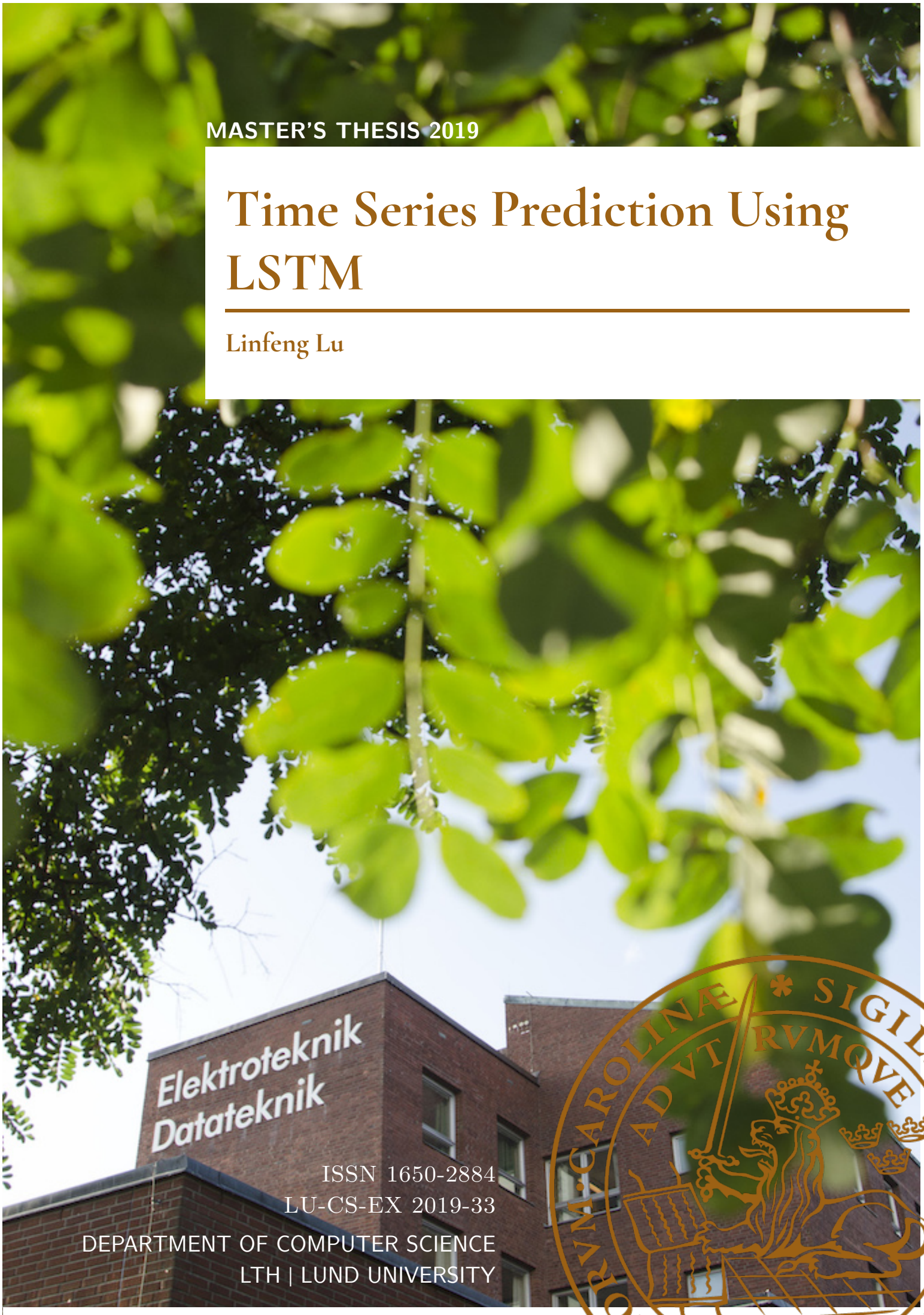


MASTER'S THESIS 2019

Time Series Prediction Using LSTM

Linfeng Lu

ISSN 1650-2884
LU-CS-EX 2019-33

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2019-33

Time Series Prediction Using LSTM

Linfeng Lu

Time Series Prediction Using LSTM

Linfeng Lu
nge11lu@student.lu.se

November 11, 2019

Master's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisors: Pierre Nugues, Pierre.Nugues@cs.lth.se
Lars Novak, Lars.Novak@sinch.com Jianghua Cao, Jianghua.Cao@sinch.com

Examiner: Flavius Gruian, flavius.gruian@cs.lth.se

Abstract

Time series prediction is the use of a model to predict future data based on previously observed data. Time series prediction and anomaly detection is important for many businesses in the world today. In this report, we experiment with three different models for time series prediction and anomaly detection on data about mobile message traffic volume and Successful Delivery Rate (SDR). The data are provided by Sinch, which is a telecommunications and cloud communications platform as a service company. We compare and analyze the experiment results and find that the model architecture that includes a LSTM encoder-decoder could improve the model performance on Sinch data, but not in all cases, due to the difference of the datasets.

Keywords: Time Series Prediction, LSTM, anomaly detection, encoder, decoder

Acknowledgements

We would like to thank our supervisor Pierre Nugues for his continuous guidance and advice on our thesis work. In addition, we thank Flavius Gruian for his help and support. We also would like to thank our supervisors Lars Novak and Jianhua Cao at Sinch for the opportunity of doing our thesis there, as well as for the guidance and support along the way. We also would like to thank Niklas Jönsson for collecting the data for us. We also would like to thank all the colleagues at Sinch who supported us.

Contents

Chapter 1
Introduction

In this chapter, we give the background of our work, state the research questions, and give an overview of the previous relevant works.

1.1 Background

Time Series forecasting is the use of a model to predict future values based on previously observed values (Chatfield, 2000). Time series forecasting and estimation of the prediction uncertainty are important for anomaly detection, which is important for many businesses in the world, such as banks and hedge funds (Zhu and Laptev, 2017). Recently, time series modelling based on the long short term memory (LSTM) model became more and more popular (Assaad et al., 2008). It has been shown that a model which includes an LSTM encoder-decoder performs well on time series prediction and anomaly detection with the trips data at Uber (Zhu and Laptev, 2017).

The thesis work has been done in collaboration with Sinch, which is a telecommunications and cloud communications platform as a service (PaaS) company. The main service of the company is to deliver mobile messages for its client companies. These client companies use Sinch's platform to send mobile messages to their customers. Most top10 fortune 100 companies are Sinch's client companies and Sinch is trusted by major banks in the U.S, Germany, and Nordic countries. Every month, billions of messages are sent through Sinch.

Figure 1.1 shows the flow of the mobile messages. The client company's messages are sent to Sinch's platform first, and then they are routed to different suppliers and operators, and finally operators send them to the mobile phones of the client company's customers.

Since the services Sinch provides are commodity services, it is very important to control the cost and the quality of the service in order to keep on being competitive in the market. So it is necessary to keep a close watch on the key metrics such as mobile message traffic volume and Successful Delivery Rate (SDR). When a client company's traffic volume behaves anomaly, it could indicate that the client company might switch to use a competitor's plat-

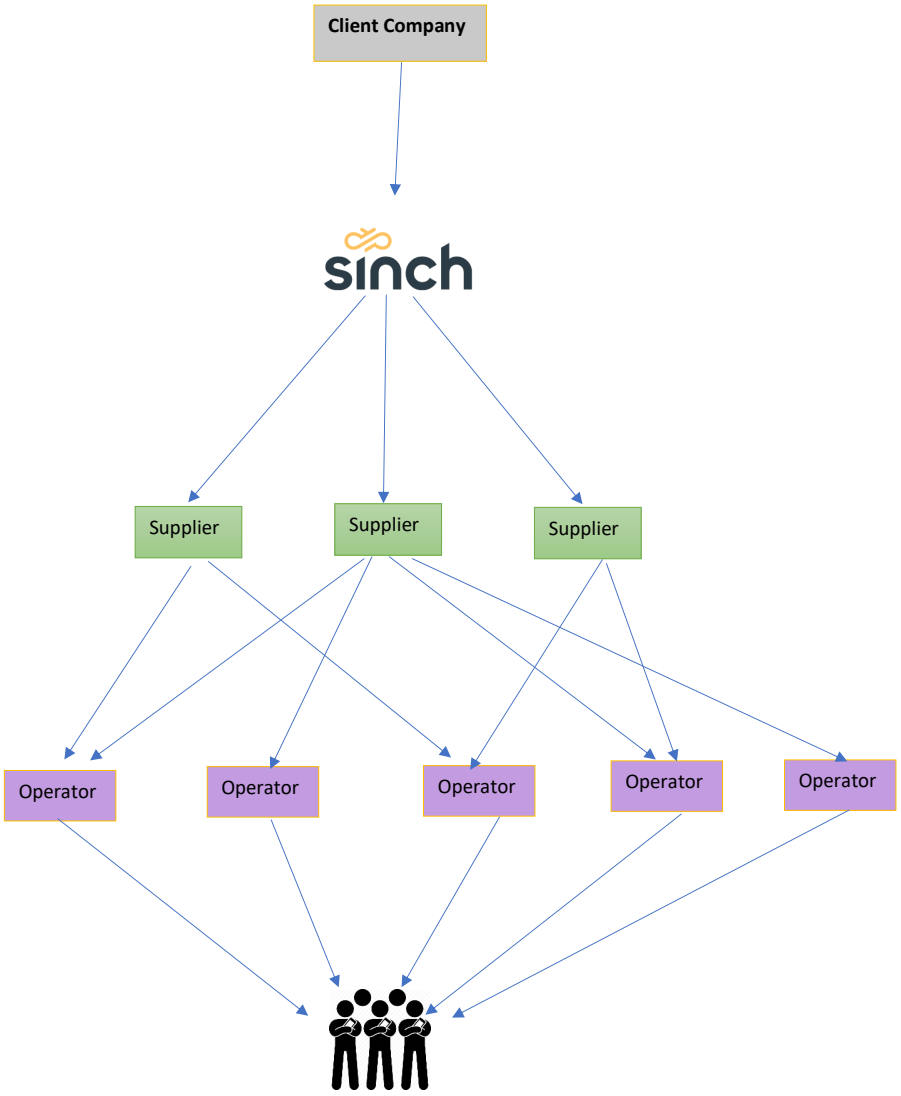


Figure 1.1: The flow of the mobile messages

form to deliver its messages. When the SDR behaves anomaly, it could possibly be caused by some technique failure, which will reduce the quality of service provided to the client company. Therefore, there is a need to detect the traffic volume anomaly and SDR anomaly. Currently, Sinch has no method to do this.

In this master's thesis, we implement three different models and compare their prediction results to find which one could perform better than others, given datasets provided by Sinch.

1.2 Contributions

The contributions of this thesis, we implemented the models with LSTM encoder-decoder proposed by Laptev et al. (2017) and Zhu and Laptev (2017), and tested them on Sinch's data. This gives an insight on how these models work with real data in the telecommunication sector. We compared the performance of these two models which were not compared with each other before and we found out that the model Zhu and Laptev (2017) proposed outperform the model proposed by Laptev et al. (2017) given the dataset we have. This thesis also will help Sinch better detect its traffic volume anomaly and SDR anomaly since currently there is no mechanism in the company to do this.

1.3 Research Question

Time series prediction is usually carried out with linear regression (Tim et al., 1996). There are recent works using deep learning in the form of LSTM that were also applied to time series prediction. Given the difference in datasets, it is not completely clear how the different algorithms will behave on specific data.

The goal of the research is as follows:

Implement three different machine learning models for time series prediction and anomaly detection:

- A regression model
- A model which includes a LSTM encoder-decoder and a regression forecaster
- A model which includes a LSTM encoder-decoder and a LSTM based forecaster

Given the datasets provided by Sinch as training data and test data, the prediction results and anomaly detection results are compared to tell which model works better for the datasets from Sinch.

1.4 Related Works

Tim et al. (1996) compared time series forecasts based on regression model with forecasts from traditional statistical time series method. They found that regression model generally perform better than traditional statistical method when forecasting real time series.

Laptev et al. (2017) has published a neural network forecasting model which could perform better than classical time series methods when dealing with interdependent, long time

series (Zhu and Laptev, 2017). However this model does not provide any information about the uncertainty, therefore, it caused a large false anomaly rates when the model prediction has large variance such as during holidays (Zhu and Laptev, 2017). The prediction uncertainty could be used to assess how much to trust the prediction produced by the model, therefore it is useful to anomaly detection (Zhu and Laptev, 2017).

Gal and Ghahramani (2015) proposed a Monte Carlo dropout (MC dropout) framework, which provides uncertainty estimation without requiring any change of the existing model architecture. This framework is not only generic, but also easy to implement, so it could be directly applied to an existing neural networks (Gal and Ghahramani, 2015). In the thesis, we will apply this MC dropout framework to all the models.

Zhu and Laptev (2017) adapted The MC dropout framework to conduct time series prediction and anomaly detection. Zhu and Laptev (2017) proposed a novel model architecture that provides time series prediction and quantify the prediction uncertainty using Bayesian neural network. They did experiments on the proposed model on trips data, and successfully applied this model to time series anomaly detection at Uber (Zhu and Laptev, 2017). This model could handle extreme event (e.g., holidays, festivals) prediction better than many classical time series models (Zhu and Laptev, 2017).

In this thesis, we will implement a regression model and the above two model architectures proposed by Laptev et al. (2017) and Zhu and Laptev (2017) respectively. We will explain more about these model architectures in Chapter 4.

1.5 Outline

The rest of the thesis is organized as follows:

Chapter 2 contains the theoretical background needed to understand the thesis.

Chapter 3 describes the datasets and model architectures that we use for our solution.

In Chapter 4, we provide detailed experiments to evaluate the model performance on Sinch data.

Finally, Chapter 5 concludes the report and give the possible future work could be done.

Chapter 2
Theory

This chapter contains the theoretical background needed to understand our solutions.

2.1 Time Series Prediction

Time series is a sequence of data points measured at consistent time intervals over a period of time (Konar and Bhattacharya, 2017). Examples of time series datasets are the daily price of the Sinch stock, daily temperature in Lund, etc.

Time series prediction is the use of a model to predict future values based on previously observed values (Chatfield, 2000). Time series prediction has become a popular topic for many businesses such as banks, hedge funds, etc, since it can help the businesses make better investment strategies and decisions (Konar and Bhattacharya, 2017).

Classical time series models, e.g. Auto-Regressive Integrated Moving Average (ARIMA), are the models that do not use machine learning techniques. They have some limitations such as they can only work on univariate data. They do not support multiple variables to be taken as inputs. (Laptev et al., 2017).

2.2 Long Short Term Memory (LSTM)

Recurrent neural networks (RNN) are a family of neural networks for processing sequential data (Goodfellow et al., 2016). RNN can keep contextual information from the input data sequence due to its recurrent connections (hoon Oh et al., 2017). RNN can make good use of its input information since the loops within RNN help it memorize previous events, but RNN cannot handle long term correlations (Tang et al., 2016).

LSTM is an improved version of RNN. LSTM is able to increase the effective context size of RNN by replacing each hidden unit with a memory block (Mulder et al., 2015). Figure 2.1

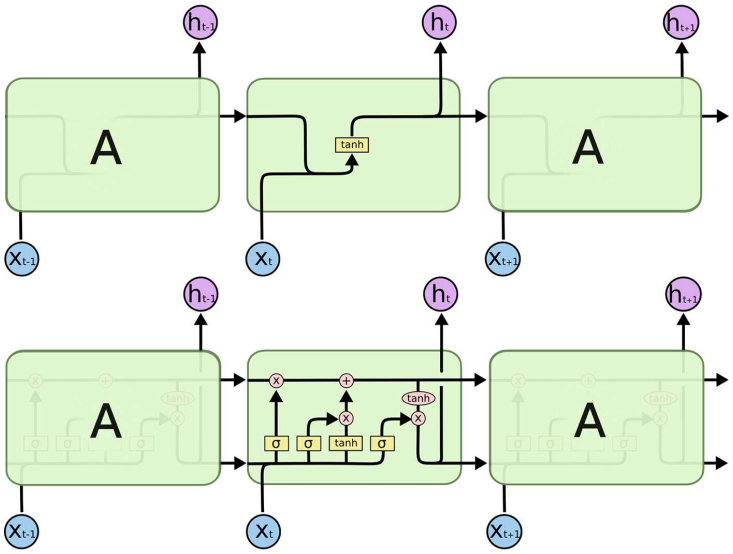


Figure 2.1: RNN and LSTM layout with Cell Connections (Mohan and Gaitonde, 2018). The top figure is RNN and the bottom figure is LSTM

shows RNN and LSTM Layout with cell connections (Mohan and Gaitonde, 2018). In addition to the outer recurrence of the RNN, LSTM cells have an internal recurrence (Goodfellow et al., 2016). This makes LSTM can handle long term correlations better than RNN.

A LSTM cell contains a system of gating unit that is applied on it (Goodfellow et al., 2016). The LSTM controls the input information flow through these three kinds of gates by selectively removing information (forget gate), adding information (input gate), or letting it through to the next cell (output gate) (Mohan and Gaitonde, 2018). The forget gate can reduce over-fitting since it does not retain all information from the previous time steps (Mohan and Gaitonde, 2018). The function of the gates and selective information control are the main reason that LSTMs can retain contextual information for a longer time (Mohan and Gaitonde, 2018).

Figure 2.2 shows the internal structure of an LSTM cell (Mohan and Gaitonde, 2018). The forget gate is denoted by f_t, input gate by i_t and output gate by o_t (Mohan and Gaitonde, 2018). The cell input is denoted by x_t, the cell output is represented by h_t and the cell state is given as C_t (Mohan and Gaitonde, 2018). Below are the equations to calculate the gates and

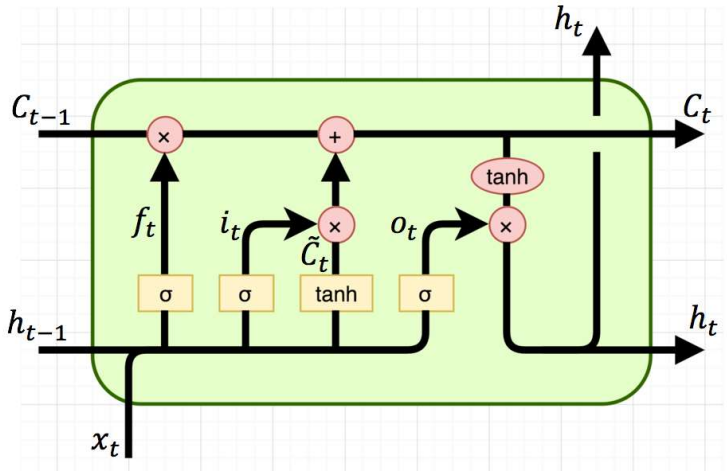


Figure 2.2: Architecture of a LSTM Cell (Mohan and Gaitonde, 2018)

states, where W denote the weights for the corresponding gate (Mohan and Gaitonde, 2018).

$$\begin{aligned} f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \widetilde{C}_t &= tanh(W_c \cdot [h_{t-1}, x_t] + b_C) \\ C_t &= f_t * C_{t-1} + i_t * \widetilde{C}_t \\ o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t * tanh(C_t) \end{aligned} \tag{2.1}$$

LSTM has been found successful in many applications, such as speech recognition (Graves and Jaitly, 2014) and machine translation (Graves and Jaitly, 2014).

During recent years, time series modeling based on the LSTM technique became popular (Hochreiter and Schmidhuber, 1997). It is probably because it can incorporate exogenous variables easily and extract feature automatically (Assaad et al., 2008). For instance, when fed with multi-variable historical observations of the target and exogenous variables, LSTM can blend the information of all variables into the memory cells and hidden states which are used for prediction (Assaad et al., 2008).

2.3 Prediction Uncertainty

Let $f^{\hat{W}}(.)$ represents a trained neural network, where $\hat{W}$ denote the fitted parameters (Zhu and Laptev, 2017). If we are given a new sample x^*, the model prediction could be represented as $\hat{y}^* = f^{\hat{W}}(x^*)$ (Zhu and Laptev, 2017).

A prediction interval is an estimate of an interval in which a future observation will fall, with a certain probability, given what has already been observed.

In order to evaluate the uncertainty of the model prediction $\hat{y}^*$, we need to first quantify the prediction standard error η, then an approximate α-level prediction interval is

$$[\hat{y}^* - Z_{\alpha/2}\eta, \hat{y}^* + Z_{\alpha/2}\eta] \tag{2.2}$$

where $Z_{\alpha/2}$ represents the upper $\alpha/2$ quantile of a standard normal (Zhu and Laptev, 2017).

The prediction interval is very useful for anomaly detection (Zhu and Laptev, 2017). For example, anomaly will be reported when the observed value lies outside the constructed 95% prediction interval (Zhu and Laptev, 2017). We can see that underestimating η could cause high false positive rates (Zhu and Laptev, 2017).

According to the MC dropout framework, proposed by Gal and Ghahramani (2016), if we apply stochastic dropouts after each hidden layer, then we can approximate the model output as a random sample generated from the posterior predictive distribution (Gal and Ghahramani, 2016). This means we can estimate the model uncertainty by the sample variance of the model predictions in a few repetitions (Gal and Ghahramani, 2016). For example, given a new input x^*, we compute the neural network output with random dropout with a certain probability p at each layer (Zhu and Laptev, 2017). We repeat this stochastic feed forward B times and the prediction results are $\{\hat{y}^*_{(1)}, ..., \hat{y}^*_{(B)}\}$ (Zhu and Laptev, 2017). The sample variance can be calculated as (Zhu and Laptev, 2017):

$$\begin{aligned} \eta^2 &= \widehat{Var}(f^W(x^*)) = \tfrac{1}{B}\textstyle\sum_{b=1}^{B}(\hat{y}^*_b - \bar{\hat{y}}^*)^2 \\ \bar{\hat{y}}^* &= \tfrac{1}{B}\textstyle\sum_{b=1}^{B}\hat{y}^*_b \end{aligned} \tag{2.3}$$

We can use the sample variance to approximate the model uncertainty (Zhu and Laptev, 2017).

