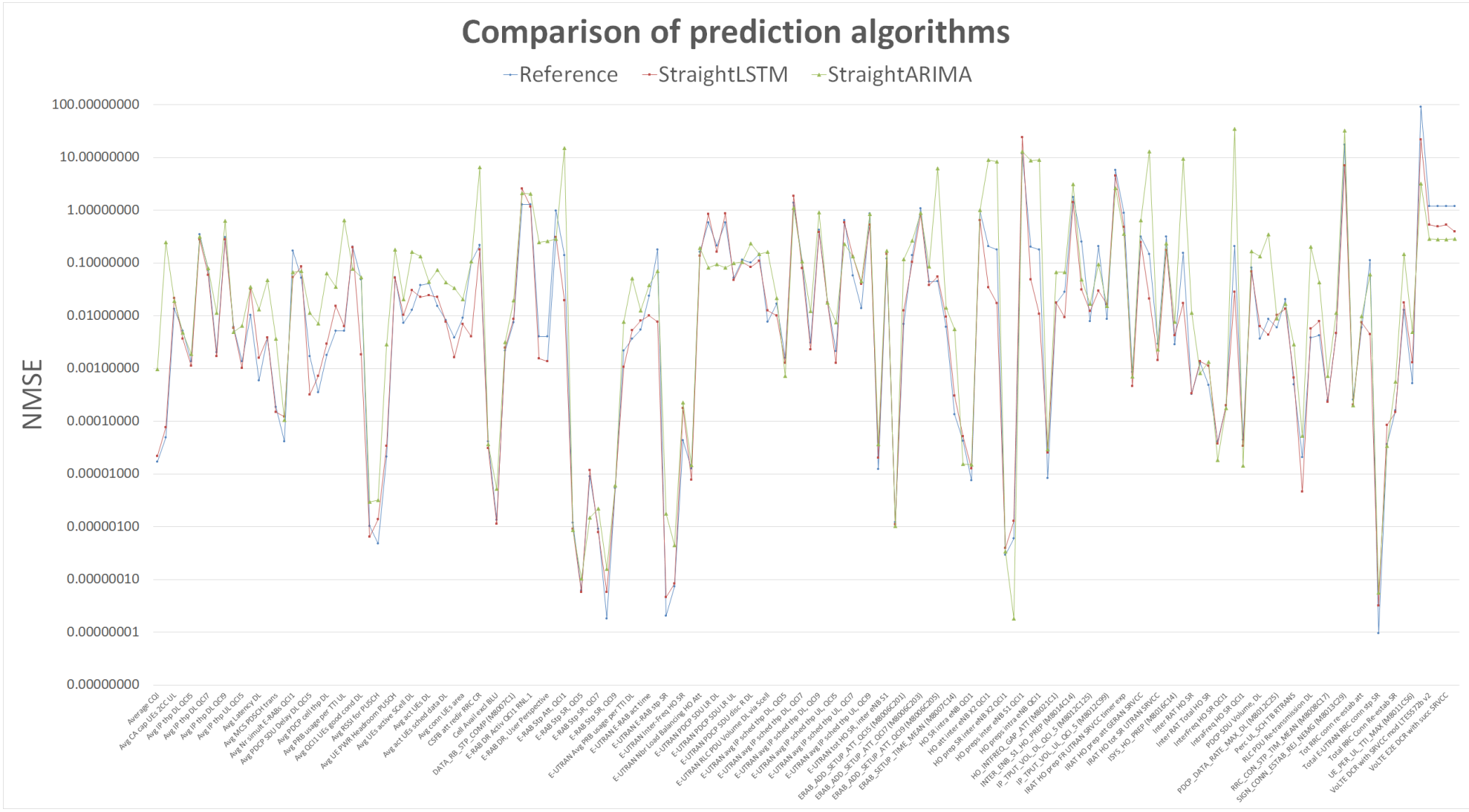

Figure 20: Comparison of prediction algorithms

- In many cases, the Straight LSTM algorithm (neural network model) is outperforming the Straight ARIMA model.
- There seem to be KPIs for which all algorithms tend to work better than for other KPIs, independent of the relative performance of the algorithms.
- For some of the KPIs, the predictions were surprisingly well. See e.g. Figure 18.
- There are some KPIs where we think the reference algorithm outperforms all other methods.
- Adding the linear regression component (not shown above but based on individual observations) can often have an adverse effect. We think this can be due to an accumulation of error or the dependency not being sufficiently captured.

The results indicate that KPIs that show repetitive patterns on a daily basis can be pretty accurately forecast. What is missing from our implementation is a better means to deal with very erratically varying KPIs driven e.g. by data traffic, special events that cause data traffic etc.

8 Challenges With Implementation

During the implementation work, we faced a number of challenges which are worthwhile to note here.

