Modeling the Sleeping Behaviour of a Patient Using Supervised Learning Techniques

Aseel AlOrbani*, Razan AlFar*, Dimuthu Hemachandra°
*Department of Computer Science, Western University, London, Ontario
°Department of Biomedical Engineering, Western University, London, Ontario

Abstract -With the advancement of machine learning technologies most of our household items are becoming smart devices leading towards smart houses and smart cities. Takings steps forward, we investigate how these cutting-edge technologies can be used to make lives easier for patients and elderly people. Eight sensors have been recorded the movement of a house where a patient live with a caregiver through out a period of a year. These data were then analyzed with two different experiments, one with data through out the year and the second with data from one month. Experiment 1 used decision trees to make predictions for the sleeping behaviour and it produces a maximum test accuracy of 66.63%. However, the experiment two produces a highest test accuracy of 81.3% from both Random Forest and Support Vector Machine algorithms. Even though we used Neural Networks and Recurrent Neural Networks to see the ability of improving the accuracy, we observed no further improvements. Being able to predict the sleeping pattern of a patient who has irregular sleeping behaviour with over 81% is quite promising and these algorithms can be further developed to embed in smart devices in houses to make detect any abnormalities in the sleeping behaviour and trigger an alarm for the caregivers.

1.Introduction:

Smart cities are considered as one of the hot topics recently as technology is merging 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 complex software systems autonomously and efficiently. Smart systems 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 on a different environment. Models running in such big scale systems should have low latency and high accuracy to be independent in taking actions. In specific, the current study is related to 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 people are individuals who need extra care, should be comforted, and might sometime need constant monitoring. Therefore, studies and researches have been directed towards helping those people by any possible means. Therefore, building models to monitor their status at home, notify their caregiver/relative in case of emergencies, to keep track of their sleeping pattern and lifestyle during the day.

The project focuses on monitoring and analysing the sleeping behaviour of a patient in the house. Sleeping patterns can give strong clues of how healthy person's lifestyle is, and monitoring it can improve their lifestyle. For example, a doctor could take a decision to increase or decrease the medicine dose after checking their sleeping pattern. Other minor actions could be taken by the housing system to ensure a reliable environment to offer the patient a better sleeping experience. Such actions are changing the room and the house's temperature, and lightings. Therefore, the model should make some predictions when the patient sleeps in order to make environment ready and suitable for the patient to sleep. In fact, such preparations for sleeping will aid the patient to sleep faster. Hence, the model will help the patient to sleep better and improve their physical and mental health.

This report introduces the methods taken to build up such model. In the second section, the data set used project as the main input is

explained. In the third section, the data science methods of all trials done are defined and illustrated with technical details. In the fourth section, the results of methods applied are compared by accuracy and time needed for computation. Finally, in the last section, one method is chosen based on many specifications required to be used in practice.

2.Data:

Data set used in this project belong to a house in London, Ontario, that have two residents, one of them is a patient and the other is a caregiver who is in the house most of the day. In this house, eight sensors are distributed in different rooms during 2017 and 2018. 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 (Table 1). 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.

Note: The data set have been provided to us by our supervisor. It is still not available for public, however, it can be provided to anyone interested upon request.

Table 1: List of sensors and their available status

Sensor Number	Sensor Name	Available Status
1	Medical Button	Online, Offline
2	Front Door Contact	Opened, Closed
3	Sliding Glass Dr Contact	Opened, Closed
4	Pat's Bathroom Motion	Activated, Idle
5	Pat's Bedroom Motion	Activated, Idle
6	Living Room Motion	Activated, Idle
7	Chair Sensor	Occupied, Vacated, Briefly Vacated
8	ABS Bed Sensor	Occupied, Vacated

3.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. Therefore, we carried out two different experiments with different strategies to filter out data.

Experiment 1: As a first trial, only bed, bedroom and chair motion sensor were kept, and all the rest of sensors were 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. 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 help with the prediction as well but including them was turning the bed and bedroom sensors to be considered as noise due to the small number of readings.

After the preprocessing, the status of each sensor was changed into binary values, where: 0 represents the status where the person is woken up. It replaces the bed sensor reading being “vacant”, the chair sensor being “occupied” or bedroom motion being “idle”. And 1 represents the person sleeping where the bed sensor reads “occupied”. To have a visual inspection of the sleeping pattern of this patient, the number of hours that the patient was sleeping for each day was calculated. Figure 1, Shows bar graph of sleeping hours for the month of March 2017.

Going forward, input is narrowed down into four criteria: month, day, hour, and minute and only the first 6 days of each month was selected to improve the efficiency of the models. Data from odd days were used as the training data and that from even days as test data. We used three different classifiers that uses decision trees. Bagging, random forests with maximum of one, two, and three features and AdaBoost. Bagging depends on tree

classifiers, each one of them choses random elements from the input data for its own training. In bagging tree classifiers do not depend on each other. Whereas AdaBoosting depend on weak learners who take one feature as an input for training and then pass error measure to next weak learner to predict better. Therefore, in AdaBoost classifiers depend on each other, which means order of which classifier to go first matters. Then *GridSearchCV* tool available in *scikit learn* in python was used to choose the best parameters to get the best accuracy of trees. The test data was then used to validate these models.

Experiment 2: A second experiment was designed to further reduce the noise in the data with an intension of improving the accuracy of the models. Here, 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. Further, we modified our program to select data from any given month since we do expect the sleeping pattern change from month to month. In our results we have used only the data from the month August in 2018. Figure 2, shows a graphical representation of the status vs the counter for the selected data. As in experiment 1, the data was divided in training (odd days) and test (even days) data sets. With this new set of data, we investigated the performance of RF classifier, Support Vector Machine, Neural Network and Recurrent Neural Network that are described below.

3.1 Random Forest Classifier:

Random Forest (RF) classifier[1] is one of the most widely used classifiers and has outperformed other classifiers in many aspects[2 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 accuracy[5]. 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. In our experiment we used We used *GridSearchCV* to perform hyper parameter tuning.

3.2 Support Vector Machine:

Support Vector Machine (SVM) is a supervised learning technique that represents the data as points in space, mapped so that the separate categories are divided by the widest possible gap by creating hyperplane or set of hyperplanes in a high dimensional space. New data are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall. A good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class, since in general the larger the margin, the lower the generalization error of the model. For our experiment we used a nonlinear kernel, Radial Basis Function with a gamma value of 1.

3.3 Neural Network:

A neural network takes in inputs and process them into hidden layers using weights which are altered during training. The model then

predicts on them 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.

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.

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 were used.

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 checking the accuracy of the test data.

3.4 Recurrent Neural Network (RNN):

Recurrent neural networks, of which LSTMs ("long short-term memory" units) are a type of artificial neural network originated to recognize patterns in sequences of data, such as numerical times series data emanating from sensors[8]. RNNs and LSTMs are different from other neural networks is as they consider time and sequence.

Define Model: The model was defined as sequential since the LSTM layers are 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. A Dense layer of value 1 was added, to make our model more robust and indicating a single value prediction in the output.

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.

Fit Model: the same steps as the neural network explained previously were taken.

3.5 Comparison Metrics:

Different comparison matrices were used to evaluate the performance of our models.

Confusion Matrix: A confusion matrix is an often-used square matrix that summarizes perditions as true positives, true negatives,

false positives and false negatives. True positives and negatives are correctly classified predictions. A false positive is when the model predicts that the entry is of a category when it is not, and a false negative is when the model predicts that the entry is not of a category when it is.

Accuracy: The accuracy of the model is the ratio of true predictions to total predictions. Thus, it is a percentage of how many of the predictions were correct. A high accuracy therefore refers to a model that makes the most number of true predictions.

$$Accuracy = \frac{true\ predictions}{total\ predictions}$$

Recall: is the ratio of the total number true positives for a class and the total number of entries in the tested data that were of that class. A high recall indicates the class is correctly being recognized by the model and there are a small number of false negatives.

$$Recall = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

Precision: The ratio of true positives for a class and the total number of predictions the model made for that class. A high precision indicates an example labeled as positive is indeed positive and there are a small number of false positives.

$$Precision = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

F1-Score: The F1-Score is another widely used accuracy measures that represents both the precision and recall of the model. This measure uses the harmonic mean in order to weigh the lower value more heavily.

$$\text{F1\_Score} = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}}$$

4. Results:

Data Visualization: Figure1 was obtained to visualize the sleeping behaviour of this patient in the month of August 2018. Figure 1a, is a bar plot of the number of hours that the patient slept for each day and figure 1b, shows a scatter plot of the status of the bed sensor during the same month as a function of the time in minute during each day. It can be observed that the amount of sleep the patient gets is not consistent over the month, and also during a given day the subject takes small naps and also wakes up in the middle of the night.

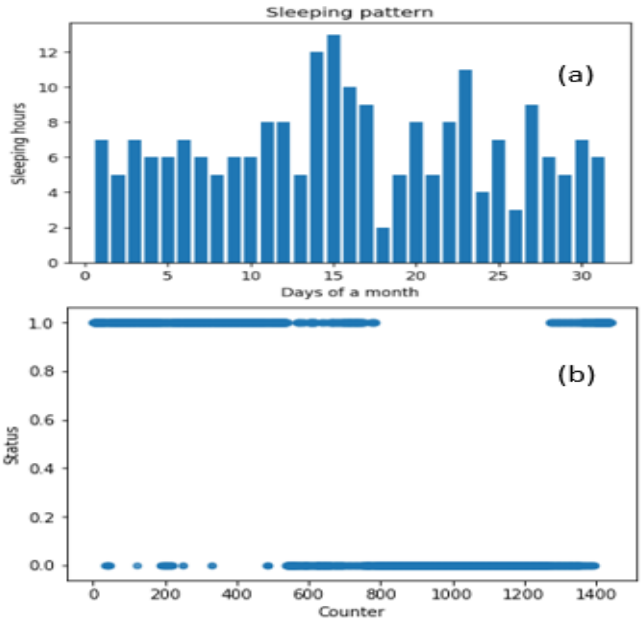

Figure 1:(a) Bar graph of the number of hours of sleep per day in the month of August. (b) Status of the bed sensor (1: sleep, 0: awake) as a function of counter (time in minutes)

Experiment 1: As mentioned in the methods section, the experiment 1 was performed using Month, Day, Hour, and Minute as features to predict the sleeping state of the patient using different decision tree classifiers. The highest test accuracy of 66.63% was reported from the model AdaBoost. The Bagging and the RF models seemed to be giving very similar results (test accuracy of 65%). Table 3 in appendices summarizes the accuracies of each model.

Experiment 2: Preprocessed data from the second experiment of the month of August was used in this here and the performance of the RF, SVM, NN and RNN models to predict the sleeping behaviour of the patient are summarized in table 2 below. The corresponding test accuracies for the 4 models are 81.3%, 81.3%, 80.8% and 76.0%. Confusion matrices for the 4 models are shown in figure (2).

After applying the neural networks on the data and training it over 2000 epochs, the accuracy, loss, RMSE, and the MSE were calculated over each epoch. These plots are shown in the appendix (Figures 3, 4). The overall time taken for NN was 20 min. and RNN was 70 min.

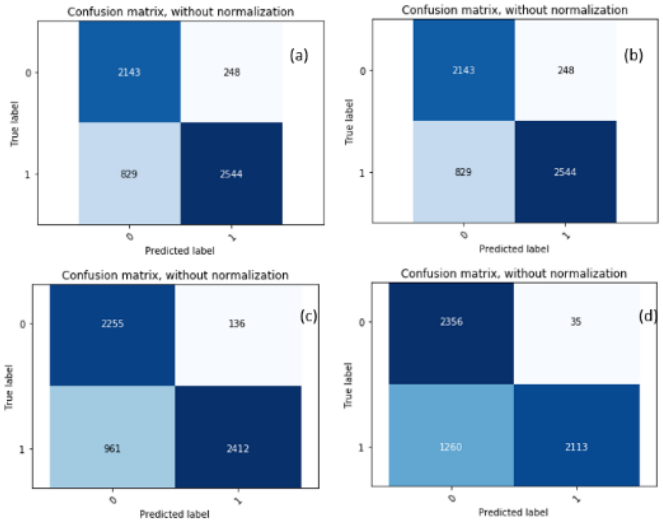

Figure 2:Confusion matrices for the RC (a), SVM (b), NN(c), RNN(d).

Table 2: Training and testing accuracy for experiment 2.

	RF	SVM	NN	RNN
Training accuracy	91.5%	91.5%	85.8 %	83.7 %
Test accuracy	81.3%	81.3%	80.8 %	76.0%
TP	2544	2544	2518	2000
TN	2143	2143	2141	2383
FP	248	248	250	8
FN	829	829	855	1373
Precision:	0.911	0.911	0.910	0.996
Recall:	0.754	0.754	0.747	0.593
F1 Score:	0.825	0.825	0.820	0.743

5. Discussion:

Two experiments were carried out to evaluate the performances of different classifiers with their ability to predict the sleeping behaviour of a patient that have been monitored through out a year using in house sensors. Visual inspections of the data showed that the sleeping behaviour of the subject is very irregular and the amount of sleep the subject takes varies over the time.

First experiment used the bed, bedroom and the chair sensor data through out the year and trained decision tree based classifiers to predict the sleeping patterns. The accuracies were very low with a maximum of 66.63% for the AdaBoost method. All the other classifiers also have shown very similar results. This explains that fitting a model to predict the sleeping pattern for the entire year is not that accurate. A good explanation would be the sleeping behaviour changes due to seasonal changes and also the increase of the noise as we include more data.

However, the second experiment was able to provide more promising results after narrowing down the analysis to one month. The RF and SVM was able to produce the same results recording an accuracy of 81.3%. Even though we tried to improve these accuracies using more sophisticated deep learning algorithms, they also provided very similar results (80.8% for NN and 76.0% for RNN) showing no further improvement of the testing accuracy.

In conclusion, the machine learning techniques can predict the sleeping pattern of the patient within a month with a higher accuracy compared to that for a year. better diagnosing their state and help to improve their medical records. Having a caregiver in the hose has made it complicated to interpret the data because the motion sensors also detect the second person's behavior. If the data can be obtained just from the patient, it would help improve the performance of these models. All the codes used in this project is available at the GitHub https://github.com/aorbani/SmartCity_DS for anyone wish to reproduce the results with the permission to access the data.

For future work, we will extend our models so that it has a separate model trained for each month of the year. These models can be embedded in smart devices at home to learn the sleeping behavior of the patients or elderly people to trigger an alarm when detects any unusual behavior or to control the temperature of the house accordingly.

Acknowledgement: We would like to thank our supervisor professor Michael Bauer, and our course instructor professor Jorn Diedrichsen for the constant help. Our warm gratitude goes to the course's Teaching assistants Demetri and Nathaniel as well.

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] Khandelwal, Renu. "Building Neural Network Using Keras for Classification." Medium, Data Driven Investor, 10 Jan. 2019, medium.com/datadriveninvestor/building-neural-network-using-keras-for-classification-3a3656c726c1.

[7] Tanwar, Sanchit. "Building Our First Neural Network in Keras." Medium, Towards Data Science, 9 Sept. 2019, towardsdatascience.com/building-our-first-neural-network-in-keras-bdc8abbc17f5.

[8] "A Beginner's Guide to LSTMs and Recurrent Neural Networks." Pathmind, pathmind.com/wiki/lstm.

