Predicting Mood With Smartphone Behaviour

Joshua Oyeto[2732733], Sergio Alejandro Gutierrez Maury[2647606], and Victor Retamal Guiberteau[2741820]

Vrije Universiteit Amsterdam (VU)
Data Mining 2022

1 Introduction

Smartphones are now critical assets in the daily functioning of the human world. According to Statistica [1], as at 2020, the penetration rate is 78% meaning $6.4bn$ of the world's $7bn$ people have mobile phone subscriptions. On the average, each individual spend about $5-7$ hours a day using their phone.Hence, it is plausible to link a person's mental state with their smartphone usage patterns.The mental state feature of importance here is the mood. Moods are inextricably intertwined with people's interactions with the world. This is a bidirectional influence: our experiences in the world engender emotional reactions, and in turn these emotions shape our behaviour and interactions.

In this advanced assignment of the Data Mining Techniques course at the Vrije Universiteit Amsterdam, the relationship between daily interactions with smartphone features(apps) and the self-reported mood rating of the users is statistically analysed and modelled using both feature engineered and temporal machine learning methods. The resultant models could allow for the prediction of a phone's user typical mood for the next day from the data of phone features they interact with.

1.1 Related Work

In the last two decades, the field of affective computing has explored predicting mood, well-being, happiness, and emotion from sensor data gathered through various sources. Harper and Southern [2] investigate how a uni-modal heartbeat time series, measured with a professional electrocardiogram device, can predict emotional valence when the participant is seated.

Several studies estimated mood, stress, and health with data from multimodal wearable sensors, a smartphone app, and daily manually reported behaviors such as academic activities and exercise, claiming maximum accuracies of 68.48% [3], 74.3% [4], 82.52% [5], all with a baseline of 53.94%. Another study scored 78.7%, with a baseline of 50.4%[6]. The studies mentioned above were executed professional equipment and expensive data recording processes. One other study that used less expensive methods predicted mood from passive data, specifically, keyboard and application data of mobile phones with a maximum accuracy of 66.59% (62.65% if without text). However, this project simplified

mood prediction to a classification problem with only three classes. Furthermore, compared to a high baseline of more than 43% (due to class imbalance), the prediction accuracy of about 66% is relatively low.

2 Data Preprocessing

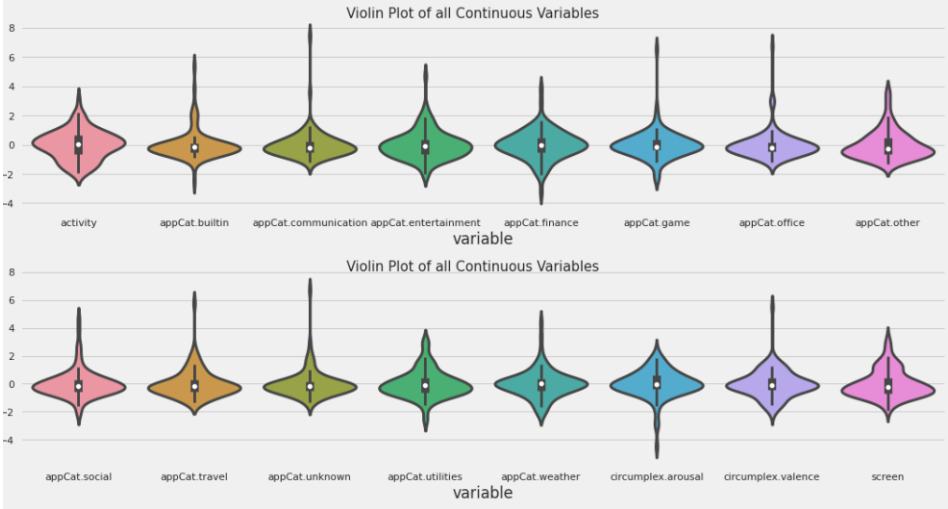

Fig. 1: Distribution of all Continuous Variables

2.1 Exploratory Data Analysis

The dataset used in this project has five months of smartphone usage data, from February to June 2014 for 27 users. It has 376,912 rows of data containing the following columns: unique ID of the participant, timestamp of each record/row, variable(type of record), and value (value of record). The data has been summarized in the table below.

Some rows discovered to have negative duration were deleted. After executing user and date level aggregation by reshaping the data, all columns, aside the date and user id, have null values. Nearest neighbour algorithm [7] with 10 neighbours was employed to fill up the null values. The distributions in the dataset remained the same after imputation, confirming the suitability of the imputation method.

Figure 1 show the distribution of all continuous variables in the dataset of the user. The Y-axis has been scaled to allow variables to be plotted on the same plane. Most variables are spread around the median. Call and SMS columns were excluded because they had a value of 1 for all users. Valence and Arousal

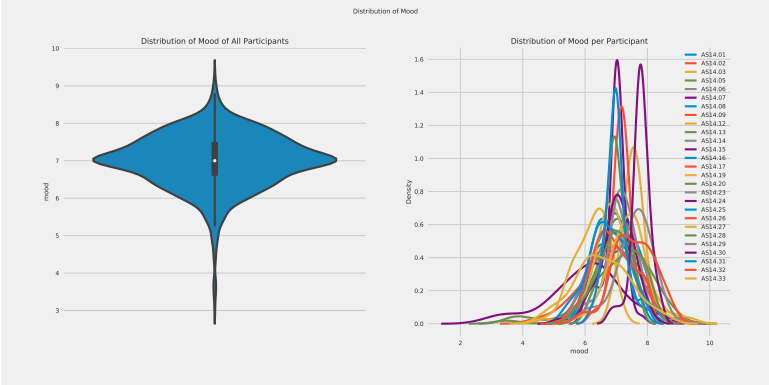

Fig. 2: Distribution of Mood

in Figure 1 have negative values, this is because these are self-reported values from the users that range from -2 to 2.

Figure 2 is the density plot of mood values from all users on the right and the violin plot of the mood of all users on the left. From these plots, we can see that the average mood ranges from 6 to 8 for all participants

Figure 3 shows the time variation of mood for any user over the period covered in the dataset. Inspection of this chart would reveal that the series is stationarity, i.e. mean and standard deviation are fairly constant. A further Augmented Dickey-Fuller (ADF) test with p-value of 0.0001 confirmed this assumption.

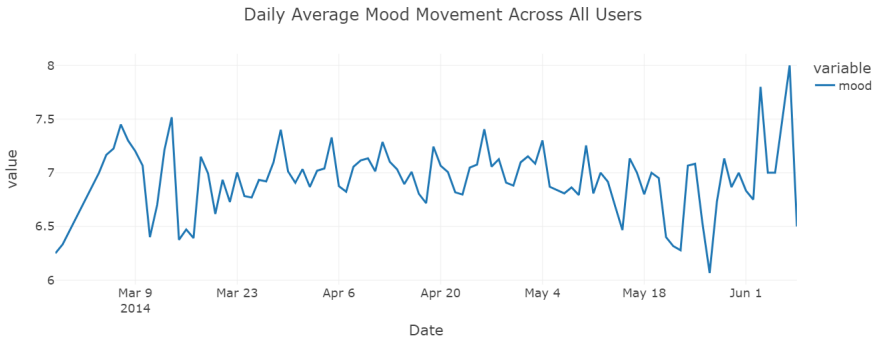

Fig. 3: Daily Average Mood Variation For All Users

Finally, we conducted a preliminary study on multicollinearity. We found that there were no strong correlations among independent variables and that for some users, Valence is strongly positively correlated with the target variable mood.

2.2 Feature Engineering

The objective is to predict the average mood of the user for the next day using the data of the user for the previous days. This indicates the input features and the target (mood) be aggregated on a day level.Ghandeharioun et al. [8] showed the usefulness of this type of aggregation in predicting depression from wearable sensor data. In addition, line plots of the input features show marked sinusoidal movements over every seven(7) day interval, hence, seven-day moving and exponential averages of these features are engineered as additional variables. The exponential averages are added in order to capture more recent events. Furthermore, the input features and the target variable, mood, are also shifted one-day and two days backward and the results added as additional features in the dataset.

The date and user level aggregations and the deletion of rows with null in the target column: mood resulted in a dataset with a 1,268 rows. 705 rows were deleted. The missing rows in the mood column were imputed because their imputation distorted the stationarity and distribution of the mood. It was decided that the original distributed in the dataset should be preserved.

Excluding user id and the date column, a total of 94 columns including existing and newly created features are in the dataset. Table 1 shows the aggregation and engineering performed on the variables.

Table 1: Feature Engineering

Method	Variables
Date and User Level Aggregation	All Variables
Date and Exponential Averaging	All Variables
Value Shifting	All features and the target (mood)

2.3 Feature Selection

Excluding user id and the date columns, the feature space is now made of 94 variables. In order to avoid the curse of dimensionality and reduce noise in the feature space [9], it is important that feature selection is done.

Feature Engineered Model For the feature engineered model, the *boruta* algorithm [10] was used in selecting the important features for the task. *boruta* attempts to curate features down to the "all-relevant" stopping point, not the "minimal-optimal" stopping point. What does "all-relevant" mean? The paper defines a variable as being relevant if there is a subset of attributes in the dataset among which the variable is not redundant when used for prediction. The important features are: appCat. office, circumplex valence, day of week, travel shift 1d, game shift 2d.

Temporal Model In the temporal model a top bottom approach to select features was implemented, ending with a selection of all the feature engineered

features except from, those with exponential averaging. With the top bottom approach, we tested the performance of the models while removing features and comparing them with a baseline. The baseline consisted on the performance of the same architecture with all the features. The features were removed if the performance of the model increased with the elimination of the feature.

3 Modelling

During this project, we implemented various supervised machine learning algorithms. The problem to solve falls in the category of supervised learning, where we aim to predict the target variable mood. The section introduces the modelling decision for the baseline model, the feature engineer model and the temporal data model.

3.1 Baseline model

The first step taken towards the solution of the prediction problem is the standard baseline model to put improvements from future models in perspective. The model implemented in this project is a regression version of the Zero Rule algorithm [11]. This model predicts the mood of the user at time, t utilizing the mood at time $t-1$.

3.2 Feature Engineered model

The feature engineered model was trained on the *boruta* recommended features using the open source, low code Pycaret framework in python. Pycaret allows the trial of many algorithms and include tunable components for outlier removal, transformation, feature interaction, performance visualization, parameter logging up, and deployment. Outlier removal, feature interaction, multicollinearity check, and normalization using Z-score were set up in the framework.

The framework allows the building of a model for each user in the dataset. There are more than 25 regression algorithms in the framework ranging from Random Forest Regressor, Gradient Boosting Regressor, Orthogonal Matching Pursuit, etc. These algorithms will be tuned and tested on the dataset of each user and the best model for each user returned.

In order to handle a completely new user, a clustering model built using K-mean clustering algorithm [12] implemented in the python package, *Sklearn* was used. Four(4) clusters were identified by the algorithm, and each user (hence each model) is mapped to a cluster id. The clustering model is used to identify the cluster a new user belongs, and this cluster information is used to select the model to be used in predicting the average mood for the next day for the new user.

3.3 Temporal model

For the temporal model, TensorFlow was used to model the architecture of the temporal model due to the modularity and easy implementation of new ideas. The data ingestion for the temporal model followed a sequence window pattern, where we predicted the label of t by inputting in the model from $t-1$ to $t-4$. The decision to take this design choice was to include in the model's input the information for the previous days and allow the models to take advantage of their memory. Two architectures were tested for this project.

LSTM Long-Short-Term-Memory networks (LSTM) are a specific type of Recurrent Neural Networks (RNN) engineered with gates to address the vanishing gradient problem. Introduced by Hochreiter 1997 [13] LSTM has become a popular alternative when dealing with sequential data. With a particular number of gates, LSTM units share information between them.

GRU Gated Recurrent Units (GRU) are RNN units, engineer in a simpler way than LSTM units. GRUs include fewer gates than its analogs LSTM, and recent work has proven them to be a useful tool when dealing with temporal data. [14]

The architecture is similar in both cases, with different final hyperparameters after tuning. Both networks consist of two layers of the specific RNN unit (LSTM or GRU) and a Fully Connected Layer. The hyperparameters present in these architectures are: Number of unit per layer, Dropout, Recurrent Dropout, learning rate and epochs.

3.4 Data Splitting

For the two model categories above (feature engineered and temporal models), the data was split into three parts: training, validation, and test set. User 'AS14.33' was taken as test set to accurately evaluate the performance of the models. By removing one user, we can test how the model generalize to unseen data, replicating what would be a real world application. For the training and validation sets, the data was sorted by time in ascending order, and for each user, a split of 80/20 was done. This means that a section of each user's data existed in both the training and validation set.

4 Results

In this section, the experiment set up for and the results from the baseline model, the feature engineered model, and the temporal model are discussed. Their performances are also compared.

4.1 Experimental Setup

As pointed out in section 3, the baseline was based on the naive zero rule algorithm, in which the next day event is predicted from the previous day record.

For the feature engineered model, the global hyperparameters to be searched for each user's model was setup in the *Pycaret*. The tuning API in *Pycaret* called *tune model* was used to get the optimum values. These parameters are presented in table 2 below. In addition, a cluster model was built using *kmeans* which help decide which model to use for a new user.

For the temporal model (both the LSTM and GRU), *TensorFlow* was used, and the hyperparameters tuned via the *keras tunner* API. The tuned hyperparameters are in table 2.

The metrics used in comparing the feature engineered model and the temporal model are mean absolute error and the mean squared error on the test set.

Table 2: Experimental Set-Up

Model	Hyperparameter	Values
GRU	Units	112
	Learning Rate	0.01
	Dropout	0.5
	Optimization	Adam
LSTM	Units	96
	Learning Rate	0.01
	Dropout	0.2
	Optimization	Adam
Pycaret Model	Cross validation Strategy	Time Series
	Cross Validation	10
	Normalization	Z-score
	Transformation	Yeo-Johnson
	Outlier-threshold	0.05
	Metric	MSE & MAE

The default tuning space in the Pycaret framework for models for the 27 users was retained and used. This default tuning space has shown consistent performance in many research and practical uses.

4.2 Baseline Model

The naive modelling approach generated results that were used as the worst case scenarios. These results in are table 5 below.

4.3 Feature Engineered Model

With the global hyperparameters mentioned in sub-section 4.1, and algorithm level hyper-parameter tuning, eight(8) algorithms were used in building models for the 27 users as shown in table 3.

It is interesting that Orthogonal Matching Pursuit algorithm [15] modelled more than 65% of the users, considering that it is also widely used in signal processing tasks. Moods as captured in the dataset have sinusoidal basis. Figure 6 shows the average actual mood versus the average predicted mood by the Pycaret models. Most points overlap, indicating high accuracy for completely new users. The Pycaret models averaged an MAE of 0.35 in the validation set, while the results for all metrics on the test set are presented in Table 4 below. These results were collected over 10 experimental runs.

Table 3: Pycaret's Best Performing Algorithms

Algorithm/Model	Number of Users Modelled
Orthogonal Matching Pursuit	18
K Neighbors Regressor	2
Decision Tree Regressor	1
Extra Trees Regressor	1
Gradient Boosting Regressor	1
AdaBoost Regressor	1
Linear Regression	1
Random Forest Regressor	1

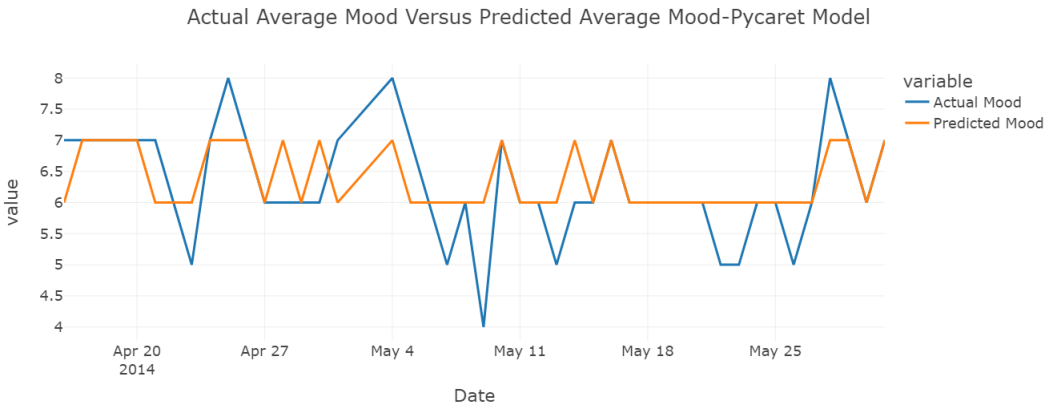

Fig. 4: Actual Versus Predicted Mood - Pycaret Model

4.4 Temporal Model

The two architectures for the temporal models were tested with the hyperparameters showed in Table 2. The sequence length selected for the window pattern was of 4 previous days. From the two architectures, GRU with MAE 0.48 and MSE 0.5 outperformed LSTM with MAE 0.54 and MSE 0.58.

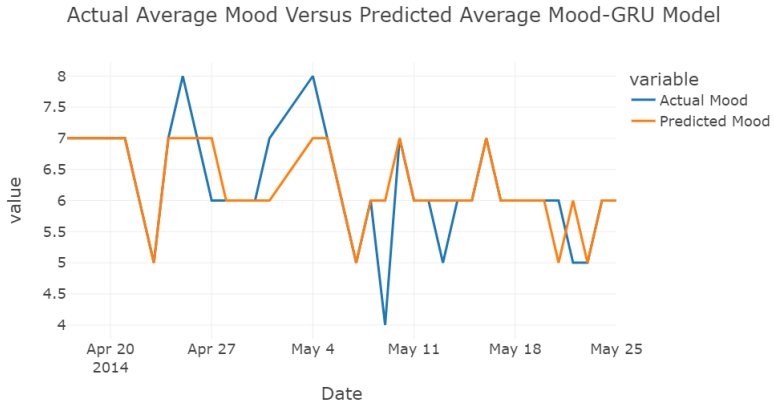

Fig. 5: Actual Versus Predicted Mood - GRU Model

4.5 Feature Engineered Model Vs Temporal Model

Table 4 shows the comparison of the best performing temporal model, GRU,the Pycaret family of models, and the baseline using three (3) metrics: MAE, MSE, and RMSE. Both Pycaret and GRU outperformed the baseline across all metrics by an average of 70%. Pycaret outperformed the GRU model across the three metrics by an average of 12.5% and achieved greater stability as indicated by the lower standard deviations. The better performance of Pycaret maybe due to the low correlations/similarities among moods for different users, as indicated by the n-way ANOVA test p-value of $2.5e-325$ and a low Pearson correlation coefficient of 0.042.

Table 4: Performance Comparison

Model	MAE	MSE	RMSE
Baseline	1.41	0.74	1.18
GRU	0.48±0.04	0.50±0.03	0.69±0.05
Pycaret	0.42±0.001	0.47±0.001	0.68±0.002

5 Conclusion

From the results obtained from the several experimental runs, machine learning systems with the right data can model many real life problems. The slightly better performing Pycaret family of models would be more suitable for an application higher accuracy is important. This maybe, however, come with a technical debt in terms of computational costs and more difficult implementation. The GRU model has lower implementation complexity and maybe more suitable for rapid prototyping. GRU architecture outperformed LSTM. One reason behind this performance could be the lower capacity to be overfitting the dataset by the GRU. Since several features were given as an input, it could be easily the case. However, since the models were tested in an unseen user, we can be sure this is a good representation of the generalization power of the models.

References

[1] Statista, *Smartphone penetration worldwide 2020*, Dec. 2021. [Online]. Available: https://www.statista.com/statistics/203734/global-smartphone-penetration-per-capita-since-2005/.

[2] R. Harper and J. Southern, "A Bayesian Deep Learning Framework for End-To-End Prediction of Emotion from Heartbeat," *IEEE Transactions on Affective Computing*, pp. 1–1, 2020. DOI: 10.1109/taffc.2020.2981610.

[3] G. De Choudhury, S. Counts, and E. Horvitz, "Predicting depression via social media," *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2013. DOI: 10.13140/RG.2.2.29631.3600.

[4] N. Jaques, S. Taylor, A. Azaria, A. Ghandeharioun, A. Sano, and R. Picard, "Predicting students' happiness from physiology, phone, mobility, and behavioral data," *2015 International Conference on Affective Computing and Intelligent Interaction (ACII)*, 2015. DOI: 10.1109/acii.2015.7344575.

[5] ——, "Multi-task, Multi-Kernel Learning for Estimating Individual Wellbeing," *MIT Media Lab*, p. 7, 2015. DOI: 10.1109/taffc.2020.2981610.

[6] S. Taylor, N. Jaques, E. Nosakhare, A. Sano, and R. Picard, "Personalized Multitask Learning for Predicting Tomorrow's Mood, Stress, and Health," *IEEE Transactions on Affective Computing*, vol. 11, no. 2, pp. 200–213, 2020. DOI: 10.1109/taffc.2017.2784832.

[7] K. Taunk, S. De, S. Verma, and A. Swetapadma, "A Brief Review of Nearest Neighbor Algorithm for Learning and Classification," *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, 2019. DOI: 10.1109/iccs45141.2019.9065747.

[8] J.-M. Moreno-Roldán, M.-Á. Luque-Nieto, J. Poncela, and P. Otero, "Objective Video Quality Assessment Based on Machine Learning for Underwater Scientific Applications," *Sensors*, vol. 17, no. 4, p. 664, 2017. DOI: 10.3390/s17040664.

[9] N. Venkat, "The Curse of Dimensionality: Inside Out," *Sensors*, 2018. DOI: 10.13140/RG.2.2.29631.36006.

[10] M. B. Kursa, A. Jankowski, and W. R. Rudnicki, "Boruta – A System for Feature Selection," *Fundamenta Informaticae*, vol. 101, no. 4, pp. 271–285, 2010. DOI: 10.3233/fi-2010-288.

[11] J. Brownlee, *How To Implement Baseline Machine Learning Algorithms From Scratch With Python*, May 2020. [Online]. Available: https://machinelearningmastery.com/implement-baseline-machine-learning-algorithms-scratch-python/.

[12] S. . Mannor, X. . Jin, J. . Han, *et al.*, "K-means clustering," Dutch, *Encyclopedia of Machine Learning*, pp. 563–564, 2011. DOI: 10.1007/978-0-387-30164-8_425.

[13] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735.

[14] P. T. Yamak, L. Yujian, and P. K. Gadosey, "A Comparison between ARIMA, LSTM, and GRU for Time Series Forecasting," *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*, 2019. DOI: 10.1145/3377713.3377722.

[15] Wikipedia contributors, *Matching pursuit*, Apr. 2022. [Online]. Available: https://en.wikipedia.org/wiki/Matching_pursuit.

[16] *PyCaret — pycaret 2.3.5 documentation*, 2022. [Online]. Available: https://pycaret.readthedocs.io/en/latest/index.html.