- Unclean KPI raw data. An implementation has to handle e.g. missing KPI values (i.e. time stamps with no data or blank cells in an spreadsheet) and measurement errors causing significant outliers. The data used for model training should be free of anomalies.
- Different formats of how MS Excel raw KPI data is stored by e.g. MS Excel 365 compared to Lbre Office Excel.
- Both the ARIMA and LSTM Neural Network require substantially more training data than what was provided. Ideally at least 8 weeks of hourly data. This will not only help learn the general trend better but will also mean that the classification of whether outside the trend is less effected by outliers.
- Issues occurred with ARIMA, when the model was not able to capture any trend from the historical data of a KPI. This was caused by the automated hyper-parameter selection not being good enough. Manual fine-tuning would be necessary.
- Certain KPI are very difficult to model due to them having intuitively no dependence on other KPIs. An example is "Avg Cell Availability" which is a relatively random measurement. The anomaly detection for such KPIs must provide greater leeway for wobbling.
- Establishing traffic KPIs that are driving target KPIs through using correlation doesn't give great results. Having expert knowledge of how KPIs are defined and measured and how they might depend on other KPIs for systematic reasons would produce better results.

9 Recommendation for next steps

Future work could build on this analysis in various ways.

- The developed algorithms should be tested in situations when a change event has happened and an anomaly has developed. This could be simulated or live KPI data.
- Experimentation with larger training data to determine that over-fitting has no occurred.
- Initial classification of the KPIs can be performed using methods such as frequency analysis and/or expert know-how. Based on the classification of the KPIs, different types of prediction algorithms could be used. No one algorithm will work for all KPIs.
- Classification of KPIs could also be undertaken based on grouping KPIs based on the Normalised Mean Squared Error we have calculated in our implementation. In combination with initial classifications, this would allow to map groups of KPIs to most suitable prediction algorithms.
- Exploration of further techniques to predict KPIs.
- Trialling using mutual information to try to determine dependent KPIs: This Was attempted but not successful yet.

Package	Description	Use in implemented software
Pandas	Library for data manipulation and analysis	Used for data manipulation
pyplot	Library for creating graphics plots	For several charts in this report
numpy	Numeric Python library	Used for algorithms
scikit-learn	Python machine learning library	Used for linear regression algorithm
keras	Machine learning tool on top of TensorFlow	Used for the LSTM implementation
StatsModels	Statistics library that implements ARIMA algorithm	To realise ARIMA predictor
XlsxWriter	Library that supports writing data from Python and Pandas dataframes to Microsoft Excel	To store predicted data in Excel format, e.g. for future post-processing in Excel

Table 1: Main Python packages used in the implementation.

10 Software Implementation

In this section, we provide some basic information about the analysis software that has been implemented.
We have chosen an implementation in Python over working with Microsoft Excel and the VBA language for two reasons: First, Python is widely used for data analytics and machine learning applications. Second, a large number of special libraries are available that assist in fast software implementation.

Table 1 lists the most important Python packages we have used.

11 Quick User Guide For The Software

This section is a "mini user guide" that simply shows how to use the software.

11.1 Preparing for the software to run

If need be, the software can be configured using the file "config.py".

The software requires the following input files:
Data comes in in an Excel workbook. See HourlyData.xlsx

11.2 Running the software

The main Python script is called predictor.py

To run the software in the Python environment from the command line, enter:

python predictor.py all - for running all algorithms
python predictor.py ref - for running only the reference algorithm

The main program takes the following arguments:
all: In this case, all implemented predictor algorithms are used to process the KPI data.
ref: In this case, only the reference algorithm is used to process the KPI data.

11.3 Output of the software

The software produces the following types of output files:
Image files for charts
Excel spreadsheet files for each implemented prediction algorithm. Each file holds for each KPI predicted and actual KPI data (for test purposes).

11.4 Expected behaviour of the software

When the program is called with python predictor.py all, then the expected behaviour is as follows:

- The program reads the input file that holds the KPI data.

- KPI data is split into training and testing data.
- Using a loop, the program applies in sequence 5 different predictor algorithms to the training data.
- The program stores predicted versus actual test data in a result file per algorithm, draws a comparison graph in the respective folders, and calculates the NMSE results.

12 Conclusion

Our analysis into network anomaly detection based on Key Performance Indicators has focussed on looking into different prediction algorithms whilst applying a simple decision method to determine wether an anomaly has occured or not.
One conclusion is that there doesn't seem to be a single, unique optimal solution for KPI predictors that would work best across all KPIs.
Some of the methods proposed could accurately forcast certain KPIs, appear to perform well and could possibly be used without much further extension.
For many others, this doesn't apply, and thus, further experimentation is recommended. To this end, we identified a number of suggested next steps.

13 Acknowledgement

I would like to thank Eric Jones from Nokia for the guidance and support throughout this short, but highly interesting project.

References

[1] Arima models for time series forecasting. https://people.duke.edu/ rnau/411arim.htm.

[2] How to create an arima model for time series forecasting in python. https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python.

[3] Keras recurrent layers. https://keras.io/layers/recurrent.

[4] Time series prediction with lstm recurrent neural networks in python with keras. https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras.

[5] Understanding lstm networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs.