



Independent project (degree project), 15 credits, for the degree of  
Bachelor of Science with a major in Computer Science  
Spring Semester 2019  
Faculty of Natural Sciences

## Investigation of Event-Prediction in Time-Series Data

How to organize and process time-series data for event prediction?

Shameer Kumar Pradhan

**Author**

Shameer Kumar Pradhan

**Title**

Investigation of Event-Prediction in Time-Series Data – How to organize and process time-series data for event prediction?

**Supervisor**

Kamilla Klonowska, Ph.D. – Kristianstad University

David Jakobsson, Director of IT Software Development – Diaverum AB

**Examiner**

Dawit Mengistu, Ph.D. – Kristianstad University

**Abstract (maximum 250 words)**

The thesis determines the type of deep learning algorithms to compare for a particular dataset that contains time-series data. The research method includes study of multiple literatures and conduction of 12 tests. It deals with the organization and processing of the data so as to prepare the data for prediction of an event in the time-series. It also includes the explanation of the algorithms selected. Similarly, it provides a detailed description of the steps taken for classification and prediction of the event. It includes the conduction of multiple tests for varied timeframe in order to compare which algorithm provides better results in different timeframes. The comparison between the selected two deep learning algorithms identified that for shorter timeframes Convolutional Neural Networks performs better and for longer timeframes Recurrent Neural Networks has higher accuracy in the provided dataset. Furthermore, it discusses possible improvements that can be made to the experiments and the research as a whole.

**Keywords**

Classification, Data Analysis, Deep Learning, Event Prediction, Machine Learning, Time Series

# Table of Contents

|   |           |
|---|-----------|
| <b><i>Introduction</i></b> .....                    | <b>5</b>  |
| Background .....                                    | 5         |
| Research Questions .....                            | 5         |
| Thesis Structure .....                              | 5         |
| <b><i>Methodology</i></b> .....                     | <b>7</b>  |
| Literature Study.....                               | 7         |
| <b><i>Literature Review</i></b> .....               | <b>8</b>  |
| <b><i>Theory</i></b> .....                          | <b>10</b> |
| Convolutional Neural Networks .....                 | 10        |
| Recurrent Neural Networks .....                     | 12        |
| <b><i>Experiment</i></b> .....                      | <b>15</b> |
| Database Description.....                           | 15        |
| Machine Description.....                            | 16        |
| Classification .....                                | 16        |
| Prediction .....                                    | 17        |
| <b><i>Results and Discussion</i></b> .....          | <b>18</b> |
| Social Aspects .....                                | 20        |
| Limitations.....                                    | 20        |
| Possible Improvement .....                          | 21        |
| <b><i>Conclusion</i></b> .....                      | <b>22</b> |
| <b><i>Bibliography</i></b> .....                    | <b>23</b> |
| <b><i>Appendix</i></b> .....                        | <b>26</b> |
| Appendix 1: Classification scripts explanation..... | 26        |

|   |           |
|---|-----------|
| <b>Appendix 2: Prediction script explanation.....</b> | <b>27</b> |
| <b>Appendix 3: Test results.....</b>                  | <b>29</b> |
| <b>Test 1.....</b>                                    | <b>29</b> |
| <b>Test 2.....</b>                                    | <b>30</b> |
| <b>Test 3.....</b>                                    | <b>30</b> |
| <b>Test 4.....</b>                                    | <b>31</b> |
| <b>Test 5.....</b>                                    | <b>31</b> |
| <b>Test 6.....</b>                                    | <b>32</b> |
| <b>Test 7.....</b>                                    | <b>32</b> |
| <b>Test 8.....</b>                                    | <b>32</b> |
| <b>Test 9.....</b>                                    | <b>33</b> |
| <b>Test 10.....</b>                                   | <b>33</b> |
| <b>Test 11.....</b>                                   | <b>34</b> |
| <b>Test 12.....</b>                                   | <b>34</b> |

# Introduction

## Background

According to an article published on Forbes, 2.5 quintillion bytes (i.e.  $2.5 \times 10^{18}$  bytes) of data is generated each day [1]. Data analysis company Domo estimates that 1.7 MB of data will be created each second for everyone on earth by 2020 [2]. It equates to over 143 GB of data per person per day. Data is created in every walk of life such as online shopping, online search, teleconference, entertainment, and so on. As the pace of data creation increases, development of tools for analyzing such data is also being accelerated. Computers are also being used, not just to get insight from existing data, but also to predict future values and events as well. To recognize patterns in data, one needed to have domain expertise in the specific field. However, with the improvement of various machine and deep learning algorithms, such pattern recognition has been opened to any data scientist, engineers, and developers.

An anonymized time-series dataset was provided by Diaverum, a venerable company in southern Sweden. Additional information regarding the dataset is provided in *Database Description* subheading of *Experiment* section. The author was given two milestones by the company to achieve. The first milestone is to classify certain data and the second is to predict a certain event in that data. Detail information regarding the milestones and tasks performed is provided in subsequent sections.

As requested by the company, information such as table names, field names, and the actual event are anonymized.

## Research Questions

**“How should time-series data be organized and processed for the purpose of prediction using deep learning techniques?”**

**“Which deep learning algorithm would be effective with the provided dataset?”**

## Thesis Structure

The thesis is divided into multiple sections. Following the *Background*, research questions are provided. It is followed by the *Methodology* section where the way literature

study was conducted is mentioned. *Literature Review* section contains summaries of literatures studied for the purpose of this thesis. This section is followed by *Theory*, in which the theory behind the algorithms used in the experiment is discussed. The experimentation includes database description. Finally, the results are discussed, and further improvements are suggested. The appendices include information on the development of scripts for classification and prediction purposes. It also includes the details of the 12 tests conducted.

## **Methodology**

The thesis is written after conducting literature study and experiment. The experiment is conducted in two steps. The classification is done as directed by the company, in which, a certain observation is classified based on a threshold value. Following the classification, the event is predicted using an appropriate algorithm. The methods to perform these tasks are determined after studying multiple articles and papers. The aim of the thesis is to find a way of processing the data provided so that it is easier to make the prediction. Since the prediction problem is a supervised one, it would be useful to explore the performance of some deep learning algorithms by conducting a number of tests.

## **Literature Study**

To embolden the experiment performed in the thesis, multiple articles and papers are reviewed. The literature study was performed so as to find out what other authors and researchers have already devised in the field of this thesis. Strong consideration was made to make sure the articles are peer reviewed. The content of the articles ranged from data preprocessing to deep learning. Apart from articles, this thesis also cites blog posts.

The literature study research was performed using online tools like ACM Digital Library [3], HKR Summon [4], and arXiv [5]. Extracts from books were retrieved via online sources and through physical copies of the books. Some of the search terms used in research for the thesis include data preprocessing, classification, deep learning, time-series analysis, and event prediction.

## Literature Review

In [6], Kotsiantis et al. discuss the need of data preprocessing and various methods of it. If there is noise or redundant information in data, it could cause unreliability and difficulty in training of the models. They state that there could be idiosyncrasies in the data such as outliers and null values. Since there could be a large number of features, method to select the feature subset can reduce the dimensionality of the data. The authors also warn about overfitting of the machine learning models, which cannot generalize upon unseen data. They also propose normalizing data if the data is in extreme range. These methods are applied in the analysis of the dataset.