According to the empirical rule, for data that is distributed normally, about 95% of the data fall within two standard deviations on each side of the mean (see Figure 2.3) (Ozdemir, 2016). In the thesis, we assume our data are normally distributed.

The z-score, also called standard score, is the number of standard deviations from the mean a data point is (Ott and Longnecker, 2008):

$$z = \frac{y - \mu}{\sigma} \tag{2.4}$$

where μ is the mean, σ is the population standard deviation and y is the measurement (Ott and Longnecker, 2008). For example, if a data point z-score less than 2, then it fall into the 95% prediction interval.

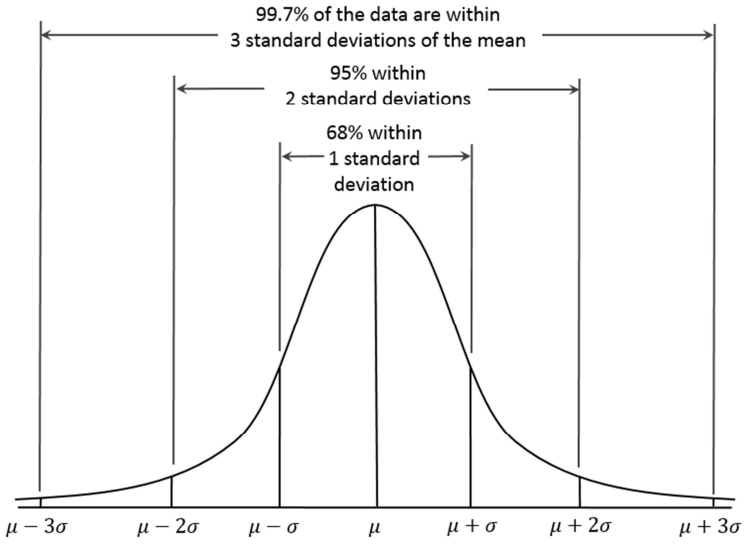


Figure 2.3: the empirical rule (Ozdemir, 2016)

2.4 Tools

2.4.1 Python

Python (Python Software Foundation, 2019) is a popular language used for machine learning. In the thesis, the implementation is in Python.

2.4.2 Keras

We use Keras to implement the machine learning models. Keras is a deep-learning framework for Python that provides a good way to define and train deep-learning models (Chollet, 2017). Initially, Keras was developed for enabling fast experimentation (Chollet, 2017). Its user-friendly API makes it easy to quickly prototype deep-learning models.

Keras can be freely used in commercial projects since it is distributed under the permissive MIT license (Chollet, 2017).

Keras is a model-level library, which means it provide high-level building blocks for developing deep-learning models (Chollet, 2017). It relies on a specialized, optimized tensor library to handle low-level operations, which serves as the backend engine of Keras (Chollet, 2017). Several different backend engines can be plugged seamlessly into Keras, such as TensorFlow backend, Microsoft Cognitive Toolkit (CNTK) and Theano backend (Chollet, 2017).

2.4.3 Pandas

Pandas (Wes McKinney, 2018) is a machine learning library for data manipulation and analysis. We use Pandas to preprocess the data.

2.4.4 Matplotlib

We used Matplotlib library (Hunter, 2007) to visualize the data. It helped us know the data and find important trends.

2.4.5 Scikit-Learn

Scikit-Learn (Pedregosa et al., 2011) is an open source library. Neural network are sensitive to unscaled data (Hochreiter and Schmidhuber, 1997), therefore we use `sklearn.preprocessing.StandardScaler` to transform our data such that its distribution will have a mean value 0 and standard deviation of 1 (Sammut and Webb, 2011).

We also use `sklearn.metrics.mean_absolute_error`. Mean Absolute Error (MSE) of a machine learning model is the mean of the absolute values of the individual prediction errors on over all data points in the test set (Sammut and Webb, 2011). Individual prediction error is the difference between the true value and the predicted value for the data point (Sammut and Webb, 2011).

Chapter 3
Approach

In this chapter, we will give a detailed description of the datasets given by Sinch. We will also explain the three models we use.

3.1 Terms

Before we talk about the datasets, first we define some terms that we will use. Here is the definition for the following terms:

- Client: The client company which sends messages to its customers mobile devices through Sinch. For a client to conduct business with Sinch, the client must have an account.
- Operator: A provider of the wireless communication services. The end user's mobile device always belongs to one of the operators.
- Count: The message traffic volume, that is, the number of messages.
- Successful Delivery Rate (SDR): The percentage of messages that were actually delivered to the end user's mobile devices, calculated by dividing the number of messages that have been successfully delivered to the end user's mobile devices by the total number of messages sent. The reason that a message is not delivered successfully could be bad routing setup or supplier connections failure.

3.2 Dataset

For each client company, we got three CSV files: the daily count file, the hourly successfully delivered count file, and the hourly total count file.

Table 3.1: The daily count file for a client company, to protect client company's privacy, the operator and count is not the real operator and count of the client company. It is just an example to give reader a sense how the data was given

Index	Date	Operator	Count
1	2013-05-06	1	200000
2	2013-05-06	2	100000
3	2013-05-06	3	100000
4	2013-05-07	1	250000
5	2013-05-07	2	120000
6	2013-05-07	3	90000
7	2013-05-08	1	210000
8	2013-05-08	2	110000
9	2013-05-08	3	110000
...	...	...	...
6883	2019-08-17	1	120000
6884	2019-08-17	2	220000
6885	2019-08-17	3	150000
6886	2019-08-18	1	210000
6887	2019-08-18	2	170000
6888	2019-08-18	3	150000

The daily count file has three columns, namely, Date, Operator and Count. An example of a daily count file is shown in Table 3.1. The dataset consists of all the data that have been collected from May 6,2013 to August 18, 2019. The time interval is 1 day. Each operator is represented by a unique integer ID. The count is the number of messages sent by this operator for this client during this day.

The hourly successfully delivered count file has three columns, namely, date time, operator and successful-count (see Table 3.2). The dataset consists of all the data that have been collected from October 3, 2017 to August 18, 2019. The time interval is 1 hour. Each operator is represented by a unique integer ID. The successful-count is the number of messages sent by this operator for this client during an hour starting from the datetime stated and were successfully delivered.

The hourly total count file (see Table 3.3) is similar to the hourly successfully delivered count file. The columns about datetime and operator are exactly the same as the hourly successfully delivered count file. The only difference is that the the column about successful-count is replaced by count, which indicate the total number of messages sent by this operator for this client during an hour starting from the time stated.

3.3 Models

We have implemented three models and use them for the experiments. Model I is a multi layer regression model, while both model II and model III include an encoder-decoder which

Table 3.2: The hourly successfully delivered count file for a client company, to protect client company's privacy, the operator and successful-count is not the real operator and successful-count of the client company. It is just an example to give reader a sense how the data was given

Index	datetime	Operator	successful-count
1	2017-10-03 00:00:00	1	9900
2	2017-10-03 00:00:00	2	5000
3	2017-10-03 00:00:00	3	4820
4	2017-10-03 01:00:00	1	7800
5	2017-10-03 01:00:00	2	6000
6	2017-10-03 01:00:00	3	4500
7	2017-10-03 02:00:00	1	10000
8	2017-10-03 02:00:00	2	4900
9	2017-10-03 02:00:00	3	4960
…	…	…	…
16723	2019-08-18 22:00:00	1	6000
16724	2019-08-18 22:00:00	2	11000
16725	2019-08-18 22:00:00	3	7500
16726	2019-08-18 23:00:00	1	10000
16727	2019-08-18 23:00:00	2	6000
16728	2019-08-18 23:00:00	3	7000

is implemented by two-layer LSTM cells.

3.3.1 Model I

Model I is a regression model which has three fully connected layers, with 128, 64, 16 hidden units, respectively (see Figure 3.1). This architecture is chosen because the supervisors recommended. The input is a sequence of values, daily or hourly.The output could be the value of next day or next hour.

```
### DEFINE REGRESSION MODEL ###
inputs = Input(shape=(X_train.shape[1], X_train.shape[2]))
flat = Flatten()(inputs)
dense1 = Dense(128)(flat)
dense2 = Dense(64)(dense1)
dense3 = Dropout(0.3)(dense2)
dense4 = Dense(16)(dense3)
dense5 = Dropout(0.3)(dense4)
out = Dense(1)(dense5)

model = Model(inputs, out)
model.compile(loss='mse', optimizer='adam', metrics=['mse'])

### FIT MODEL ###
history = model.fit(X_train, y_train, epochs=2, batch_size=128)
```

Table 3.3: The hourly total count file for a client company, to protect client company's privacy, the operator and count is not the real operator and count of the client company. It is just an example to give reader a sense how the data was given

Index	datetime	Operator	count
1	2017-10-03 00:00:00	1	10000
2	2017-10-03 00:00:00	2	5000
3	2017-10-03 00:00:00	3	5000
4	2017-10-03 01:00:00	1	8000
5	2017-10-03 01:00:00	2	6000
6	2017-10-03 01:00:00	3	6000
7	2017-10-03 02:00:00	1	10000
8	2017-10-03 02:00:00	2	5000
9	2017-10-03 02:00:00	3	5000
...	...	...	...
16723	2019-08-18 22:00:00	1	6000
16724	2019-08-18 22:00:00	2	11000
16725	2019-08-18 22:00:00	3	8000
16726	2019-08-18 23:00:00	1	10000
16727	2019-08-18 23:00:00	2	8000
16728	2019-08-18 23:00:00	3	7000

3.3.2 Model II

Zhu and Laptev (2017) proposed the following neural network architecture for time series prediction (see Figure 3.2). Given an input time series, the encoder constructs the learned embedding, and concatenate it with external features, and then feed the final input to the final prediction network (Zhu and Laptev, 2017).

The neural network can be divided into two components:

- an encoder-decoder framework that is used to capture the inherent pattern in the time series that is learned during pre-training (Zhu and Laptev, 2017)
- a prediction network that takes the learned embedding from encoder-decoder and potential external features as input (Zhu and Laptev, 2017)

We adopted this architecture. The input and output is the same as model I. The external features are the features that relate to the target. For example, Zhu and Laptev (2017) use previous 28 days daily completed trips to predict the completed trips of the upcoming day, one of the external features could be the temperature of the upcoming day. Since we do not have any external features in our dataset, we will not concatenate any external features.

Encoder-Decoder

We introduced an encoder-decoder in model II and model III.

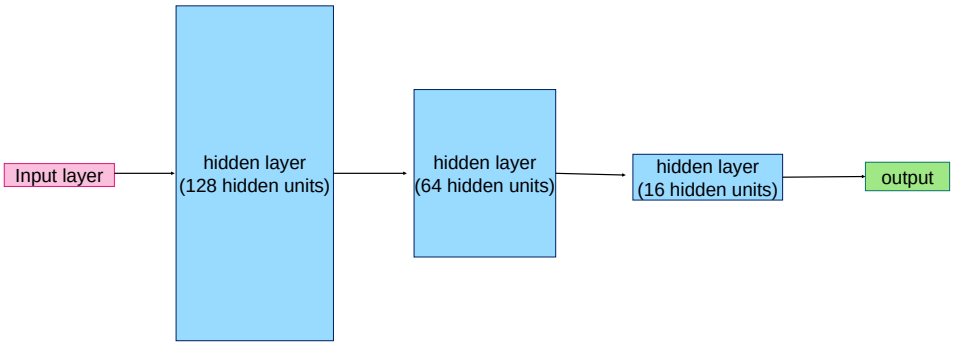


Figure 3.1: Model I

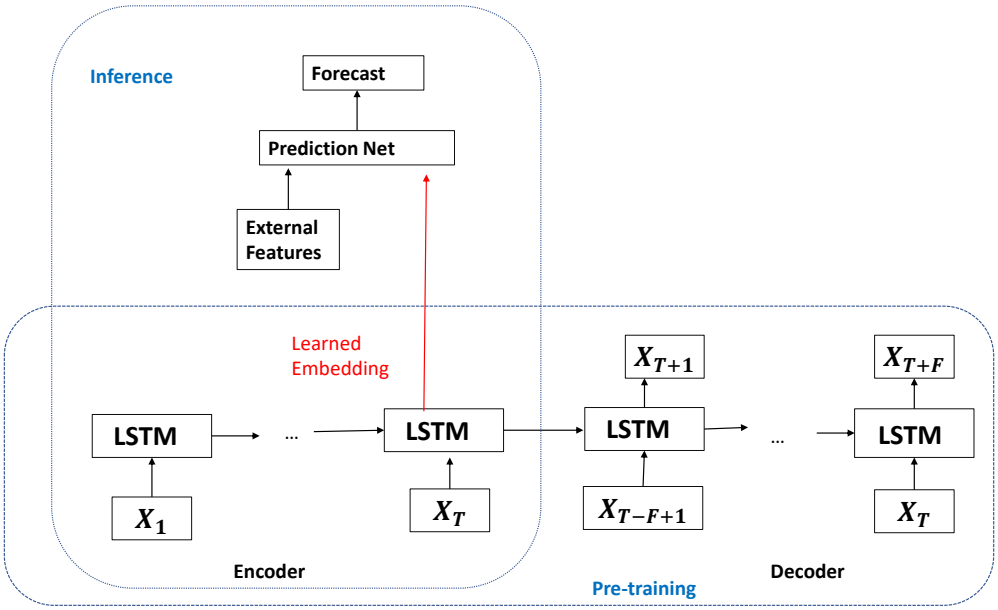


Figure 3.2: Model II. After Zhu and Laptev (2017)

The encoder is trained to extract the representative features from a time series and the decoder is used to reconstruct the time series from the encoded space (Zhu and Laptev, 2017). During pre-training, we fit the encoder that can extract representative embeddings from a time series, which could provide useful features for prediction (Zhu and Laptev, 2017).

When predicting the test set, the quality of encoding of every sample will give information about how close it is to the training set.

Following Cerliani (2019), we implemented the encoder-decoder framework using two-layer LSTM cells, with 128 and 32 hidden states, respectively (Cerliani, 2019).

Here is a code snippet for the Encoder-Decoder structure (Cerliani, 2019).

```
### DEFINE LSTM AUTOENCODER ###
inputs = Input(shape=(sequence_length, 1))
encoded = LSTM(128, return_sequences=True, dropout=0.3)(inputs,
    training=True)
decoded = LSTM(32, return_sequences=True, dropout=0.3)(encoded,
    training=True)
out = TimeDistributed(Dense(1))(decoded)

autoencoder = Model(inputs, out)
autoencoder.compile(optimizer='adam', loss='mse', metrics=['mse'])

### TRAIN AUTOENCODER ###
autoencoder.fit(X_train, X_train, batch_size=64, epochs=12, shuffle
    =True)

### ENCODE INPUTS ###
encoder = Model(inputs, encoded)
XX = encoder.predict(X)
```

Prediction Network

The encoder-decoder could be seen as an intelligent feature-extraction black box after it is pre-trained. We extract the last LSTM cell states of the encoder as learned embedding (Zhu and Laptev, 2017). Using the learned embedding as features, we can train the prediction network to do forecasting (Zhu and Laptev, 2017). External features can be concatenated to the embedding vector and passed together to the final prediction network (Zhu and Laptev, 2017). Since external features are not available in our datasets, we will just pass the embedding vector to the final prediction network (Zhu and Laptev, 2017).

We use the same regression model as model I as the prediction network.

Inference

After we trained the full model, we only need to involve the encoder and the prediction network for the inference (Zhu and Laptev, 2017).

The model uncertainty η can be estimated by applying MC dropout to both the encoder and the prediction network (Zhu and Laptev, 2017). We can construct an approximate α-level prediction interval by $[\hat{y}^* - Z_{\alpha/2}\eta, \hat{y}^* + Z_{\alpha/2}\eta]$ where $Z_{\alpha/2}$ represents the upper $\alpha/2$ quantile of a standard normal (Zhu and Laptev, 2017).

According to Zhu and Laptev (2017), the uncertainty estimation is relatively stable across a range of the drop out probability, p, so we could just choose the one that achieves the best

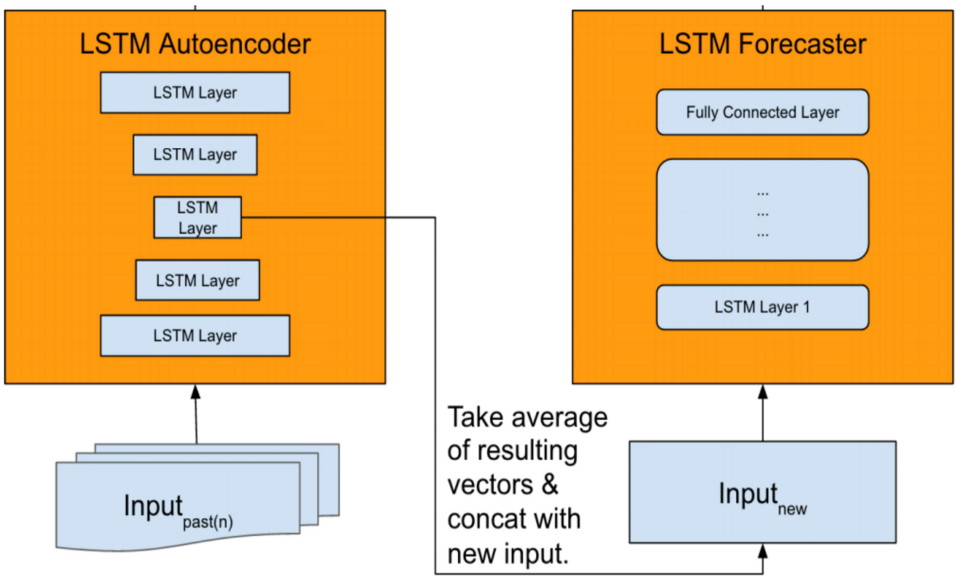


Figure 3.3: Model proposed by Laptev et al. (2017)

performance. The standard error of the estimated prediction is proportional to $1/\sqrt{B}$, where B is the number of iterations (Zhu and Laptev, 2017). Zhu and Laptev (2017) pointed out that to achieve a stable estimation, a few hundreds of iterations are enough. In our experiments, we find that 100 iterations is enough to reach a stable estimation for our datasets.

3.3.3 Model III

Laptev et al. (2017) proposed another model architecture (see Figure 3.3), which is quite similar to what Zhu and Laptev (2017) proposed.

The model first primes the network by auto feature extraction, which is important to catch complex time series dynamics (Laptev et al., 2017). This is different from the standard feature extraction methods where the features are manually derived (Laptev et al., 2017). Then we use an ensemble technique (e.g., averaging) to aggregate the feature vectors (Laptev et al., 2017). The new input are then concatenated with the final vector before they are fed to LSTM Forecaster to predict the results. Laptev et al. (2017) found that it produced better results to have a separate auto-encoder module.

Following Cerliani (2019), we implemented the model in the following way. The input and output is the same as model I.

Encoder-Decoder

LSTM encoder-decoder first trained on the training data, then we only use the encoder as a feature creator (Cerliani, 2019). The encoder could provide a good feature extraction that is useful for later (Laptev et al., 2017).

We implement the Encoder-Decoder the same way as we did for model II.

Prediction

We implement the prediction part with a LSTM model instead of a regression model (Cerliani, 2019). This is the only difference between model II and model III. The LSTM model consists two-layer stacked LSTM cells, with 128 and 32 hidden states, respectively, followed by a fully connected layer for the final output. The following is a code snippet for the prediction model (Cerliani, 2019).

```
###DEFINE FORECASTER ###
inputs3 = Input(shape=(XX[:train.shape[0]].shape[1], XX[:train.
    shape[0]].shape[2]))
lstm1 = LSTM(128, return_sequences=True, dropout=0.3)(inputs3,
    training=True)
lstm1 = LSTM(32, return_sequences=False, dropout=0.3)(lstm1,
    training=True)
dense1 = Dense(50)(lstm1)
out3 = Dense(1)(dense1)

model3 = Model(inputs3, out3)
model3.compile(loss='mse', optimizer='adam', metrics=['mse'])

### FIT FORECASTER ###
history = model3.fit(X_train1, y_train1, epochs=15, batch_size=256,
     verbose=2, shuffle=True)
```

Chapter 4
Evaluation

This chapter will first present the experiment results one by one, and then we will discuss the results.

4.1 Results

In this chapter, we present the results in figures and tables to provide a good overview. The discussion will provide our thoughts on the experiment results.

4.1.1 Prediction for a client's traffic next day

First we preprocessed the daily count file by grouping the Count by Date (see Table 4.1 below). We divide the data into training sets and test sets, with an 80-20 split, which is a normal practice.

We constructed the samples by using a sliding window with step size one, where each sliding window contains the previous 20 days as input, and intended to forecast the upcoming day (Zhu and Laptev, 2017). In fact, we have tried different window size and find that 20 days works best among them. Following Cerliani (2019), we transform the raw data using scikit learn standard scaler. At test time, we reverted the transformations to obtain predictions at the original scale.

Then we apply model I, model II and model III respectively to the data for 100 times. As we explained before, we can achieve a stable estimation for our datasets with 100 iterations. Figure 4.1 shows the training data used for this experiment. Figures 4.2, 4.3 and 4.4 visualize the true values and our predictions of a client's traffic next day based on its traffic during last 20 days. True values are shown with the yellow line, and predictions are shown with the blue line, where the 95% prediction band is shown as the light blue area.

Table 4.2 show the mean of the mean absolute error for this 100 times prediction, as well as how many percentage of the ground truth values lie outside the 95% prediction interval.

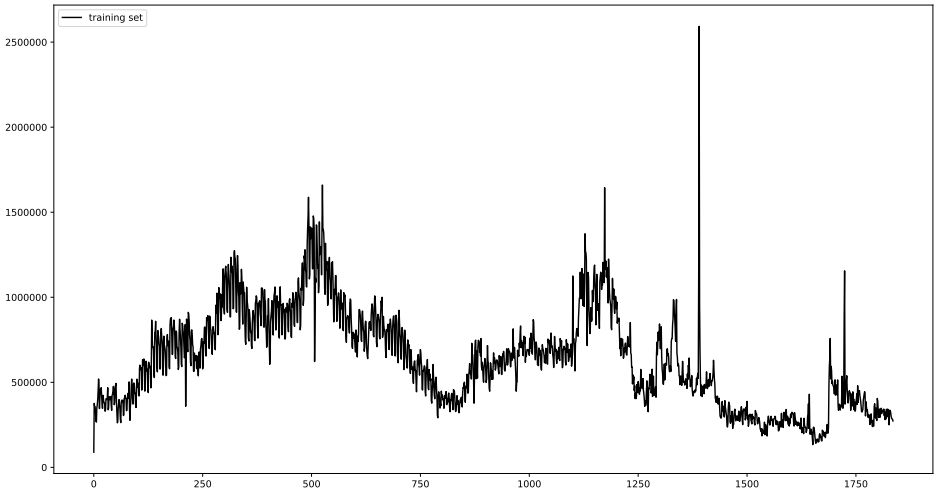


Figure 4.1: Training data for predicting for a client's traffic next day
The horizontal (x) axis represent time in days, and the vertical (y) axis represent the count in units

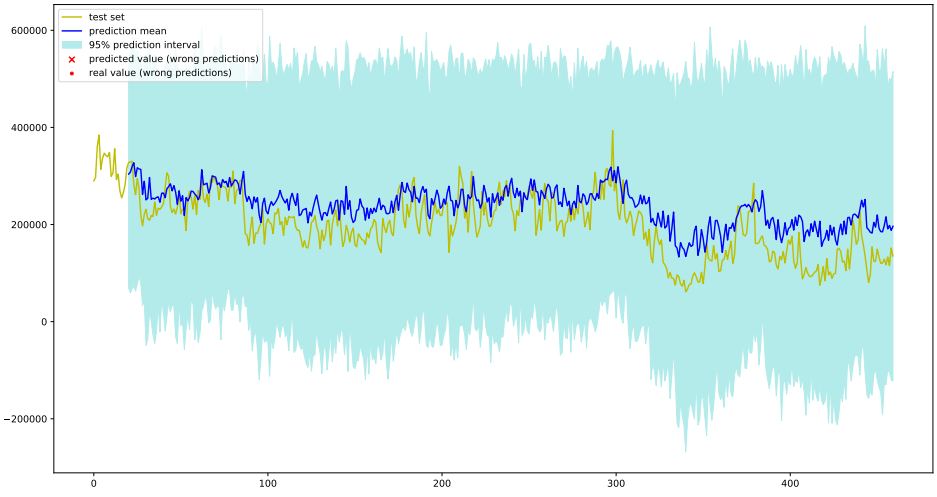


Figure 4.2: Prediction for a client's traffic next day model I
The horizontal (x) axis represent time in days, and the vertical (y) axis represent the count in units

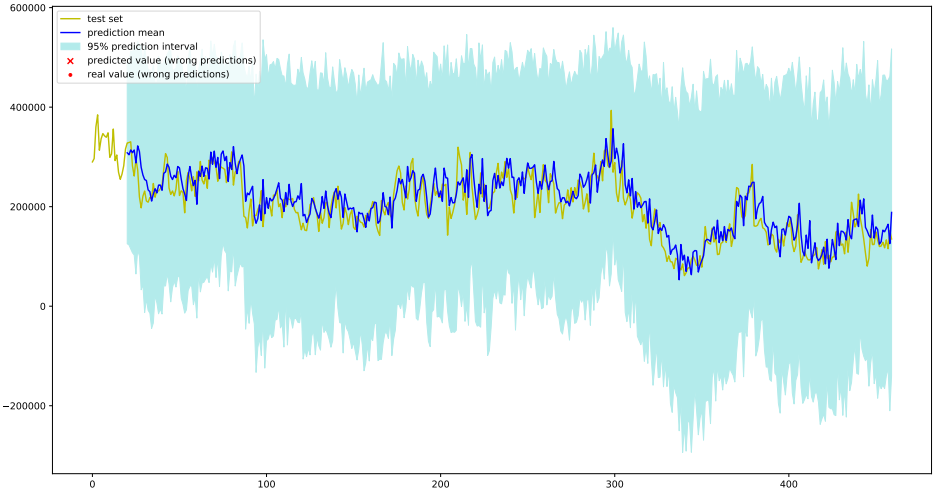


Figure 4.3: Prediction for a client's traffic next day model II
The horizontal (x) axis represent time in days, and the vertical (y) axis represent the count in units

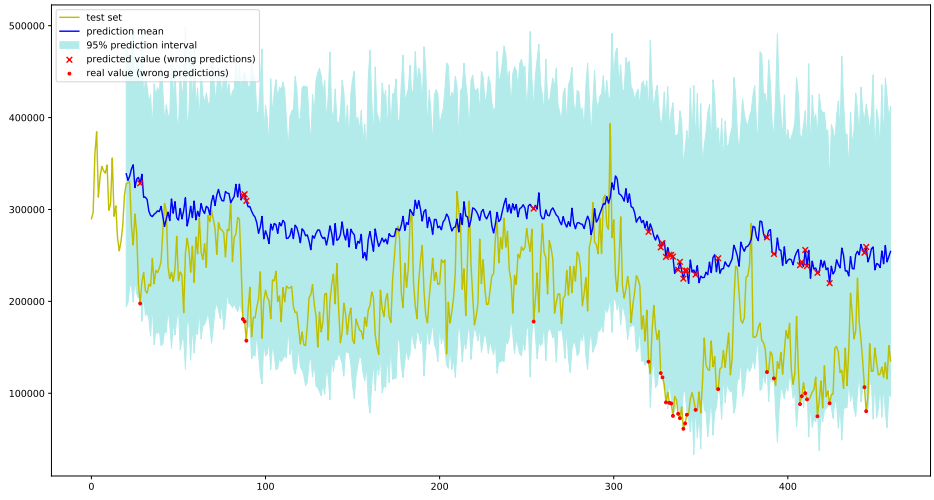


Figure 4.4: Prediction for a client's traffic next day model III
The horizontal (x) axis represent time in days, and the vertical (y) axis represent the count in units

Table 4.1: The daily count file for a client company has been preprocessed by aggregating the count of different operators on the same date. This new file is used for Prediction for a client's traffic next day

Index	Date	Count
1	2013-05-06	400000
2	2013-05-07	460000
3	2013-05-08	430000
...	...	...
2295	2019-08-17	490000
2296	2019-08-18	530000

Table 4.2: Prediction for a client's traffic next day

Model	Model I	Model II	Model III
Mean error	0.3014	0.09840	0.2412
Percentage	0%	0%	2.273%

4.1.2 Prediction for a client's traffic through an operator next day

We preprocess the daily count file by filtering the entries by operators. For example, Table 4.3 shows the preprocessed data for one operator.

Then we did a 80-20 split on the data and created samples and target the same way as in Sect. 4.1.1, apply model I, model II and model III respectively to the data each for 100 times.

Figure 4.5 shows the training data for this experiment. Figures 4.6, 4.7 and 4.8 show the prediction results of a client's traffic through this operator next day based on its traffic during last 20 days. True values are shown with the yellow line, and predictions are shown with the blue line, where the 95% prediction band is shown as the light blue area.

Table 4.4 show the mean of the mean absolute error for this 100 times prediction, as well as how many percentage of the ground truth values lie outside the 95% prediction interval.

4.1.3 Prediction for a client's traffic next month

We use the preprocessed data from Table 4.1 and split the first 80% of it into a training set, and the last 20% into a test set. In other words, the first 80% of the data is training set and the last 20% of the data is test set.

We constructed the samples by using a sliding window with step size one, where each sliding window contains the previous 20 days as input, and intended to forecast the sum of the count of the upcoming 30 days (Zhu and Laptev, 2017). We tried to use different window size for the prediction, and find that 20 days is the best choice. Following Cerliani (2019), the raw data are transformed using scikit learn standard scaler (Zhu and Laptev, 2017). At test time, we reverted the transformations to obtain predictions at the original scale.

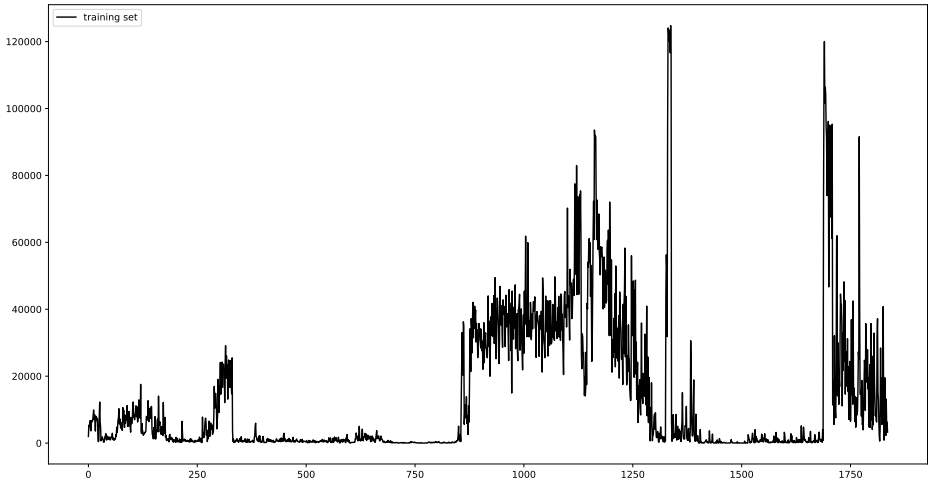


Figure 4.5: Training data for predicting a client's traffic through an operator next day
The horizontal (x) axis represent time in days, and the vertical (y) axis represent the count in units

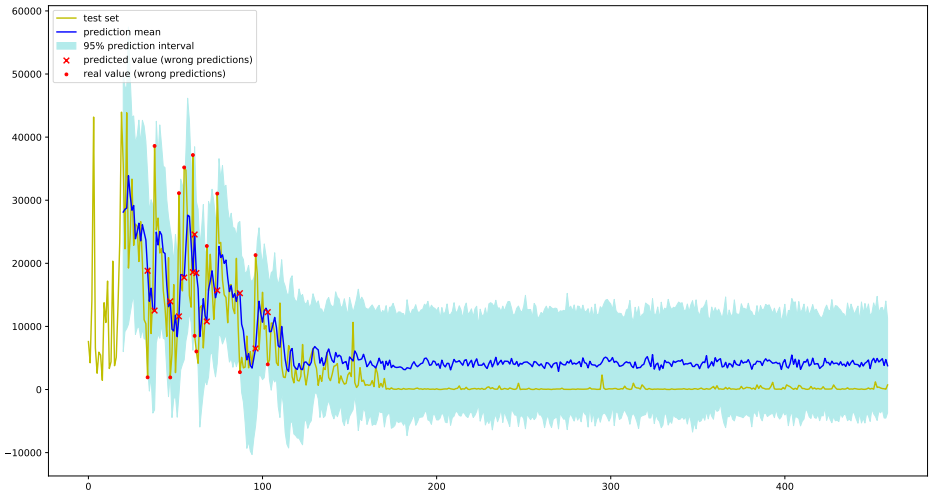


Figure 4.6: Prediction for a client's traffic through an operator next day model I
The horizontal (x) axis represent time in days, and the vertical (y) axis represent the count in units

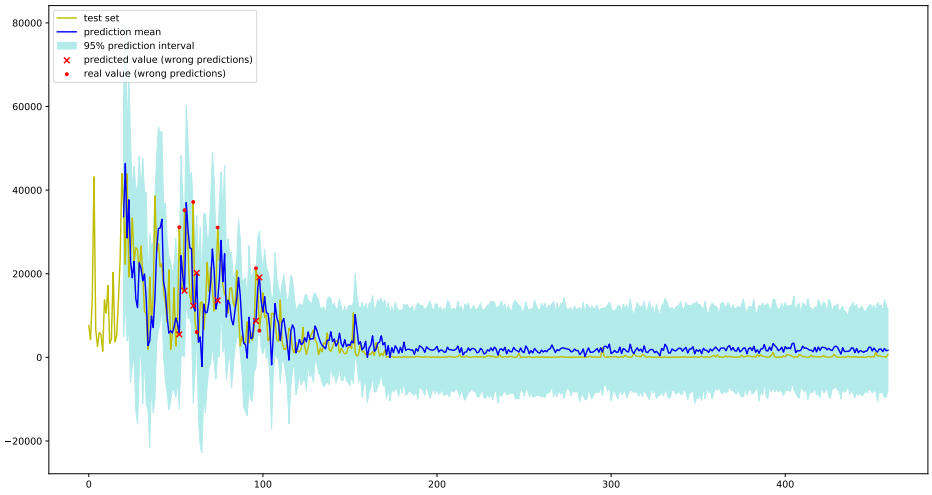


Figure 4.7: Prediction for a client's traffic through an operator next day model II
The horizontal (x) axis represent time in days, and the vertical (y) axis represent the count in units

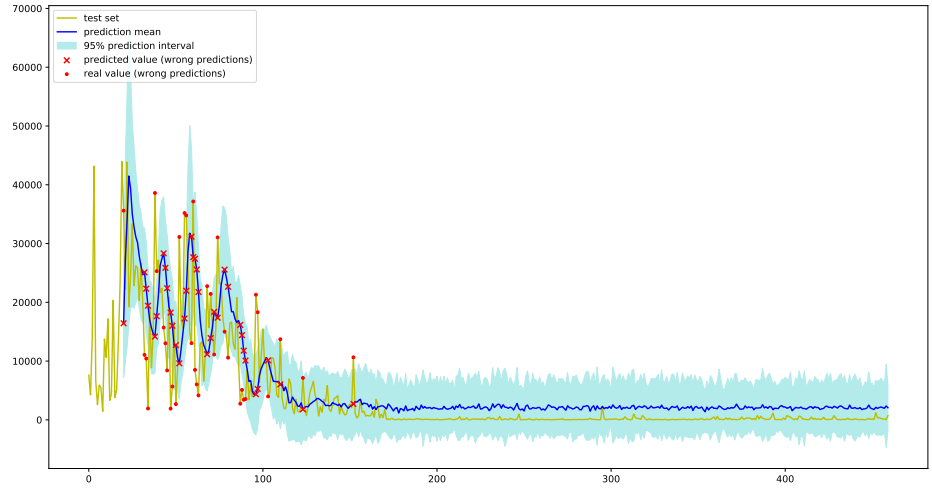


Figure 4.8: Prediction for a client's traffic through an operator next day model III
The horizontal (x) axis represent time in days, and the vertical (y) axis represent the count in units

Table 4.3: The daily count file for a client company preprocessed to be used for prediction for a client's traffic through an operator next day

Index	Date	Operator	Count
1	2013-05-06	1	200000
2	2013-05-07	1	250000
3	2013-05-08	1	210000
...	...	...	...
2295	2019-08-17	1	120000
2296	2019-08-18	1	210000

Table 4.4: Prediction for a client's traffic through an operator next day

Model	Model I	Model II	Model III
Mean error	0.2256	0.1367	0.1888
Percentage	2.954%	2.727%	7.727%

Then we apply models I, II and III respectively to the data each for 100 times. Figures 4.9, 4.10 and 4.11 show the prediction results of a client's traffic next month based on its traffic during last 20 days. True values are shown with the yellow line, and predictions are shown with the blue line, where the 95% prediction band is shown as the light blue area.

Table 4.5 shows the mean of the mean absolute error for this 100 times prediction, as well as how many percentage of the ground truth values lie outside the 95% prediction interval.

4.1.4 Prediction for a client's successful delivery rate next hour

First we use pandas to preprocess the hourly successfully delivered count file and the hourly total count file to calculate the SDR for each datetime entry (see Table 4.6 below).

We split the dataset in a 80–20 ratio, meaning the first 80% of the data is used for training and the rest of the data is used for testing. This is a common practice. We constructed the samples by using a sliding window with step size one, where each sliding window contains the previous 20 hours as input, and intended to forecast the upcoming hour (Zhu and Laptev, 2017). We actually tried different window sizes, and find 20 hours produce the best prediction results.

Then we apply models I, II and III respectively to the data for 100 times. Figure 4.12 shows the training data for this experiment. Figures 4.13, 4.14 and 4.15 show the prediction results of a client's SDR next hour based on its SDR during last 20 hours. True values are shown with the yellow line, and predictions are shown with the blue line, where the 95% prediction band is shown as the light blue area.

Table 4.7 shows the results the same way as before.

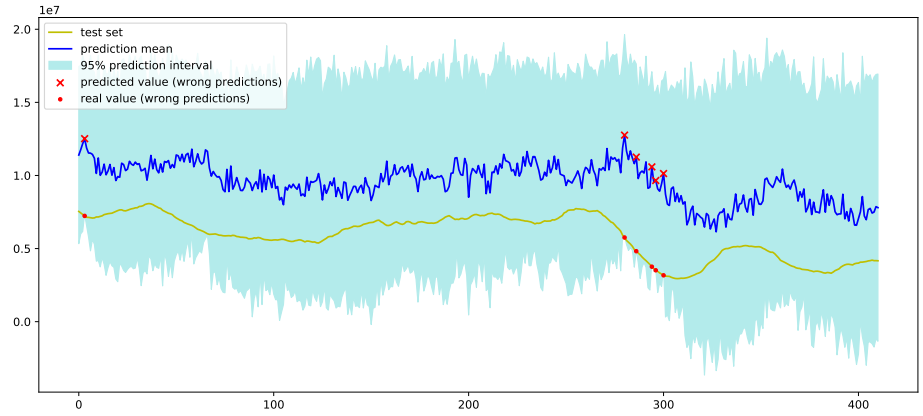


Figure 4.9: Prediction for a client's traffic next month model I
The horizontal (x) axis represent time in months, and the vertical (y) axis represent the count in units

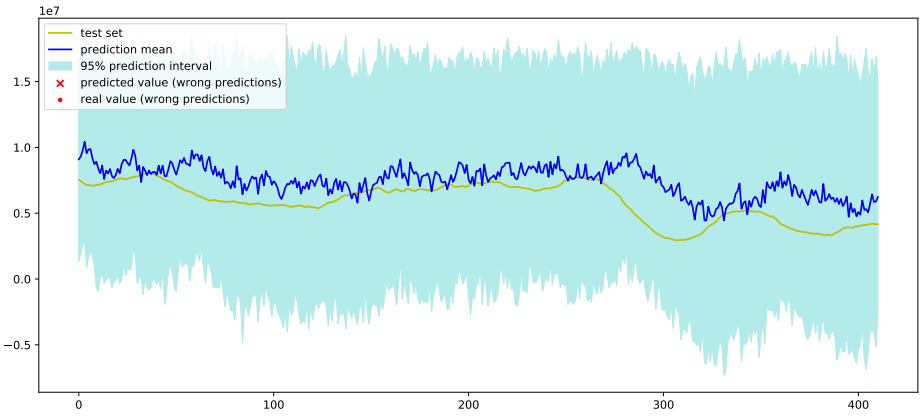


Figure 4.10: Prediction for a client's traffic next month model II
The horizontal (x) axis represent time in months, and the vertical (y) axis represent the count in units

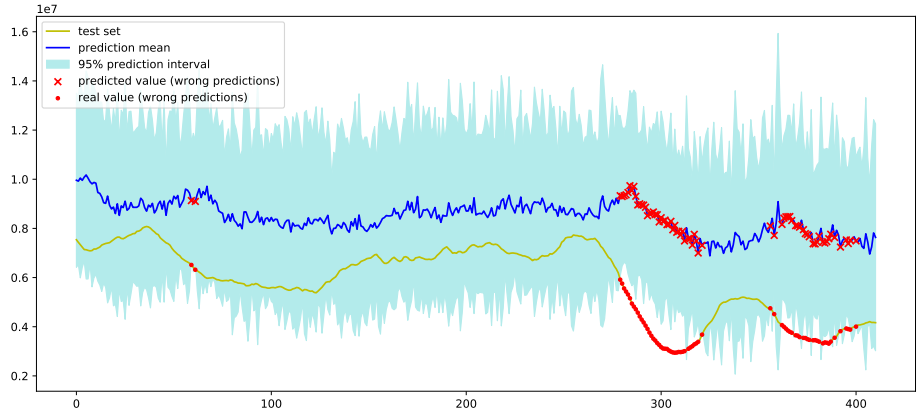


Figure 4.11: Prediction for a client's traffic next month model III The horizontal (x) axis represent time in months, and the vertical (y) axis represent the count in units

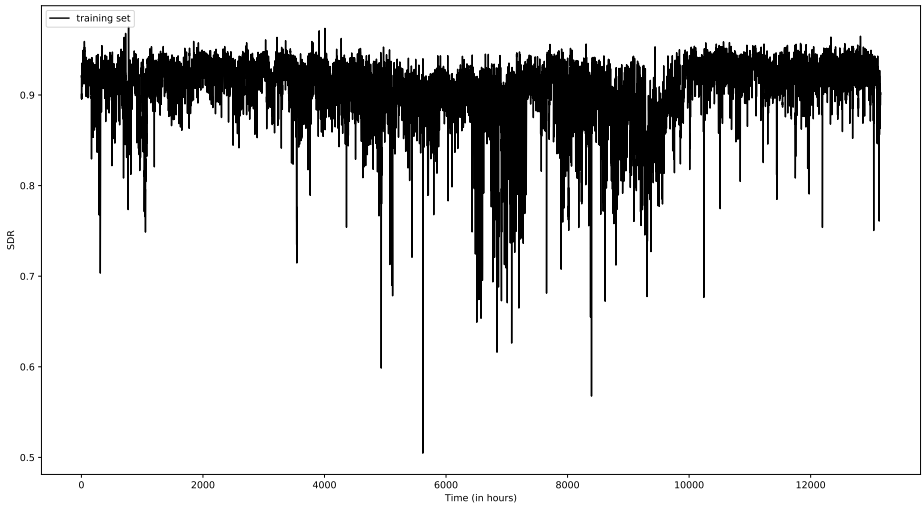


Figure 4.12: Training data for predicting a client's successful delivery rate next hour

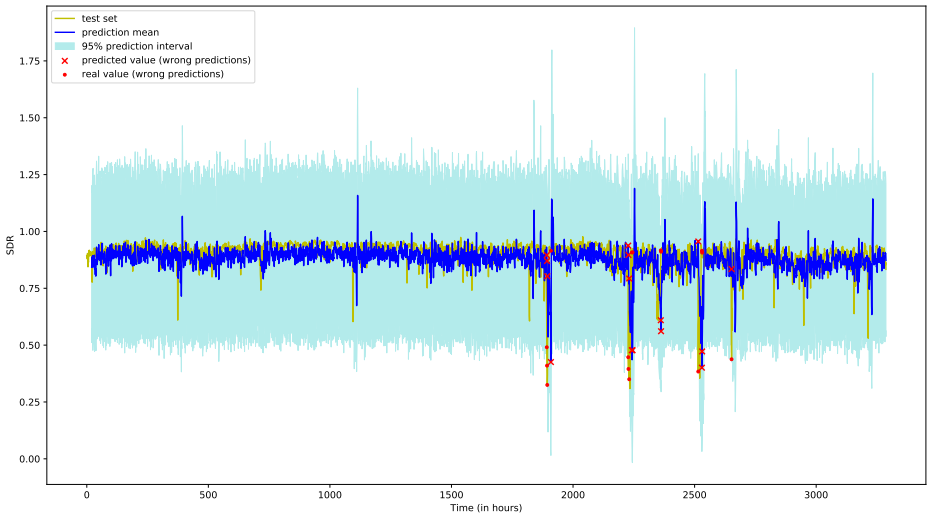


Figure 4.13: Prediction for a client's successful delivery rate next hour model I

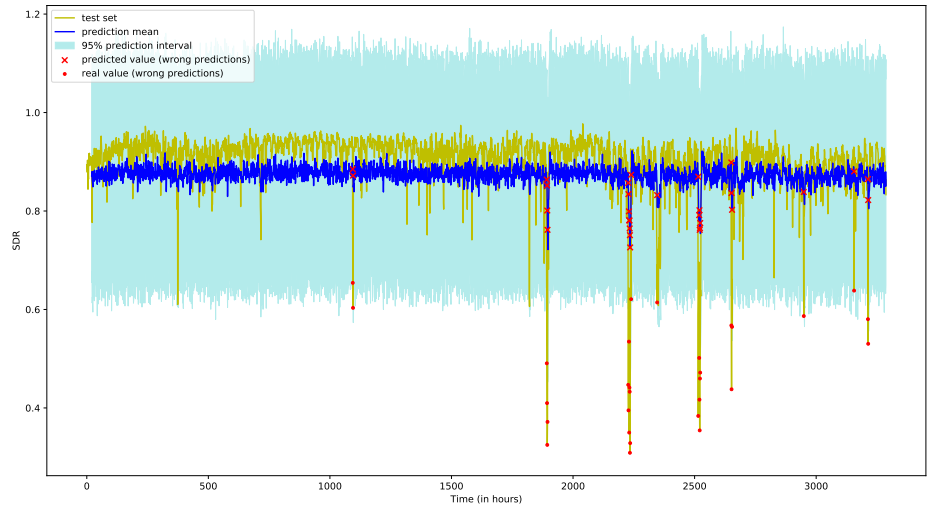


Figure 4.14: Prediction for a client's successful delivery rate next hour model II

Table 4.5: Prediction for a client's traffic next month

Model	Model I	Model II	Model III
Mean error	0.4705	0.2037	0.3293
Percentage	1.460%	0%	17.76%

Table 4.6: The hourly successfully delivered Rate for a client company

Index	datetime	successful-count	Count	SDR
1	2017-10-03 00:00:00	19720	20000	0.986
2	2017-10-03 01:00:00	18300	20000	0.915
3	2017-10-03 02:00:00	19860	20000	0.993
...	...	...	...	...
5574	2019-08-18 22:00:00	24500	25000	0.98
5575	2019-08-18 23:00:00	23000	25000	0.92

4.1.5 Prediction for a client's successful delivery rate through an operator next hour

First we use pandas to preprocess the hourly successfully delivered count file and the hourly total count file to filter the successful-count and count of this operator for each datatime entry, then we merge the dataframe and calculate the SDR for each datetime entry by dividing its successful-count with its count (see Table 4.8 below).

Then we created the samples the same way as in Sect. 4.1.4 and applied models I, II and III respectively to the data for 100 times. Figure 4.16 shows the training data for this experiment. Figures 4.17, 4.18 and 4.19 show the prediction results of a client's SDR through an operator next hour based on its SDR through the same operator during last 20 hours. True values are shown with the yellow line, and predictions are shown with the blue line, where the 95% prediction band is shown as the light blue area.

Table 4.9 shows the results the same way as before.

4.2 Discussion

We can see from Table 4.2, Figures 4.2, 4.3 and 4.4 that when predict for a client's traffic next day, model II perform better than model III due to its smaller mean absolute errors. We can also see that all the true values fall within 95% prediction interval of model I and model II. While model III could detect some anomaly points which fall out side of its 95% prediction interval.

We can see the similar situation from Table 4.4, Figures 4.6, 4.7 and 4.8 that Model II is better than Model III when predict a client's traffic through an operator next day.

According to Table 4.5, Figures 4.9, 4.10 and 4.11 we can see the similar trend when predict a client's traffic next month. Comparing with Table 4.2, we can tell that all the models

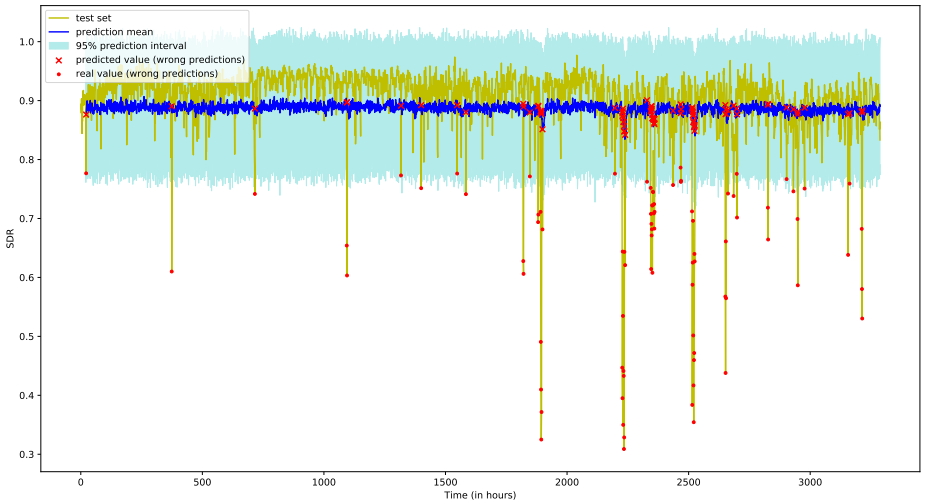


Figure 4.15: Prediction for a client's successful delivery rate next hour model III

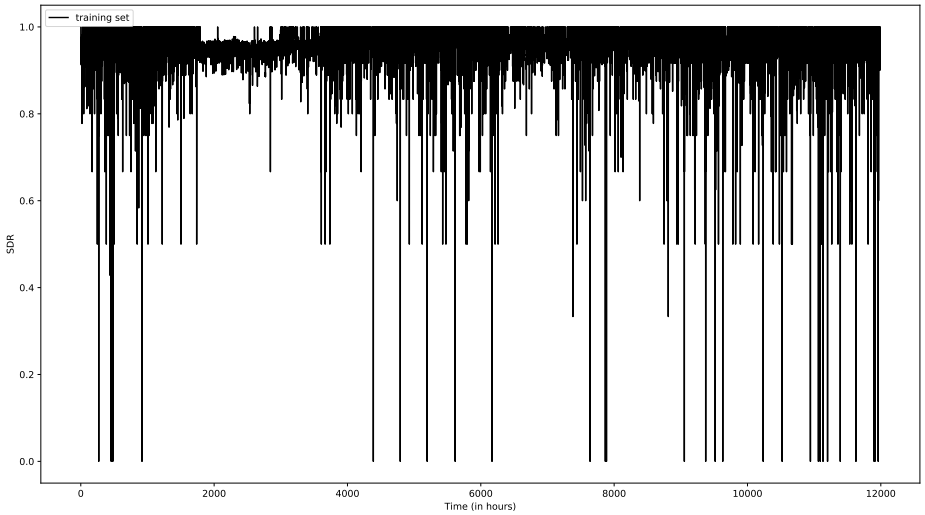


Figure 4.16: Training data for predicting a client's successful delivery rate through an operator next hour

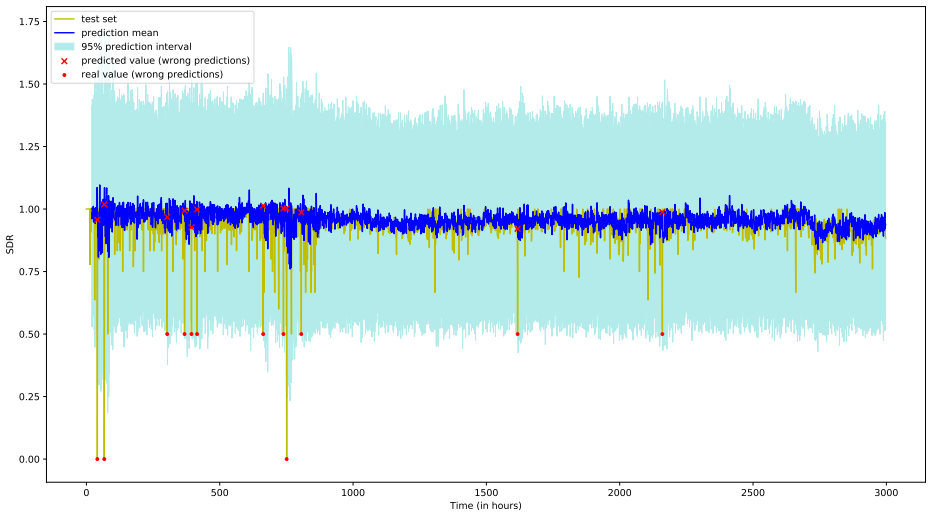


Figure 4.17: Prediction for a client's successful delivery rate through an operator next hour model I

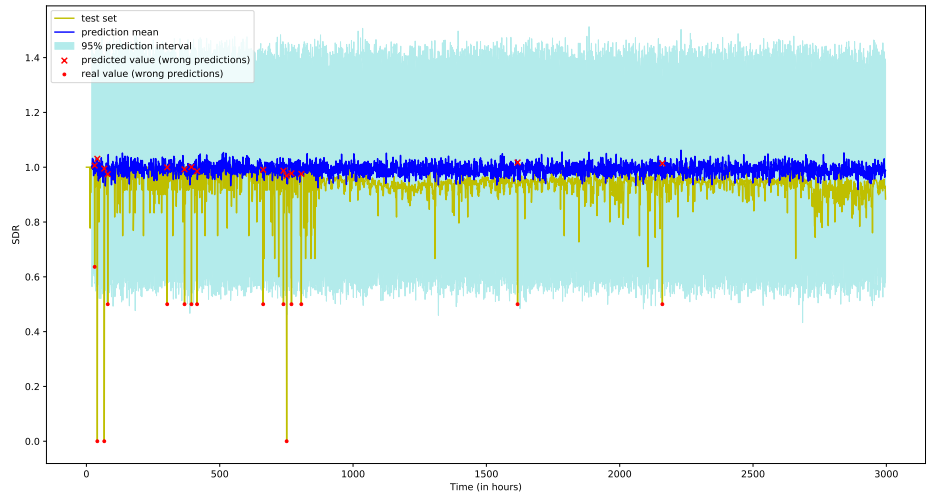


Figure 4.18: Prediction for a client's successful delivery rate through an operator next hour model II

Table 4.7: Prediction for a client's successful delivery rate next hour

Model	Model I	Model II	Model III
Mean error	0.05666	0.06955	0.08228
Percentage	0.3672%	0.5508%	3.274%

Table 4.8: The hourly successfully delivered Rate for a client company through an operator

Index	datetime	Operator	successful-count	Count	SDR
1	2017-10-03 00:00:00	1	9900	10000	0.99
2	2017-10-03 01:00:00	1	7800	8000	0.975
3	2017-10-03 02:00:00	1	10000	10000	1
...	...	...	...	...	...
5574	2019-08-18 22:00:00	1	6000	6000	1
5575	2019-08-18 23:00:00	1	10000	10000	1

perform better in predicting next day than next month since all its mean absolute error in Table 4.5 is higher than its correspond model in Table 4.2.

From Table 4.7, Figures 4.13, 4.14 and 4.15 we can see that model II is still a bit better than model III due to slightly lower mean absolute error. Model III has more true values outside the 95% prediction interval, which means it can detect more anomaly points. This is probably because it has a narrow prediction interval as we see from Figure 4.15. We can see the similar trend from Table 4.9, Figures 4.17, 4.18 and 4.19 when predict a client's successful delivery rate through an operator next hour.

We have explored different parameters for each model and the results presented here are the best result for each model we found so far. Based on the experiment results, we can see that adding a LSTM encoder-decoder for feature extraction is only slightly better than the regression model given the datasets from Sinch. In fact for some datasets, adding a LSTM encoder-decoder does not improve the prediction results. In general, when we use the encoder-decoder structure, a regression forecaster give better prediction results than a LSTM forecaster since the former usually has lower mean absolute error. However the latter usually could detect more anomaly points. But it depends on the datasets, sometimes, the latter could detect too many anomaly points and some of them are false anomaly.

We can also see that the models perform better in predict number of messages next day than next month since all the mean absolute error is smaller when predict next day than next month.

We also notice that more anomaly points are detected when predict traffic volume or SDR of a single operator of a client than the total traffic volume or SDR of this client. This is probably because the total traffic volume and SDR of this client is more stable than its individual operator.

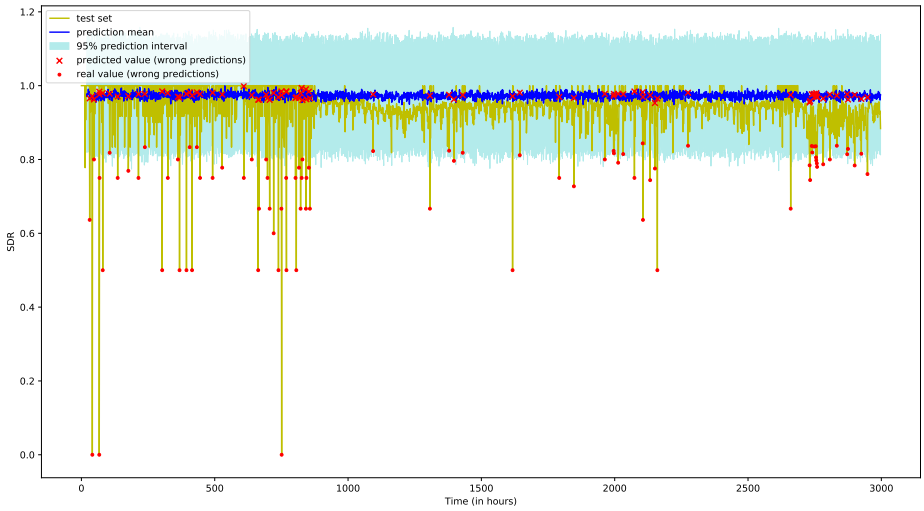


Figure 4.19: Prediction for a client's successful delivery rate through an operator next hour model III

Table 4.9: Prediction for a client's successful delivery rate through an operator next hour

Model	Model I	Model II	Model III
Mean error	0.05960	0.03738	0.08691
Percentage	0.4365%	0.5373%	4.298%

Chapter 5
Conclusion and future work

In the thesis we compare the performance of three different models for time series prediction. The first one is a regression model, the second one is a model with a LSTM encoder-decoder and a regression forecaster, the third one is a model with a LSTM encoder-decoder and a stacked LSTM forecaster.

We implemented these three models and apply them with datasets about daily and monthly mobile message traffic volume as well as hourly Successful Deliver Rate. The datasets are provided by Sinch, which is a telecommunication and cloud communications platform as a service company. We used the MC dropout technique to estimate uncertainty for the forecasting and detect anomaly.

Overall, the prediction results produced by the models are very good, since the mean absolute error is very low and most of the real values lie within 95% prediction interval. Currently Sinch does not have any method to predict traffic volume and SDR, and the company can not detect anomaly neither, so this thesis will fulfill its need. The architecture that includes a LSTM encoder decoder could improve the model performance on Sinch data, but not in all cases.

When the model already use a LSTM encoder decoder, a regression forecaster actually could outperform a stacked LSTM forecaster in prediction since the former usually has lower mean absolute errors. However the latter usually could detect more anomaly points, sometimes it could detect too many anomaly points and some of them are false anomaly.

The models can detect more anomaly points with a client's single operator's data than its all operator's data together. This is probably because a single operator's data is more fluctuated while a client's total data is more stable. The models also perform better on next day prediction than next month prediction.

Our future work will be centered around gross profit prediction using machine learning models. We will continue to use these models to predict daily average revenue per message and daily average cost per message and combine these to predict the gross profit. We will also explore to predict the gross profit in a longer time span, such as next eighteen months.

Bibliography

Assaad, M., Boné, R., and Cardot, H. (2008). A new boosting algorithm for improved time-series forecasting with recurrent neural networks. *Information Fusion*, 9(1):41 – 55. Special Issue on Applications of Ensemble Methods.

Cerliani, M. (2019). Extreme event forecasting with LSTM autoencoders. `https://towardsdatascience.com/extreme-event-forecasting-with-lstm-autoencoders-297492485037`.

Chatfield, C. (2000). *Time-Series Forecasting*. CRC Press.

Chollet, F. (2017). *Deep Learning with Python*. Manning.

Gal, Y. and Ghahramani, Z. (2015). A theoretically grounded application of dropout in recurrent neural networks.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1050–1059. JMLR.org.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. Adaptive Computation and Machine Learning series. MIT Press.

Graves, A. and Jaitly, N. (2014). Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1764–II–1772. JMLR.org.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

hoon Oh, D., Shah, Z., and Jang, G.-J. (2017). *Line-Break Prediction of Hanmun Text using Recurrent Neural Networks*. IEEE, Jeju, South Korea.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95.

Konar, A. and Bhattacharya, D. (2017). *Time-Series Prediction and Applications: A Machine Intelligence Approach*. Intelligent Systems Reference Library. Springer International Publishing.

Laptev, N., Yosinski, J., Li, L. E., and Smyl, S. (2017). Time-series extreme event forecasting with neural networks at Uber. In *Proceedings of the ICML 2017 Time Series Workshop*, Sydney.

Mohan, A. T. and Gaitonde, D. V. (2018). A deep learning based approach to reduced order modeling for turbulent flow control using lstm neural networks. *Working Paper*.

Mulder, W. D., Bethard, S., and Moens, M.-F. (2015). A survey on the application of recurrent neural networks to statistical language modeling. *Computer Speech & Language*, 30(1):61–98.

Ott, R. and Longnecker, M. (2008). *An Introduction to Statistical Methods and Data Analysis*. Available 2010 Titles Enhanced Web Assign Series. Cengage Learning.

Ozdemir, S. (2016). *Principles of Data Science*. Packt Publishing.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Python Software Foundation (2019). Python. www.python.org.

Sammut, C. and Webb, G. (2011). *Encyclopedia of Machine Learning*. Encyclopedia of Machine Learning. Springer US.

Tang, Y., Xu, J., Matsumoto, K., and Ono, C. (2016). *Sequence-to-Sequence Model with Attention for Time Series Classification*. IEEE, Barcelona, Spain.

Tim, H., Marcus, O., and William, R. (1996). Neural network models for time series forecasts. *Management Science*, 42(7):1082.

Wes McKinney (2018). Pandas. pandas.pydata.org.

Zhu, L. and Laptev, N. (2017). Deep and confident prediction for time series at Uber. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 103–110.

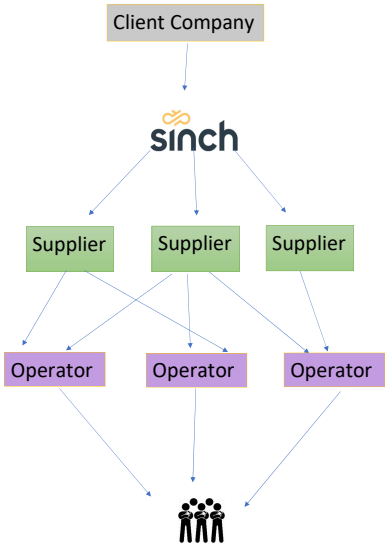
EXAMENSARBETE Time Series Prediction Using LSTM
STUDENT Linfeng Lu
HANDLEDARE Pierre Nugues (LTH)
EXAMINATOR Flavius Gruian (LTH)

Time Series Prediction Using LSTM

POPULÄRVETENSKAPLIG SAMMANFATTNING **Linfeng Lu**

Time series prediction is the use of a model to predict future values based on previously observed values. It has become more and more popular among businesses in the world. This thesis was done in collaboration with a telecommunication company to help the company to predict their mobile message traffic volume and Successful Delivery Rate. This could help the company to control their cost and product quality.

Time series prediction is about predicting the future based on the data we have about the past. It has already been widely used in financial institutions such banks, hedge funds, etc. Recently it became more and more common among all the businesses. The thesis is done in collaboration with a company called Sinch. The main business of Sinch is to deliver mobile messages for its client companies to their customers. The figure on the right shows the route of the messages. Sinch would like to predict their client company's daily traffic volume and hourly *successful delivery rate*.



Time series prediction could be carried out with different machine learning models. It become popular to use deep learning models to do the prediction.

In this thesis, we have applied three different machine learning models to a dataset provided by Sinch. The dataset contains daily and hourly mobile message traffic volume during the last years. After we preprocessed the data, we also get the hourly successful delivery rate. One of the models we used is a regression model. The two other models include a encoder-decoder.

The experiment results show that all the models perform well in traffic volume prediction and anomaly detection. The model architecture that includes a encoder-decoder could outperform the regression model but not in all cases, due to the difference of the datasets. The results also show that it is easier to predict next day than next month.