**Modeling the Sleeping Behaviour of a Patient Using Supervised Learning Techniques**

Aseel AlOrbani, Razan AlFar, Dimuthu Hemachandra, other profs

**Abstract**

**Introduction:**

Smart cities are one of the hot topics recently, where technology could come all together to make human lives easier and more connected. The aim is to build a smart city that is able to collect data, monitor and manage autonomously and in time. Smart system should be able to adapt to people’s different lifestyles and routines. Therefore, models to be built in nodes should be learning models, models should not be previously trained in different environments. Models running in such big scale system should have low latency and high accuracy to be dependable in taking actions. In specific, the current study is monitoring and controlling houses, for example system should be able to control devices in a house after monitoring the current situation by sensors distributed in the house.

Patients, elderlies and disabled are individuals who need extra care, should be comforted, and might sometime need all day long monitoring. Not everyone can have someone who do him or her these tasks. Therefore, studies and researches have been directed towards helping those people by any means possible. Therefore, building models to monitor their status at home, notify their caregiver/relative in case of emergency, to keep track of their sleeping pattern and lifestyle during the day.

The project focuses on the monitoring and analysis sleeping behaviour of a patient in their house. Sleeping pattern can give strong clues of how healthy person’s lifestyle is, and monitoring it will improve their lifestyle. For example, a doctor could make decision to increase or decrease the medicine doze after checking their sleeping pattern for the last few weeks. Other minor actions could be taken by the house system to ensure comfort environment for a better sleep when it is time to. Such action could be represented as changing the room and the house’s temperature, humidity and lightings. Therefore, the model should predict when the patient sleeps in order to make environment ready and suitable for the patient to sleep. In fact, such preparation for sleep will make the patient feel sleepy and sleep in time. This way this model will help the patient sleep better and healthy sleep pattern helps in improving their physical and mental health.

This report introduces the methods taken to build up such model. In the first section, the data set used project as the main input is explained. In the second section, the data science methods of all trials are defined and illustrated with technical details. In the third section, the results of methods applied are compared by accuracy and time needed for computation. In the last section, one method is chosen based on many specifications required to be used in practice.

Data set used in this project belong to a house that have two residents one of them is patient and the other is caregiver who is in the house most of the day. In this house, eight sensors are distributed in different rooms in 2017. Most of those sensors are motion sensors, which send a notification when triggered. Sensors have limited values, as an example: chair sensor have three different values, which are occupant, briefly vacant and vacant. The sensors are in action for 12 months, which gives us a sufficient data to make a good model. In one year a huge file with a lot of reading is produced, each notification in this file has a time stamp, source of reading and value of reading. In this project, bed and bedroom motion sensors are the most important and are mainly used as the source for prediction models. Additionally, other sensors are used to enhance the performance of prediction.

**Data:**

Data set used in this project belong to a house that have two residents one of them is patient and the other is caregiver who is in the house most of the day. In this house, eight sensors are distributed in different rooms in 2017. Most of those sensors are motion sensors, which send a notification when triggered. Sensors have limited values, as an example: chair sensor have three different values, which are occupant, briefly vacant and vacant. The sensors are in action for 12 months, which gives us a sufficient data to make a good model. In one year a huge file with a lot of reading is produced, each notification in this file has a time stamp, source of reading and value of reading. In this project, bed and bedroom motion sensors are the most important and are mainly used as the source for prediction models. Additionally, other sensors are used to enhance the performance of prediction.

**Methods:**

The data from all the sensors have been obtained as text files and as the first step, the data was converted into *panda data frames* which is popular data structure in Python programming language that is used in data analysis. Since the amount of available data was noisy (55836 readings from 7 sensors per year), due to the simultaneous activation of sensors making it difficult to be used without been filtered.