Kotsiantis et al. mention that irrelevant and redundant information, noise, and unreliable data causes difficulty in the training; and thus, provide information regarding the data pre-processing steps including cleaning, normalizing, transforming, and extracting and selecting features [6].

In the article in [7], LeCun et al. explain what deep learning is and describe few deep learning algorithms. According to the authors, deep learning discovers “intricate structure in large data sets by using the backpropagation algorithm.” Backprop is done to update the weights to reach the global minima. They state that in order to label data using existing dataset, a supervised learning algorithm would be best suited. The two supervised learning algorithms described are the convolutional neural networks and the recurrent neural networks.

Karim et al. propose the use of the long short-term memory recurrent neural network for time-series classification in the article in [8]. According to the authors, the proposed models improve the performance of the network with minimum increase in model size. The authors also state that as the input to this model can be generalized, it has a multitude of applications on which it can be applied to. A method written to experiment of this thesis also includes the use of both CNN and RNN in the same network.

In [9], Fawaz et al. provide, among other models, the convolutional neural networks, for time series classification. The authors explain how a convolution “can be seen as applying and sliding a filter over the time series.” They state that the filter shows only a single dimension in a time series data. The result of convolution of a time series is another time series, which is then fed into subsequent layers such as pooling layer.

In the paper in [10], Borovykh et al. analyze the performance of convolutional neural network on different multivariate time series and compare it with long short-term memory network. The authors use CNN for time series forecasting with financial data. However, it is applicable to the dataset provided for this thesis as well.

In the article in [11], Yin et al. compare CNN and RNN in sentiment classification, relation classification, textual entailment, answer selection, question relation match, path query answering, and part-of-speech tagging. According to the experiment conducted by the authors, RNNs perform “well and robust” in different kinds of tasks. One of such tasks is time-series analysis which pertains to this thesis.

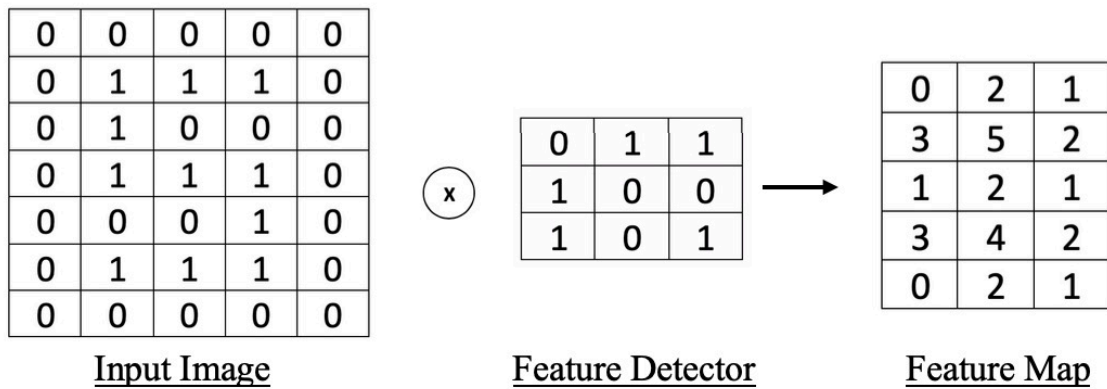
# Theory

The literature study determined that two deep learning algorithms – Convolution Neural Networks (CNN) and Recurrent Neural Networks (RNN) are suitable for event prediction in time-series data. Therefore, in the experiments for this thesis, these algorithms shall be tested. Following is the explanation of the two algorithms.

## Convolutional Neural Networks

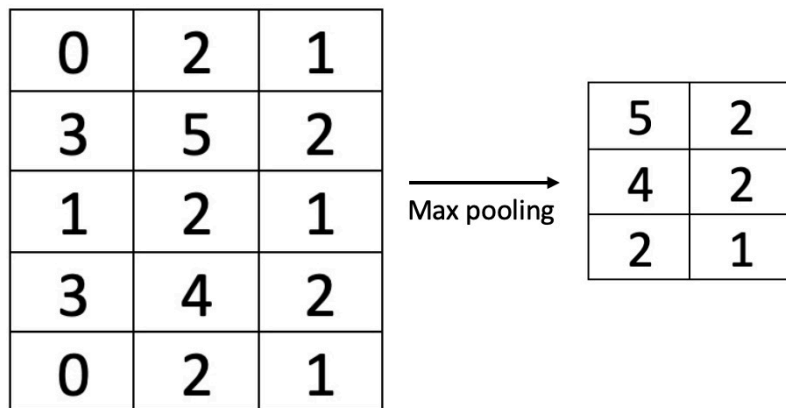
O’Shea and Nash state that Convolutional Neural Networks (CNNs) are similar to traditional artificial neural networks but has a difference in its primary use i.e. image recognition [12]. Yamashita et al. describe the building blocks of CNN in their paper [13]. Convolution is the first layer of CNN. It is a linear operation used for feature extraction. The original data is represented as a matrix. The convolution later basically reduces the size of that matrix. Even though some information is lost, the most important features are still detected by the convolution operation. Agarap suggests using Rectified Linear Unit on top of the convolutional layer to increase non-linearity in the network [14]. Convolutional operation transforms input image to a feature map using a feature detector as shown in *Figure 1*. Feature detector slides over the input image and each component multiplies respective component in the feature detector. The feature map corresponds to the number of 1s resulted from the multiplication.

In *Figure 1* the feature detector is applied to the top-left of the input image (I) i.e.  $I_{11}$  to  $I_{33}$  where  $I_{11}$  is the element in first column and first row, and  $I_{33}$  is the element in third column and third row. In this step, no same value overlaps each other. So, the resulting feature map element  $F_{11}$  is 0. Then the feature detector is moved one column to the right and is compared with  $I_{21}$  to  $I_{34}$ . Here, elements (value 1) at  $I_{22}$  and  $I_{32}$  are same as that in the feature detector. As, there are 2 same values, the feature map has a 2 at  $F_{21}$ .



*Figure 1: Convolution operation*

The second layer of CNN is the pooling layer. As explained by Scherer et al., pooling is performed to achieve spatial invariance by decreasing the resolution of the feature maps [15]. This means that even if the image is rotated, resized, or flipped side-to-side, the network should still be able to recognize it. Among the different types of pooling, Max pooling is the most popular [16]. Max pooling layer outputs a matrix that corresponds to the maximum activation over non-overlapping regions of size  $(K_x, K_y)$ , where  $K_x$  is the number of rows of the stride and  $K_y$  is the number of columns of the stride [17]. Max pooling operation with a stride of 2 on the feature map received above is shown in *Figure 2*.



*Figure 2: Max pooling*

Jin et al. explain an additional step to flatten the pooled feature map to make the input layer [18]. This step is shown in *Figure 3*.

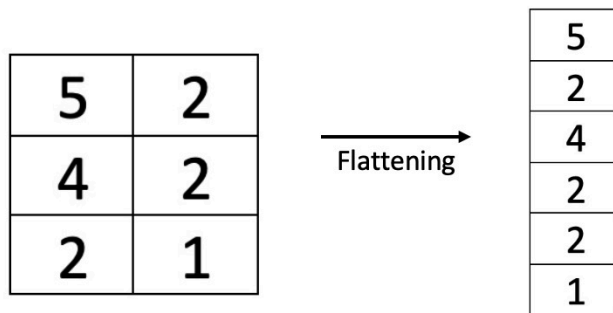


Figure 3: Flattening

Finally, the flattened feature map is passed as input layer to a fully connected layer to produce an output layer. The output layer can have any number of class as required. This is shown in *Figure 4*.

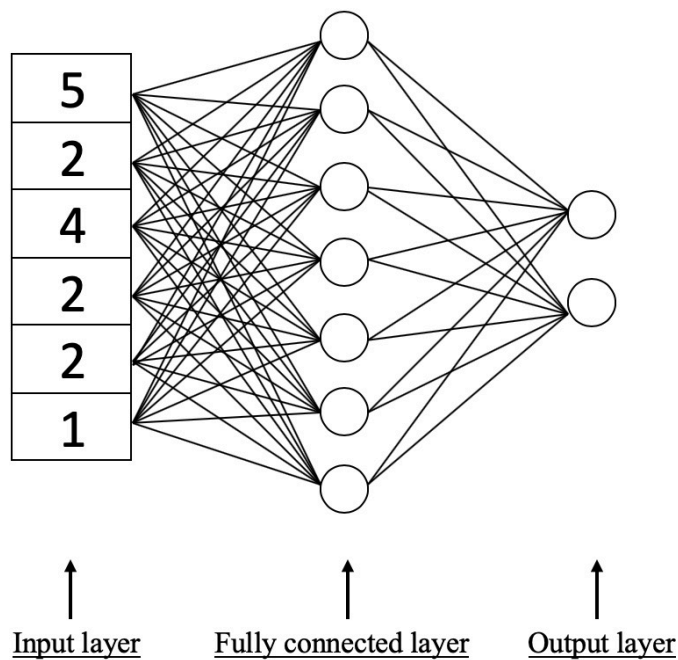


Figure 4: Full connection

## Recurrent Neural Networks

Recurrent neural networks (RNNs) are primarily used in tasks that deal with sequential data such as machine translation, music generation, and time-series prediction. RNNs use the sequence dynamics in the network of nodes to make predictions. As stated by Lipton, Berkowitz, and Elkan, standard neural networks rely on the postulation that observations are independent of each other [19]. However, there are many datasets, including the one

being used in this thesis, in which observations are related to previous observations. Lipton further adds that RNNs are a superset of feedforward neural network and possess the ability to pass information across time steps [20]. In the dataset provided, the event occurs in an observation based on the values of parameters in previous observations.

Suppose  $I = \{I_1, I_2, I_3, I_4\}$  is the input layer, where  $I_1$  through  $I_4$  are the input neurons;  $O = \{O_1, O_2\}$  is the output layer, where  $O_1$  and  $O_2$  are the output neurons; and there are 2 hidden layers as shown in *Figure 5*. The neural network in *Figure 5* is squashed together and is represented by a single circle in *Figure 6*. Each circle in *Figure 6* does not represent a single neuron but an entire layer of neurons i.e. neurons connected to itself through a temporal loop.

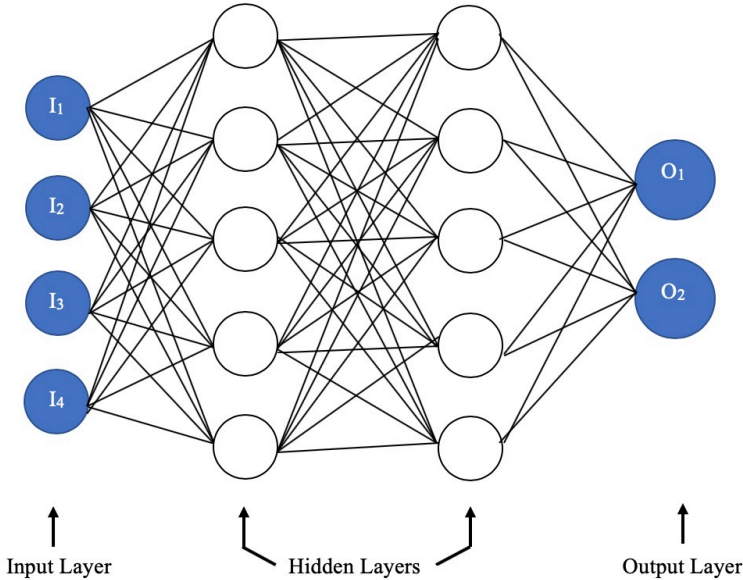


Figure 5: Neural network

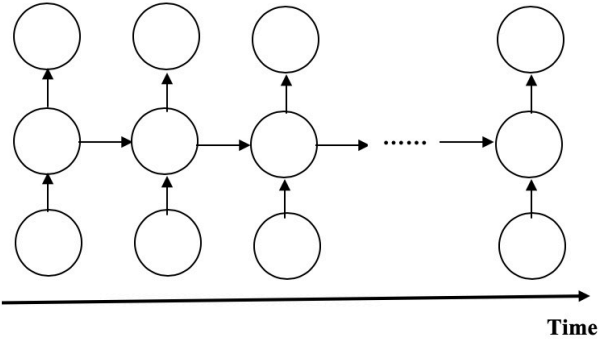
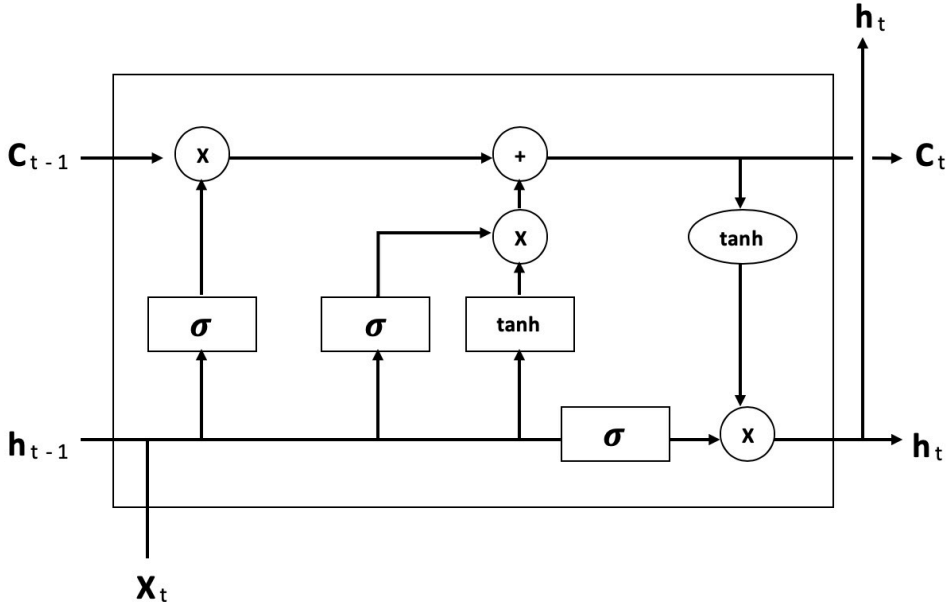


Figure 6: Recurrent neural network

According to Hochreiter, conventional backpropagation suffers from a long learning time (the vanishing gradient problem) [21]. All the neurons back through the time have to be updated during backpropagation. The lower the gradient, harder it is for the network to update the weights. The neurons closer to time  $t$  may be trained faster than those that are farther. However, the weights from all neurons are used in the training. Long Short-Term Memory Network is used as the solution for the vanishing gradient problem, so that the network can remember information for a long period. Each hidden layer consists of a large number of LSTM blocks that have a complex internal structure [22]. The inner working of LSTM is shown in *Figure 7*.

Here,  $C$  is the memory cell,  $h$  is the output,  $X$  is the input. Rectangles represent activation function and circles represent pointwise operation. Arrows represent copy. If  $C_{t-1}$  contains a data and  $X_t$  is the same data, then the memory can let to be flown freely. However, if a new data is in  $X_t$ , then the valve represented by top-left circle can be closed so as to let the new data flow to the valve represented by the middle circle. The circle with a plus sign can then extract as much information as possible. The output valve represented by the bottom-right circle extracts information from the memory and send it as an input to the next unit.



*Figure 7: Long Short-Term Memory architecture*

# Experiment

## Database Description

The dataset provided contains time-series observations that span over a period of 10 months. A time-series is a sequence of data points recorded at specific time intervals [23]. The dataset size is 3.68 GB. It is stored in a relational database - Microsoft SQL Server. It contains 7 tables out of which only 2 pertain to the work of this thesis. The tables contain different types of data such as temporal, string, and numeric. Information regarding the two tables are given below.

**Table 1: Information about Session table of the database**

|                         |  |
|-------------------------|--|
| <b>Name</b>             | Session  |
| <b>Number of fields</b> | 8  |
| <b>Field names</b>      | Id, SubjectId, DeviceId, SessionStatus, CreatedTimeStamp, SessionStartedTimeStamp, SessionEndedTimeStamp, CompletedTimeStamp |
| <b>Number of rows</b>   | 17,582   |

**Table 2: Information about DataRecord table of the database**

|                         |  |
|-------------------------|--|
| <b>Name</b>             | DataRecord   |
| <b>Number of fields</b> | 155  |
| <b>Field names</b>      | Id, SessionId, ReceivedTimeStamp, TimeStamp, ... (and so on) |
| <b>Number of rows</b>   | 4,305,553  |

A *Session* is a collection of *DataRecord*. It contains a time-series of *DataRecord*. Each observation for a session in *DataRecord* corresponds to a set of data for a particular minute during the session. There are variable number of minutes in each session, and hence, there are variable number of observations in *DataRecord* for that specific session.

For example, a set of continuous *DataRecord* from time 7:04:00 to 10:04:00 is considered one session with possibly 180 observations of *DataRecord*, 1 every minute and thereby 180 observations for 180 minutes.

In order to determine whether the event happens in a session, first, all of the sessions have to be classified as those in which the event happened and those in which the event did not happen. Following the classification, prediction has to be performed so as to whether the event would happen a certain number of minutes after current minute. For the prediction, based on the results of the literature study, two supervised deep learning algorithms – recurrent neural networks and convolution neural networks are taken into consideration.

## **Machine Description**

A Lenovo ThinkPad running Microsoft Windows 10 Enterprise is used to develop and execute the classification and prediction scripts. It has 6.38 GB of available physical memory.

## **Classification**

The first milestone relates to the classification of sessions into those in which the event happened and those in which the event did not happen. This information is not available in the provided dataset explicitly. The definition of the event, as stated and directed by the company, is as follows:

1. If value of parameter  $P$  measured before the session start (the pre-session value)  $< X_1$ , the event is regarded to happen if the absolute nadir value of parameter  $P$  becomes less than  $X_2$  between the start and the end times.
2. If value of parameter  $P$  measured before the session start (the pre-session value)  $\geq X_1$ , the event is regarded to happen if the absolute nadir value of parameter  $P$  becomes less than  $X_3$  between the start and the end times.

Two scripts are developed in Python to classify the observations. The scripts titled *EventClassifierInSession.py* and *EventClassifierinDataRecord.py* are included as parts of appendix of this thesis. The description of how the scripts work are also included in Appendix 1

## Prediction

To predict whether the event is likely to happen or not in a particular session, deep learning models need to be trained. As Yin et al. compare two deep learning algorithms namely the Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) in their experiment, this thesis also uses those networks in the experiment. The operations of the algorithms are explained in *Theory* section of this thesis. The prediction script is written in Python programming language.

Following is a list of libraries used in the script:

1. Pandas
2. Numpy
3. Keras
4. Pyodbc
5. Sklearn
6. Random
7. Timeit

The way the prediction script works is explained in Appendix 2.

and SoftMax function for binary output. Adam optimizer and binary cross entropy loss functions are used in this neural network as well.

The RNN consists of 5 LSTM layers and 1 dense layer with 100 neurons in each layer, and a dropout of 0.5. The number of epochs tested is 50. For CNN, Conv2D layer with a rectified linear unit (relu) activation function is used. It is followed by a MaxPooling2D layer with a pool size of 2x2. Dropout regularization is also used in this neural network. Dense layers with 256 and 128 neurons are added. The dense layers use relu as activation function. The output layer includes a Dense layer with 2 neurons. The final layer uses SoftMax as the activation as the result needs to be binary. For both of the networks, the optimizer used is adam and the loss function is binary cross entropy.

As the aim of the experiment is to find out how correct is the prediction, the metrics used are accuracy and confusion matrix in both RNN and CNN. The training data set is, then, fitted with both networks. Prediction is made for the test data. Score is evaluated for the test data and a confusion matrix is created.

## Results and Discussion

12 tests were conducted as part of the experiment. The tests include 6 tests with Convolutional Neural Networks and 6 tests with Recurrent Neural Networks. The results of individual tests are explained in *Appendix 3* of this thesis.

**Table 3: Tests 1-6 summary**

| Test No. | Algorithm | Timeframe  | Accuracy |
|----------|-----------|------------|----------|
| 1        | RNN       | 5 minutes  | 60.86%   |
| 2        | CNN       | 5 minutes  | 64.89%   |
| 3        | RNN       | 15 minutes | 61.73%   |
| 4        | CNN       | 15 minutes | 63.25%   |
| 5        | RNN       | 30 minutes | 66.02%   |
| 6        | CNN       | 30 minutes | 64.84%   |

**Table 4: Tests 7-12 summary**

| Test No. | Algorithm | Timeframe  | Accuracy |
|----------|-----------|------------|----------|
| 7        | RNN       | 5 minutes  | 61.00%   |
| 8        | CNN       | 5 minutes  | 64.36%   |
| 9        | RNN       | 15 minutes | 61.92%   |
| 10       | CNN       | 15 minutes | 62.48%   |
| 11       | RNN       | 30 minutes | 66.17%   |
| 12       | CNN       | 30 minutes | 65.69%   |

The 12 tests conducted show comparable results. The tests were done for timeframes of 5 minutes, 15 minutes, and 30 minutes for two sets of parameters of data. In 2 out of 3 timeframes of the test, Convolutional Neural Networks produced higher accuracy than Recurrent Neural Networks.

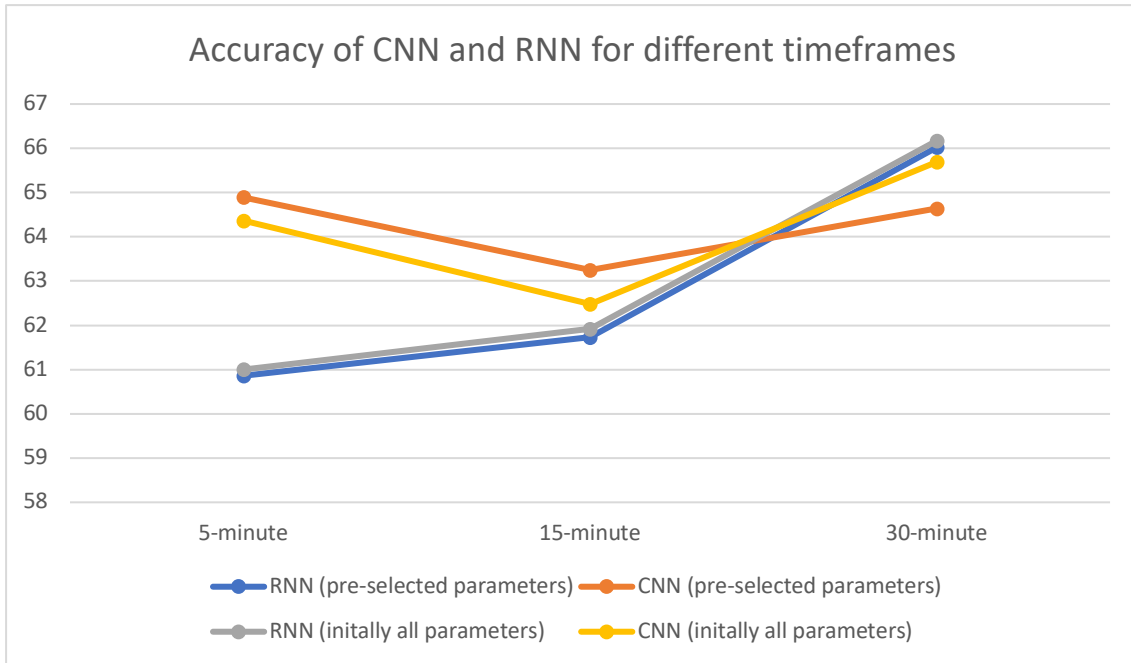


Figure 8: Accuracy score of CNN and RNN for different timeframes

For the 5-minute and 15-minute frames, CNN has between 0.56% and 4.03% higher accuracy than RNN. Only in the 30-minute frame, RNN fares better than CNN, albeit with smaller margins between 0.48% and 1.18%. A reason behind such results could be that for RNN a larger number of rows in dataset would mean more training data, and thus perform well.

Table 5: Difference in accuracy in different timeframes

| Timeframe  | Algorithm that produces higher accuracy | Accuracy difference in test with 43 pre-selected parameters | Accuracy difference in test with initially all parameters |
|------------|---|---|---|
| 5 minutes  | CNN                                     | 4.03%   | 3.36%   |
| 15 minutes | CNN                                     | 1.52%   | 0.56%   |
| 30 minutes | RNN                                     | 1.18%   | 0.48%   |

Although Karim et al. propose the use of the long short-term memory RNN for time-series classification in [8], it produces lower accuracy than CNN for the particular dataset used in this thesis. However, RNN generating higher accuracy than CNN in 30-minute timeframes means that if there are more observations in the training set, RNN produces better results than CNN.

In terms of the number of parameters (43 pre-selected parameters vs all columns initially), the results are as follows. In 15-minute and 30-minute timeframes, the latter has higher accuracies with for both RNN and CNN than the former. Same is the case for 5-minute timeframe with RNN. However, in CNN with 5-minute timeframe, the observations with 43-pre selected parameters have a higher accuracy than those with all columns initially. This result could be attributed to the fact that after removing columns as stated in *Appendix 2*, the total number of columns is only 48, which is comparable to the 43-pre selected columns.

## **Social Aspects**

The dataset provided for this thesis is anonymized so as to prevent it from being personally identifiable. The anonymization of the dataset do not have any bearing to the results of this thesis. Most of the parameters in the dataset contain numeric values only, which do not suggest any identification of a particular person.

The experiments conducted for this thesis could be used to prevent the event from happening during a session, if precautions are taken. If an event is predicted to happen in next 5, 15, or 30 minutes, appropriate actions taken can help mitigate the chance of the event occurring. However, new data collected in such scenario would not be helpful in training the models because it will not have any information regarding an event actually happening onwards.

## **Limitations**

The experiment conducted for this thesis is limited to the two deep learning algorithms – the CNN and the RNN. Similarly, the algorithms have been applied in a specific manner that fits the dataset provided.

## **Possible Improvement**

In terms of possible improvements in the future, one of the primary tasks that can be performed is to compare other algorithms such as by creating an ARIMA (Autoregressive integrated moving average) model [24] as well. Similarly, the deep learning networks currently used in the experiments can be honed more to use only those parameters that would have direct effect in the occurrence of the event.

The parameters of the networks can also be tweaked. For instance, the number of neurons used in the experiments conducted are set to 100. However, different numbers of neurons can be tested. Different number of LSTM layers can also be tested such as using 10 or 15 layers. The dropout frequency can also be tested for a range between 0 and 1. Furthermore, other performance metrics such as precision, sensitivity, specificity, and F1 score [25] can also be tested with to compare the two algorithms more rigorously.

Finally, the experiments conducted for this thesis can also be tested with other datasets so as to improve it even further.

## Conclusion

The work presented in this thesis compared the two deep learning algorithms – Convolutional Neural Networks and Recurrent Neural Networks. A number of steps were taken to properly organize the data so as to perform prediction using the deep learning algorithms. The observations in the dataset provided were not classified initially. Classification of event and non-event sessions was performed. The classification step includes modification of the database itself. Additional columns were created in *Session* table that includes Boolean information on whether an event occurred in a particular session or not and if it does occur, which row id in *DataRecord* table does it occur at. New column was created in *DataRecord* table as well. It includes Boolean information on each observation with numeric value of parameter P so as to identify if a particular observation is where the event occurs. This is well-explained in *Appendix 1* of this thesis.

After all observations have been classified, the data was fed into a prediction script. Such prediction was done for timeframes of 5 minutes, 15 minutes, and 30 minutes for two sets of parameters of data. This is explained in detail in *Appendix 2* of this thesis.

The comparison between the two deep learning algorithms identified that for shorter timeframes Convolutional Neural Networks performed better and for longer timeframes, Recurrent Neural Networks had higher accuracy in the provided dataset.

This thesis attempts to provide a technique to perform classification and prediction in complex dataset, which could be useful in other applicable datasets as well. It can also help other researchers identify a suitable algorithm as per need.

Use of machine and deep learning is increasing rapidly in a plethora of fields. Whether it be autonomous transportation to robotics to health science, machine and deep learning are forging strong path for artificial intelligence-powered societies. Leaving aside from personal opinions about such a society, there is no doubt that the benefits and efficiencies they provide would be immense for societies. This thesis has tried to explore a tiny fraction of one of such fields by providing the technique for organizing and processing of time-series data and for predicting events in such data, but much road is ahead of us.

## Bibliography

1. Marr B. Forbes. [Online].; 2018 [cited 2019 March 2]. Available from: <https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/>.
2. Domo. Domo.com. [Online].; 2018 [cited 2019 March 2]. Available from: <https://www.domo.com/solution/data-never-sleeps-6>.
3. Association for Computing Machinery. ACM Digital Library. [Online]. Available from: <https://dl.acm.org>.
4. Högskolan Kristianstad. Högskolan Kristianstad Summon. [Online]. Available from: <http://hkr.summon.serialssolutions.com>.
5. Cornell University. arXiv.org. [Online]. Available from: <https://arxiv.org>.
6. Kotsiantis SB, Kanellopoulos D, Pintelas PE. Data Preprocessing for Supervised Learning. International Journal of Computer Science. 2006; 1(1).
7. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015 May 27;(521): p. 436-444.
8. Karim F, Majumdar S, Darabi H, Chen S. LSTM Fully Convolutional Networks for Time Series Classification. IEEE Access. 2018; 6.
9. Fawaz HI, Forestier G, Weber J, Idoumghar L, Muller PA. Deep learning for time series classification: a review. Data Mining and Knowledge Discovery. 2019;; p. 1-47.
10. Borovykh A, Bohte S, Oosterlee CW. Conditional Time Series Forecasting with Convolutional Neural Networks. In ; 2017: Centrum Wiskunde & Informatica.
11. Yin W, Kann K, Yu M, Schütze H. Arxiv.org. [Online].; 2017 [cited 2019 March 23]. Available from: <https://arxiv.org/pdf/1702.01923.pdf>.
12. O'Shea K, Nash R. Arxiv.org. [Online].; 2015 [cited 2019 March 4]. Available from: <https://arxiv.org/pdf/1511.08458.pdf>.

13. Yamashita R, Nishio M, Do RKG, Togashi K. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*. 2018 August; 9(4): p. 611-629.
14. Agarap AFM. Arxiv.org. [Online].; 2019 [cited 2019 March 4]. Available from: <https://arxiv.org/pdf/1803.08375.pdf>.
15. Scherer D, Müller A, Behnke S. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. In 20th International Conference on Artificial Neural Networks (ICANN); 2010; Thessaloniki, Greece.
16. Deshpande A. adeshpande3.github.io. [Online].; 2016 [cited 2019 March 4]. Available from: <https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/>.
17. Nagi J, Ducatelle F, Di Caro GA, Cireşan D, Meier U, Giusti A, et al. Max-Pooling Convolutional Neural Networks for Vision-based Hand Gesture Recognition. In IEEE International Conference on Signal and Image Processing Applications; 2011.
18. Jin J, Dundar A, Culurciello E. Flattened Convolution Neural Networks for Feedforward Acceleration. In International Conference on Learning Representations; 2015; San Diego, CA, USA.
19. Lipton ZC, Berkowitz J, Elkan C. arXiv.org. [Online].; 2015 [cited 2019 March 2]. Available from: <https://arxiv.org/pdf/1506.00019.pdf>.
20. Lipton ZC. University of California, San Diego. [Online].; 2015 [cited 2019 March 2]. Available from: <http://zacklipton.com/media/papers/recurrent-network-review-lipton-2015v2.pdf>.
21. Hochreiter S. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*. 1998 April.
22. Grosse R. University of Toronto. [Online].; 2017 [cited 2019 March 2]. Available from:

- [http://www.cs.toronto.edu/~rgrosse/courses/csc321\\_2017/readings/L15%20Expanding%20and%20Vanishing%20Gradients.pdf](http://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/readings/L15%20Expanding%20and%20Vanishing%20Gradients.pdf).
23. Martin DEK. NC State University Department of Statistics. [Online].; 2014 [cited 2019 January 24]. Available from: <https://www.stat.ncsu.edu/people/martin/courses/st782/Notes/Definition,%20examples%20of%20time%20series%20and%20objectives%20of%20analysis%20%28sp%2014%29.pdf>.
24. Brownlee J. Machine Learning Mastery. [Online].; 2017 [cited 2019 May 9]. Available from: <https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/>.
25. Sunasra M. Medium. [Online].; 2017 [cited 2019 May 9]. Available from: <https://medium.com/thalus-ai/performance-metrics-for-classification-problems-in-machine-learning-part-i-b085d432082b>.
26. Godase A. Classification of Data Streams with Skewed Distribution. Master's Dissertation. Pune, India: College of Engineering, Department of Computer Engineering and Information Technology; 2012.

# Appendix

## Appendix 1: Classification scripts explanation

First, the *Session* and *DataRecord* tables are modified to include additional columns. Columns titled *Event* and *EventIndex* are added to *Session* table and a column titled *Event* is added to *DataRecord* table. The *Event* column in *Session* table indicates whether the event happened in that particular session. If the event did happen then the ids of the observations where the event happened (i.e. values are below 90 or 100 depending on the pre-value) are stored in *EventIndex* column, separated by a comma. The *Event* column in *DataRecord* table indicates whether in that particular observation the event happened. The event is represented by a 1 and non-event by a 0.

*EventClassifierInSession.py* script needs to be executed first. The start and end session ids are to be provided to the script. This is so as to make it possible to execute the script in multiple batches of sessions. This script finds out the pre-value and if it does not exist, gives it a default value of 90. Then, the script determines if the event happened or not by traversing through all observations in *DataRecord*. Finally, it assigns True/False value in the *Event* column of *Session* table and appends the index of that observation to the value in *EventIndex* column of the same table.

*EventClassifierInDataRecord.py* script is executed after the first script. The start and end session ids are to be provided to the script. This gets all the sessions from the *Session* table and according to the value in *Event* and *EventIndex* columns, assigns 0 or 1 to the *Event* column in *DataRecord* table. Eventually, all of the observations in which parameter P has a non-null value gets an *Event* value of 0 or 1.

## Appendix 2: Prediction script explanation

First, a connection with the SQL database is made using *get\_connection()* method. This step uses pyodbc library to make the connection. The method *get\_dfs()* queries *Session* and *DataRecord* tables. From *Session* table, *Id*, *StartedTimeStamp*, *Event*, and *EventIndex* columns are retrieved. All columns are retrieved from *DataRecord* table. The returned values are stored as two separate Pandas dataframe named *df\_session* and *df\_obs*.

Then, 107 columns are removed from *df\_obs* dataframe. These removed columns include values such as version number, static values, metadata, etc. The first row of the dataframe is also removed as it represents a test case. Similarly, sessions in which start time is null are removed from *df\_session* dataframe. Observations of sessions without a start time are removed from *df\_obs* as well. For the 43 pre-selected parameters, these steps are not performed, and parameters are selected based on the column names themselves.

Following this, those sessions that do not last longer than  $t$  minutes are removed from *df\_obs* using *remove\_session\_below\_t\_minutes()* method. The method takes dataframe, the number of minutes, and a set of session ids as parameters, and returns a modified dataframe. Observations of at least  $t$  minutes are required to make prediction. Various values of  $t$  are tested, results of which are elaborated in *Result* section of this thesis.

The method *get\_input\_output\_all\_sessions()* traverses through all session ids and for each session id, first it gets the start time and then gets the observations that have time after the start time. Then, it iterates over all of the observations that have time after the start time and for each observation gets  $t$  number of previous observations. Null values are replaced by -1. The new matrix with  $t$  number of rows is scaled and transformed using a *MinMaxScalar*. This scales all values to lie between 0 and 1. The method returns inputs and outputs for all sessions. Input includes frames that include Numpy array each containing  $t$  observations which in turn contains 46 columns. Output includes 1 column for each input frame representing whether the event happened or not at the end of that particular set of  $t$  rows.

The method *separate\_event()* goes through the input and output arrays and divides them into four lists – input frames for those observations in which the event happened, output for them, input frames for those observations in which the event did not happen, and output for them. As mentioned by Godase, skewed data can complicate task and therefore, a balanced data is required [26]. In the dataset provided, the number of observations in

which the event did not happen is significantly higher than those in which the event did happen. To minimize the effect, the n number of non-event observations are randomly selected, where n is the number of observations in which the event happened. Now, there are equal number of observations. They are combined and shuffled.

Then, the combined inputs and outputs are split into training set and test set using the *train\_test\_split()* method of sklearn library. The split size is 0.2 i.e. 20% of the combined data is used as test set. Since the output contains 0 or 1, dummy values are created. For observations where event did not happen, a 0 represents the output. However, with dummy values a 1 is placed for 0 value and a 0 is placed for 1 value. Similarly, for those in which the event happened, a 1 represents the output. With dummy values, a 0 is placed for 0 value and a 1 is placed for 1 value.

Then the neural network is created. Methods *create\_rn\_network()* and *create\_cn\_network()* create the respective neural networks. The network is initialized as a sequential network. For the recurrent network, LSTM layers are added followed by dropout regularization. Different number of neurons and the value of dropout regularizations are tested. This is explained more in the *Experiment* section. The Dense layer uses a SoftMax function to return a binary output. Adam optimizer and binary cross entropy loss functions are used.

## Appendix 3: Test results

### Test 1

The company selected 43 parameters in *DataRecord* table that are important in determining the occurrence of the event. The first experiment conducted is to test accuracy in the dataset with the 43 parameters. The training data consists of observations for previous 5 minutes. An RNN with 5 LSTM layers and 1 dense layer with 100 neurons in each layer, and a dropout of 0.5 produced an accuracy of 60.86% after 50 epochs. The value at row 1 in *Figure 8* is the score. The accuracy percentage is calculated by multiplying the score by 100 ( $0.6086... * 100$ ).

| Index | Type    | Size | Value              |
|-------|---------|------|--------------------|
| 0     | float64 | 1    | 1.1239386313639692 |
| 1     | float64 | 1    | 0.608623153911198  |

*Figure 9: Accuracy score of RNN for previous 5 minutes observations with 43 parameters*

The confusion matrix, a specific table layout allowing the visualization of the performance of an algorithm, is shown by *Figure 9*. The rows represent the actual values and the columns represent the predicted values.

So,

Row = 0 and column = 0 means, for an actual non-event, a non-event is predicted.

Row = 0 and column = 1 means, for an actual non-event, an event is predicted.

Row = 1 and column = 0 means, for an actual event, a non-event is predicted.

Row = 1 and column = 1 means, for an actual event, an event is predicted.

It shows that 1,501 sessions with no events and 1,054 with events were correctly predicted, while 577 sessions with no events and 1,066 sessions with events were predicted incorrectly. The optimizer used in this network is adam, loss function is binary cross entropy.

The accuracy score mentioned above can also be derived using a confusion matrix by following formula:  $(\text{Row0 Column0} + \text{Row1 Column1}) / (\text{Row0 Column0} + \text{Row0$

Column1 + Row1 Column0 + Row1 Column1). This is the ratio of the number of correctly predicted events divided by the total number of events.

|   | 0    | 1    |
|---|------|------|
| 0 | 1501 | 577  |
| 1 | 1066 | 1054 |

Figure 10: Confusion matrix and accuracy score of RNN for previous 5 minutes observations with 43 parameters

## Test 2

In the dataset with same 43 parameters, for training observations of previous 5 minutes, a CNN with a convolution layer, a max pooling layer, and 3 dense layers with 256 and 128 neurons is also tested. The dense layers use relu as activation function. The final dense layer uses SoftMax as the activation as the result needs to be binary. This network produces an accuracy of 64.89% after 50 epochs. As shown by the confusion matrix in Figure 9, 1,615 sessions with no events and 1,109 sessions with events are correctly predicted. 476 sessions with no events and 998 sessions with events are incorrectly predicted. The optimizer used in this network is adam, loss function is binary cross entropy.

|   | 0    | 1    | Index | Type    | Size | Value              |
|---|------|------|-------|---------|------|--------------------|
| 0 | 1615 | 476  | 0     | float64 | 1    | 0.6472958291695764 |
| 1 | 998  | 1109 | 1     | float64 | 1    | 0.648880419218864  |

Figure 11: Confusion matrix and accuracy score of CNN for previous 5 minutes observations with 43 parameters

## Test 3

For training observations of previous 15 minutes, an RNN with same structure as mentioned in Test 1 produced an accuracy of 61.73% after 50 epochs. 1,480 sessions with no events and 1,065 sessions with events are correctly predicted. 615 sessions with no events and 963 sessions with events are incorrectly predicted.

|   | 0    | 1    |
|---|------|------|
| 0 | 1480 | 615  |
| 1 | 963  | 1065 |

| $\hat{y}$ Index | Type    | Size | Value              |
|-----------------|---------|------|--------------------|
| 0               | float64 | 1    | 0.8968254294379255 |
| 1               | float64 | 1    | 0.6172689788844035 |

Figure 12: Confusion matrix and accuracy score of RNN for previous 15 minutes observations with 43 parameters

## Test 4

For training observations of previous 15 minutes, a CNN with same structure as mentioned in Test 2 produced an accuracy of 63.25% after 50 epochs. 1,424 sessions with no events and 1,184 sessions with events are correctly predicted. 636 sessions with no events and 879 sessions with events are incorrectly predicted.

|   | 0    | 1    |
|---|------|------|
| 0 | 1424 | 636  |
| 1 | 879  | 1184 |

| $\hat{y}$ Index | Type    | Size | Value              |
|-----------------|---------|------|--------------------|
| 0               | float64 | 1    | 0.8119554056029513 |
| 1               | float64 | 1    | 0.6325491146066367 |

Figure 13: Confusion matrix and accuracy score of CNN for previous 15 minutes observations with 43 parameters

## Test 5

For training observations of previous 30 minutes, an RNN with same structure as mentioned in Test 1 produced an accuracy of 66.02% after 50 epochs. 1,415 sessions with no events and 1,210 sessions with events are correctly predicted. 567 sessions with no events and 784 sessions with events are incorrectly predicted.

|   | 0    | 1    |
|---|------|------|
| 0 | 1415 | 567  |
| 1 | 784  | 1210 |

| $\hat{y}$ Index | Type    | Size | Value              |
|-----------------|---------|------|--------------------|
| 0               | float64 | 1    | 0.9535297848569075 |
| 1               | float64 | 1    | 0.6602112676056338 |

Figure 14: Confusion matrix and accuracy score of RNN for previous 30 minutes observations with 43 parameters

## Test 6

For training observations of previous 30 minutes, a CNN with same structure as mentioned in Test 2 produced an accuracy of 64.84% after 50 epochs. 1,227 sessions with no events and 1,351 sessions with events are correctly predicted. 752 sessions with no events and 646 sessions with events are incorrectly predicted.

|   | 0    | 1    | $\hat{y}$ | Type    | Size | Value              |
|---|------|------|-----------|---------|------|--------------------|
| 0 | 1227 | 752  | 0         | float64 | 1    | 0.9437394252485433 |
| 1 | 646  | 1351 | 1         | float64 | 1    | 0.6483903420523138 |

Figure 15: Confusion matrix and accuracy score of CNN for previous 30 minutes observations with 43 parameters

## Test 7

For a test that includes all of the columns of *DataRecord* table initially, the following result is obtained with RNN. The training data consists of observations for previous 5 minutes. An RNN with 5 LSTM layers and 1 dense layer with 100 neurons in each layer, and a dropout of 0.5 produced an accuracy of 61.00 % after 50 epochs. The confusion matrix includes 1,446 sessions with no events and 1,115 with events were correctly predicted, while 617 sessions with no events and 1,020 sessions with events were predicted incorrectly. The optimizer used in this network is adam, loss function is binary cross entropy.

|   | 0    | 1    | $\hat{y}$ | Type    | Size | Value              |
|---|------|------|-----------|---------|------|--------------------|
| 0 | 1446 | 617  | 0         | float64 | 1    | 1.2159219204868346 |
| 1 | 1020 | 1115 | 1         | float64 | 1    | 0.6100524058791783 |

Figure 16: Confusion matrix and accuracy score of RNN for previous 5 minutes observations with all parameters

## Test 8

For a test that includes all of the columns of *DataRecord* table initially, the following result is obtained with CNN. The training data consists of observations for previous 5 minutes. A CNN with a convolution layer, a max pooling layer, and 3 dense layers with

256 and 128 neurons is tested. The dense layers use relu as activation function. The final dense layer uses SoftMax as the activation as the result needs to be binary. This network produces an accuracy of 64.36% after 50 epochs. The confusion matrix includes 1,635 sessions with no events and 1,067 sessions with events are correctly predicted. 481 sessions with no events and 1,015 sessions with events are incorrectly predicted. The optimizer used in this network is adam, loss function is binary cross entropy.

|   | 0    | 1    | $\hat{}$ Index | Type    | Size | Value              |
|---|------|------|----------------|---------|------|--------------------|
| 0 | 1635 | 481  | 0              | float64 | 1    | 0.6383303989678011 |
| 1 | 1015 | 1067 | 1              | float64 | 1    | 0.643639828489757  |

Figure 17: Confusion matrix and accuracy score of CNN for previous 5 minutes observations with all parameters

## Test 9

For all of the columns, with training observations of previous 15 minutes, an RNN with same structure as mentioned in Test 7 produced an accuracy of 61.92% after 50 epochs. 1,452 sessions with no events and 1,101 sessions with events are correctly predicted. 616 sessions with no events and 954 sessions with events are incorrectly predicted.

|   | 0    | 1    | $\hat{}$ Index | Type    | Size | Value              |
|---|------|------|----------------|---------|------|--------------------|
| 0 | 1452 | 616  | 0              | float64 | 1    | 0.9802971280317934 |
| 1 | 954  | 1101 | 1              | float64 | 1    | 0.6192093136788802 |

Figure 18: Confusion matrix and accuracy score of RNN for previous 15 minutes observations with all parameters

## Test 10

For all of the columns, with training observations of previous 15 minutes, a CNN with same structure as mentioned in Test 8 produced an accuracy of 62.48% after 50 epochs. 1,234 sessions with no events and 1,342 sessions with events are correctly predicted. 862 sessions with no events and 685 sessions with events are incorrectly predicted.

|   | 0    | 1    | $\hat{\text{Index}}$ | Type    | Size | Value              |
|---|------|------|----------------------|---------|------|--------------------|
| 0 | 1234 | 862  | 0                    | float64 | 1    | 0.7583724235320247 |
| 1 | 685  | 1342 | 1                    | float64 | 1    | 0.6247877758190582 |

Figure 19: Confusion matrix and accuracy score of CNN for previous 15 minutes observations with all parameters

## Test 11

For all of the columns, with training observations of previous 30 minutes, an RNN with same structure as mentioned in Test 7 produced an accuracy of 66.17 % after 50 epochs. 1,265 sessions with no events and 1,366 sessions with events are correctly predicted. 678 sessions with no events and 667 sessions with events are incorrectly predicted.

|   | 0    | 1    | $\hat{\text{Index}}$ | Type    | Size | Value              |
|---|------|------|----------------------|---------|------|--------------------|
| 0 | 1265 | 678  | 0                    | float64 | 1    | 0.8311503323271001 |
| 1 | 667  | 1366 | 1                    | float64 | 1    | 0.6617203219315896 |

Figure 20: Confusion matrix and accuracy score of RNN for previous 30 minutes observations with all parameters

## Test 12

For all of the columns, with training observations of previous 30 minutes, a CNN with same structure as mentioned in Test 8 produced an accuracy of 65.69% after 50 epochs. 1,188 sessions with no events and 1,424 sessions with events are correctly predicted. 752 sessions with no events and 612 sessions with events are incorrectly predicted.

|   | 0    | 1    | $\hat{\text{Index}}$ | Type    | Size | Value              |
|---|------|------|----------------------|---------|------|--------------------|
| 0 | 1188 | 752  | 0                    | float64 | 1    | 1.095432782556929  |
| 1 | 612  | 1424 | 1                    | float64 | 1    | 0.6569416498993964 |

Figure 21: Confusion matrix and accuracy score of CNN for previous 30 minutes observations with all parameters