Appendix

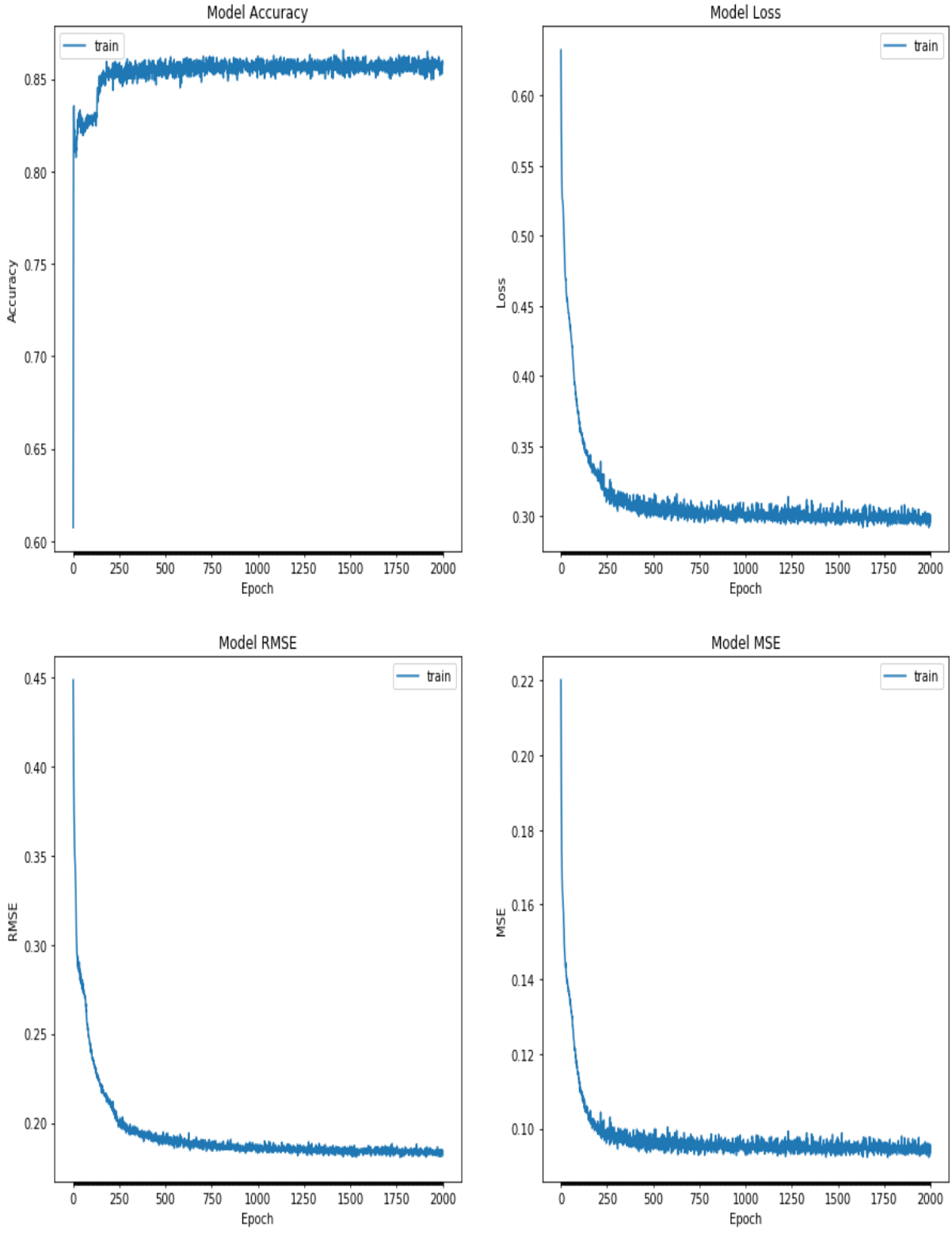

Figure 3: Neural Network running over 2000 epochs

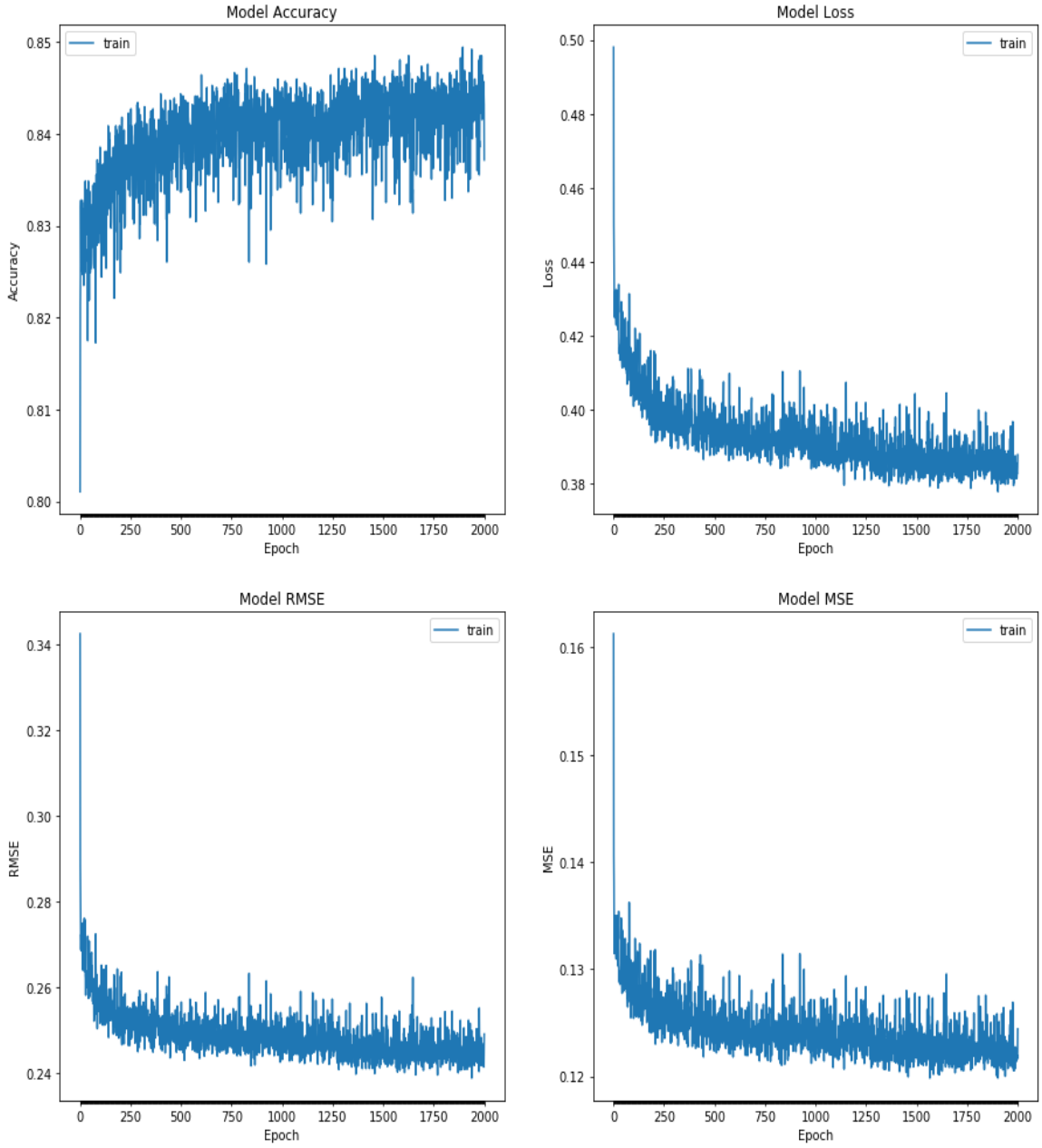

Figure 4: Recurrent Neural Network running over 2000 epochs

Table 3:Training and testing accuracy for the models in experiment 1.

model	Training accuracy	Testing accuracy
Bagging	69.49%	65.22%
RF with 1 feature	69.49%	65.16%
RF with 2 features	69.49%	65.14%
RF with 3 features	69.49%	65.12%
AdaBoost	66.1%	66.63%