Trees:

As a first trial, only bed, bedroom and chair motion sensor are kept and all the rest of sensors are dropped out. Since bed and bedroom motion sensors have little readings comparing to the rest of the sensors. We have also kept the chair sensor in order to have values for the rest of the day, since both bed and bedroom motion sensors are active mostly during the night. The choice was taking the chair sensor since it has many readings during the day opposite of bed and bedroom sensors. This sensor reading also helps the model predict better, as we know that when chair sensor motion activates then the man is not sleeping for sure. Many other sensors helps with the prediction as well, but including them was turning the bed and bedroom sensors to be considered as noise comparing the size of readings.

First, in the preprocessing stage after dropping out most of the unnecessary readings. Then the value of the columns status, which is our target, is then changed into Boolean values, where: False represents the status where the person is waken up. It replaces the bed sensor reading being vacant, the chair sensor occupied or bedroom motion idle. And True represents the person sleeping. It replaces the bed sensor occupied reading.

Going forward, input is narrowed down into four criteria: month, day, hour, and minute. Therefore, as a first trial we tried to try on bagging, random forests with maximum of one, two, and three features and AdaBoost. Then GridSearch was used to choose the best parameters to get the best accuracy of trees.

SVM:

Therefore, as a second trial an empty data frame was defined with *Day, Counter, Status* as fields and allocated 1440 rows corresponding to each minute of a given day (60 min x 24). Then a method was implemented to go through all the data and detect the time points where the patient goes to bed (ABS bed sensor activated), and when the patient wakes us (ABS bed sensor vacated) and used those time points to fill the empty data frame with 1s for time periods of sleeps and 0s when he is awake. This leads to a data set of two classes where 1 represents minutes where the patient is on the bed and 0 represents otherwise.

Also, it was observed that the patient has not been present at home on some days and comes home in an irregular pattern making the sensor readings not interpretable since all the sensors resets at midnight. Therefore, these days were also removed from our analysis. Figure () shows a graphical representation of the status vs the counter for the selected data. To model a classifier that can predict the sleeping behaviour of this patient, the data was first divided in to sets where the odd days are considered as training data and the even days as the test data for validation. Several classifiers were tested with these data to inspect which model performs the best.

Random Forest Classifier:

Out of all the available classifiers nowadays, we used the Random Forest (RF) classifier1 since it is one of the most widely used classifiers and has outperformed other classifiers in many aspects2 3 4. RF is a collection of decision trees where each decision tree within the forests is built with a different bootstrap sample drawn from the original data set and then splitting according to the best split found over a randomly selected subset of features independently at each node. Once the forest is built, the classification can be done by simply aggregating the votes of all trees. There are few hyper parameters that you can tune in RF to improve the test accuracy5. The number of trees(n\_estimators), the maximum depth of the tree (max\_depth), the minimum number of samples required to split an internal node (min\_samples\_split) and the number of features to consider when looking for the best split (max\_features) are some of them.

Neural Network:

A neural network takes in inputs and then these inputs get processed in hidden layers using weights which are altered during training. The model then predicts a prediction, and weights keep adjusting to find patterns that make better predictions. The interesting thing about neural network that one does not need to specify what patterns to look for, it learns by itself.

1. Define Model:

To build our neural network, a high-level neural network API known as Keras was used. Keras is written in Python and supports convolutional and Recurrent Neural Networks (RNNs) [6]. To create a neural network layers, sequential model type was used. The neural network includes two hidden layers where each one utilized four nodes. In this model, one input feature and one target variable are being used. Furthermore, there many types of activation functions for hidden layers that can be used. Rectified Linear Units (ReLu) was used in our model. Sigmoid activation function was used in the output layer since it is a binary classification problem.

1. Loss and Optimizer:

Compile function in Keras is used to identify the loss function and the optimizer of the model. As this is a classification problem as mentioned earlier, binary\_crossentropy was used to calculate the loss function between the actual output and the predicted output. For the optimizer, Adaptive moment estimation was utilized which is referred to as Adam. Adam is a combination of RMSProp and Momentum, where momentum considers the old gradients to help smooth out the gradient descent [7]. Moreover, to measure the performance of our neural network model, accuracy, mean squared error (MSE), mean absolute percentage error (MAPE), and root mean squared error (RMSE) metrics are being used to.

