3.2 Temporal model

Data Transformation The data was first set to time series data type and variables were flatten out to columns. Then the data from different patients were merged. Inspecting the data revealed that measures from patients are not from exact same time period and considering the uneven sparseness of the data might affect the model, so the only the data ranging from 2014-3-20 00:00:00 to 2014-5-01 00:00:00 was extracted for the analysis. Next, the data was under sampled by 0.5 hour. Missing values were filled by the values from the previous time stamp.

The temporal model used was the standard LSTM model in Keras package in Python. Although the features aggregation was not applied in this model, the Pearson correlation was done on the 19 variables and selected the 5 variables (including the output variable mood) with correlation coefficient over a cut-off 0.3.

Then to make sure the data could be fitted to a ML model, variables were first checked if they are stationary. Then the data was separate to input and output patterns where the observation at the previous time step is used as an input

to forecast the observation at the current time stamp. Besides, 12*0.5 hours' previous lag time stamps were included for each instance. The lag time stamps data and past observations as input were conducted was because during Back-propagation through time (BPTT), the recurrent layer would take the output from the previous time stamp accumulate error. So the input from prior time step would influence the error and decide the weigh update. [1] Therefore, we used the last half hour's observation as input value. Finally, we scaled the data to value between -1 and 1 to meet the default hyperbolic tangent activation function of LSTM model.

LSTM Model The LSTM model was defined with 50 neurons in one hidden layer, and one output layer for predicting the mood. Mean Absolut Error (MAE) was used as loss function, and Adam stochastic gradient descent was used as optimization scheme.

Training and validation For validation scheme the walk-forward validation was used: test set is chosen behind the training set and we increase the size of the training set each validation iteration. Since the ultimate goal is to predict the mood value for the next day, we predict the next 48 instances (24 hours) and calculate the mean as the prediction for the next day. (Fig. 5) In order to carry out statistical analysis, we do the validation 100 times on the entire dataset. To evaluate the trained model, we calculate the Mean Square Error (MSE) between forecasting mood values and actual values. Besides, we set a benchmark model where the prediction was the mood value from the previous day, and we calculated the MSE between it and actual values.

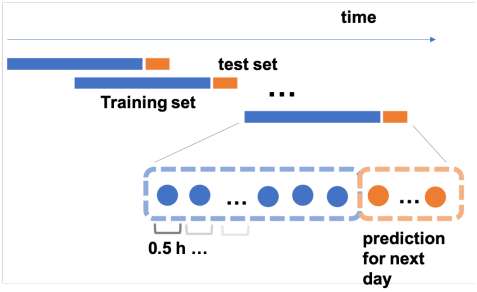

Fig. 5: Walk-forward model validation

Performance For each forward validation run, we calculated the mean of the prediction of mood values over 24 hours. After the training we calculated the MSE between the actual vs prediction, and between the actual vs benchmark.

We got 0.027 for the former one, and a lot higher value 0.23 for the latter one. This means the RNN model we built performed a lot better than the benchmark model. This difference could also be witnessed from the line plot (Fig.6) for the prediction for the last 36 days. The blue line (forecasting) and green line (actual) are almost on top of each other, while the orange line (benchmark) is far away.

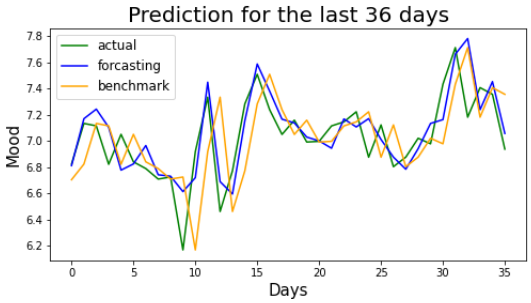

Fig. 6: Mood prediction for the last 36 days

4 Evaluation

4.1 Statistical evaluation

To further validate the differences in their performance, the statistical analysis was conducted. For the prediction of each time stamp, we calculated:

$$Percentage\ distance = \ln(\frac{\|M_{prediction} - M_{actual}\|}{M_{actual}} \times 100\%)$$

between the prediction and actual mood value, and actual and benchmark values. Log function was taken to normalize the distribution. After this, two groups have their variances respectively 2.88 and 2.83. The p-value of 3.6381e-6 got from Levene test further showed their equal variances. Now that the assumption of normal distribution and equality of variances are met, we do the two-sample t-test and got p-value 3.27e-11 (¡0.01). This demonstrates that two groups are significantly biased, and it's confirmed that RNN model is superior than the benchmark model. The same processing was done for feature engineered model. But unfortunately two distance groups have very biased variances even after normalization. This might be because feature engineered model had differentiated prediction performance for different patients. The outliers in (Fig.7, right panel) is probably because the rolling windows generated same mood values for the current day and the previous day. The overall distinguishes between the performances could be seen from the Kernel estimation plot (Fig.7): LSTM model has lower log percentage difference and so it is shifted left away. While

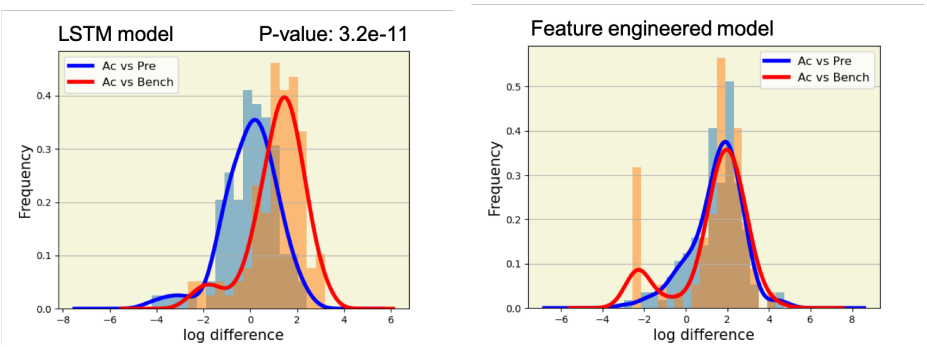

Fig. 7: Kernel density estimation

the feature engineered model's distribution is on top of the benchmark model's distribution and could deduce there's not a significant difference between the two models. This could also be supported by the MSE both models got.

4.2 Interpretation

5 Conclusion

From the above mentioned statistical analysis one can safely conclude that the performance of the temporal model versus that of the features engineering model is much better and using classical machine learning approaches in forecasting problems could be challenging. Although this approach is very fast in terms computational time during both training and testing it is very challenging to choose the most appropriate features given a problem. To overcoming this problem, requires experience on data exploration and very deep understanding of the nature of the studied problem. In addition to that it might be difficult for this model to become generalized for groups of people with very different habits while they are using their phone. On the other hand, forecasting approaches although are in general more computationally expensive could give us better results with less effort in terms of features engineering. Specifically, BPTT that is used by the recurrent neural net becomes computationally expensive as the time steps increases. Additionally, while the input time series increases there might be the possibility that the accumulated error using time derivatives vanish or explode the weights and make slow learning and model skill noisy. Further studies about the machine learning approaches could explore additional features construction, features selection, different methods of aggregation the time such as expanding windows etc, and different machine learning algorithms.

For the temporal model we could try to tune the hypeparameter with the limit size of data set with more reasonable validation scheme. Besides, Truncated BBTP could effectively reduce the computation time and somehow boost the

performance of RNN model. In some ways we could even combine the data engineering with the temporal model to improve the accuracy.

1. Mozer, M. C. (1995). "A Focused Backpropagation Algorithm for Temporal Pattern Recognition". In Chauvin, Y.; Rumelhart, D. (eds.). Backpropagation: Theory, architectures, and applications. ResearchGate. Hillsdale, NJ: Lawrence Erlbaum Associates. pp. 137–169. Retrieved 2017-08-21.