1. Fit Model

The next step would be to fit the model and the way to do it in Keras is simply by using model.fit to train the model. To complete this part, the input data need to be set were the X train and the y train data were used, number of iterations (epochs) were set to 2000, and the size of batches was given as 10. The size of batch is useful as data sets are usually very big and this makes it hard to fit all the data all at once and for that the data can be divided into batches of equal sizes.

After that, the model training history will be returned after each epoch and this will consist the measurement of the model performances that we stated earlier such as accuracy, MSE, etc. After fitting model, the model performance is checked by predicting on the test data and check the accuracy of the test data.

Recurrent Neural Network (RNN):

Recurrent neural networks, of which LSTMs (“long short-term memory” units) are the most powerful and well known subset, are a type of artificial neural network designed to recognize patterns in sequences of data, such as numerical times series data emanating from sensors, stock markets and government agencies [8]. RNNs and LSTMs are different from other neural networks is as they consider time and sequence.

The network includes a feedback loop for learning and an extra parameter for connections between time-steps, which allows them to excel in modelling temporal behavior such as time-series, language, audio, and text.

1. Define Model:

The model was defined as sequential as well since the LSTM layers were sequential to each other. Then, LSTM layers were added to the model. Each layer had the number of LSTM units, the recurrent activation function and the activation function specified, with the rest left as default. A Dropout layer with value of 0.2 was added after each LSTM layer, which is a regularization method where input and recurrent connections to LSTM units are probabilistically excluded from activation and weight updates while training the network. This helped in reducing overfitting and improving model performance. After all layers have been added, a Dense layer of value 1 was added, to make our model more robust and indicating a single value prediction in the output.

* The first parameter to the LSTM layer is the number of neurons or nodes that we want in the layer.
* The parameter after the activations is the return\_sequences, which is set to true since we will add more layers to the model.
* The first parameter to the input\_shape is the number of time steps while the last parameter is the number of indicators.

Hyperparameter values in this network were chosen by default due to their common and safe use in other worldwide LSTM models. 2 LSTM layers were selected as adding further layers did not really affect the average RMSE, MSE or MAPE, which makes sense as the power of LSTM comes mainly from its units rather than layers, and the more layers we got, the more time needed for model compilation and fitting.

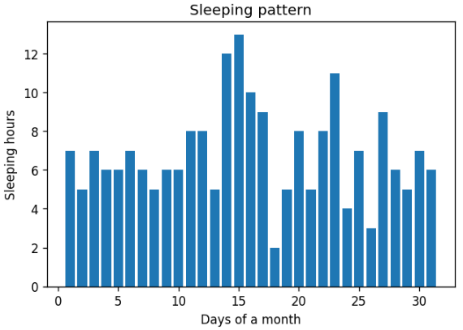
1. Loss and Optimizer:

For the loss function, binary\_crossentropy was used. Adam was used as the optimizer with a learning rate equal to 0.001.

1. Fit Model

For fitting the model, the same steps as the neural network explained previously were taken. The data was reshaped this time but the epochs and batch size were the same with values equal to 2000 and 10.

**Results:**

Sleeping pattern was one of the useful conclusion of this data, where the data frame have records of how many hours did this person sleep every day for the whole year. Collecting sleep data could help improve the patient life style and health, by linking bedtime with the house system. In order to improve the adapting feature of a smart house, the system should be able to adapt to residents’ life style. As an example, the following figure shows the given resident’s sleeping pattern for one month.

Trees:

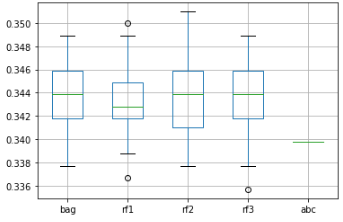
For this method, the models had four inputs which were Month, Day, Hour, and Minute. After plotting the error of the five models in Fig.1 and detailed results in Table 1.

Figure 1: error boxes for five tree models

*Table 1: Training and testing accuracy for trees models.*

|  |  |  |
| --- | --- | --- |
| model | Training accuracy | Testing accuracy |
| Bagging | 69.49% | 65.22% |
| Random forests with one feature | 69.49% | 65.16% |
| Random forests with two features | 69.49% | 65.14% |
| Random forests with three features | 69.49% | 65.12% |
| AdaBoost | 66.1% | 66.63% |

GridSearch function helped to find the best learning rate and best number of estimators for the best model found previously, which was the one with the less error rate. Sensitivity and specificity scoring functions were also two parameters to try on for finding the best model. The final model with best parameters found is the best estimator that analysis will go forward with. The model is then used to predict on the test data that was separated in the beginning before training, getting us the error rate and the accuracy in order to evaluate the model performance. Resulting in 66.83% for test accuracy, which did not improve much after tuning.

As a conclusion, those models did not give high accuracy and was not stable enough. Using trees was not efficient enough to proceed our work with.

**Neural Network:**

After applying the neural network on the data and training it over 200 epochs, the accuracy, loss, RMSE, and the MSE were calculated over each epoch. The results of this model is shown in figure 1, it can be seen from the figures that the accuracy was increasing over time which means that the model was learning better and giving better accuracy after each epoch. The model loss kept on decreasing which is optimal as the lower the loss, the better the model is. Additionally, the results for the model’s RMSE and MSE are presented in the figures as well and they are decreasing as well over time.

Table 2 summarises the results obtained from the neural network model and also it shows the time it took to compute the model over the data which is around 20 minutes by taking ten batches at a time as discussed earlier.

After training on the data, the test data which is the even days of the month was utilized to predict on the model. Table 3 outlines the results that was achieved, the accuracy and loss over time was taken into consideration as well. Confusion matrix was applied to get the true positives, true negatives, false positives, false negatives, recall, precision and finally the f1 score. The results received are considered efficient and are the best results that we got in comparison to all the other methods that we applied.

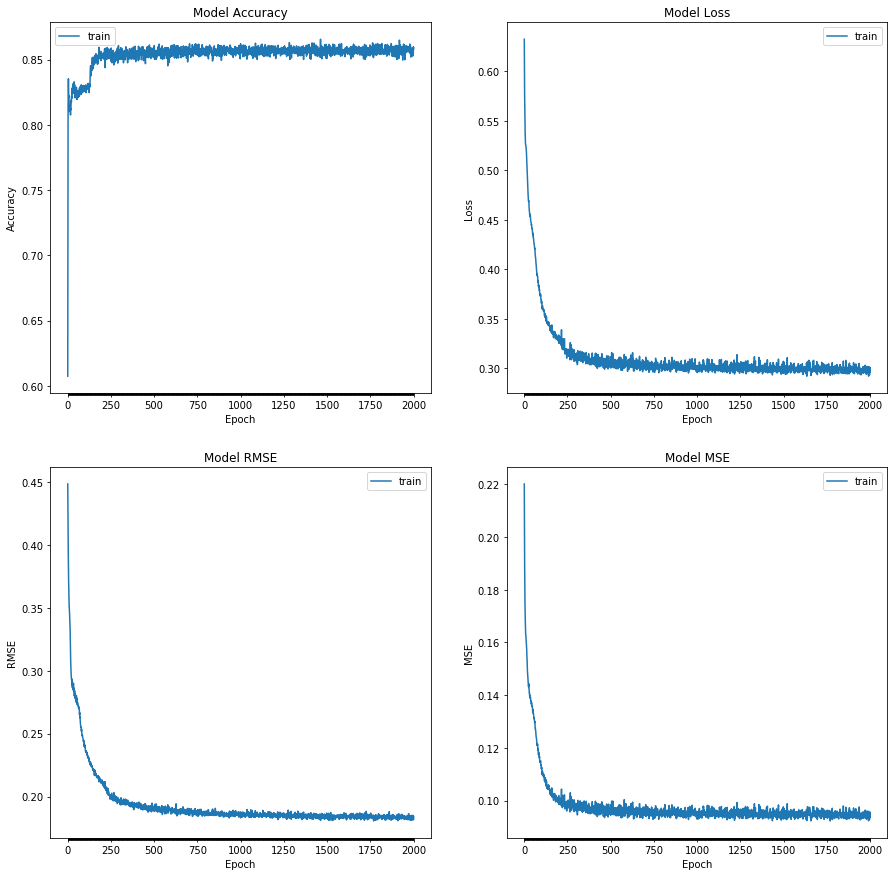


Figure : Neural Network running over 2000 epochs

Table : Results of the training data using NN model

|  |  |
| --- | --- |
| Training Accuracy: | 85.820 % |
| Training Loss: | 0.301 |
| Training RMSE: | 0.184 |
| Training MSE: | 0.095 |
| Training MAPE: | 92307960.0 |
| Total Time: | 1231.942s |

Table :Results of the testing data using NN model

|  |  |
| --- | --- |
| Test Set Accuracy: | 80.830 % |
| Test Set Loss: | 0.465 |
| Test Set RMSE: | 0.229 |
| Test Set MSE: | 0.140 |
| Test Set MAPE: | 57806708.0 |
| True Positive: | 2518 |
| True Negative: | 2141 |
| False Positive | 250 |
| False Negative: | 855 |
| Precision: | 0.910 |
| Recall: | 0.747 |
| F1 Score: | 0.820 |

**Recurrent Neural Network(RNN):**

After applying the recurrent neural network model on the data and training it over 200 epochs, the accuracy, loss, RMSE, and the MSE were calculated over each epoch. The results of this model is shown in figure 2, from the figures it is shown that the results are similar to the neural network.

Table 3 outlines the results obtained from the RNN model. The time taken to compute the model over the data is around 70 minutes by taking ten batches at a time. This is more than three times in comparison to the neural network model.

The test data was used again to predict on the model. The results received are presented in table 4. Confusion matrix was applied too to get all the required parameter to check the accuracy and performance of the model. The results achieved are considered efficient and are similar to the previous model. However, it is taking longer time so the neural network model would be more practical to use especially that the results are very similar.

Table :Results of the training data using RNN model

|  |  |
| --- | --- |
| Training Accuracy: | 83.715 % |
| Training Loss: | 0.388 |
| Training RMSE: | 0.250 |
| Training MSE: | 0.124 |
| Training MAPE: | 123610776.0 |
| Total Time: | 4232.518s |

Table :Results of the testing data using RNN model

|  |  |
| --- | --- |
| Test Set Accuracy: | 76.041% |
| Test Set Loss: | 0.482 |
| Test Set RMSE: | 0.298 |
| Test Set MSE: | 0.166 |
| Test Set MAPE: | 95184544.0 |
| True Positive: | 2000 |
| True Negative: | 2383 |
| False Positive | 8 |
| False Negative: | 1373 |
| Precision: | 0.996 |
| Recall: | 0.593 |
| F1 Score: | 0.743 |

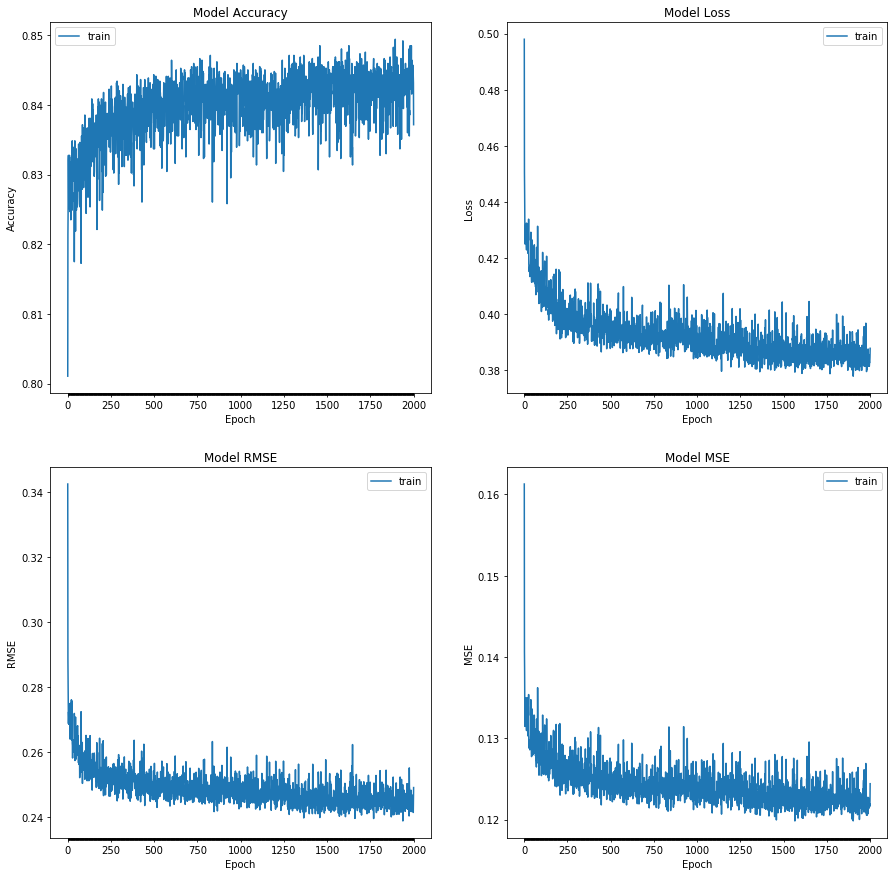


Figure :Recurrent Neural Network running over 2000 epochs

**Discussion:**

Based on our results from the previous section, the neural network model gave us the best results with training accuracy of 85.820 % and test accuracy of 80.830%. This is considered as a good performance as the accuracies are high and the training loss is 0.301 which is considered quite low. The work done using machine learning model can be used in the context of smart cities to leverage the quality of technologies performance and help ease people’s lives. The model achieved will aid in predicting sleeping patterns in several houses. This is mainly can be helpful for patients who struggle with their sleeping behaviors so this data can be helpful in better diagnosing their state and help to improve their medical records.

To help improve on the outputted results, more predictions could be done on the other sensors used to take certain actions based on the patient’s sleeping pattern. As an example, if the patient go to sleep during a certain period of time and it happens regularly then a heating sensor could be switched on in the winter automatically and so on. Furthermore, depending on the sensors that are available in different houses, more prediction can be done on different behaviors.

For future work, prediction could be done on other sensors to predict different human or environment behaviors. Additionally, more data sets with same and different environments could be taken into consideration as well to improve on the results.

**References**

1. Breiman L. Random Forests. *Mach Learn*. 2001;45(1):5-32. doi:10.1023/A:1010933404324

2. Fernández-Delgado M, Cernadas E, Barro S, Amorim D. Do we need hundreds of classifiers to solve real world classification problems? *J Mach Learn Res*. 2014;15(1):3133-3181.

3. Treeratpituk P, Giles CL. Disambiguating Authors in Academic Publications Using Random Forests. In: *Proceedings of the 9th ACM/IEEE-CS Joint Conference on Digital Libraries*. JCDL ’09. New York, NY, USA: ACM; 2009:39-48. doi:10.1145/1555400.1555408

4. A. Criminisi JS. *Decision Forests for Computer Vision and Medical Image Analysis*. Berlin, Germany: Springer International Publishing; 2013.

5. Probst P, Wright M, Boulesteix A-L. Hyperparameters and Tuning Strategies for Random Forest. *arXiv e-prints*. April 2018:arXiv:1804.03515.

6.

7.

8.