Using Bi-LSTM and XGBoost for mood Detection on a mobile app Dataset

Yuyu Bai—2732696 , Summer Xia—2703936 , Bowen Lyu—2727574

Vrije Universiteit Amsterdam

1 Introduction

Time series forecasting is a valuable forecasting method with applications in a variety of industries. Sales forecasting, for example, is necessary in the retail business, on the one hand, to plan sales in advance; on the other hand, accurate sales forecasts are also required to assist supply chain management. The main goal of this project is to construct a mood forecasting model using the time-series data obtained from a mobile phone app. Traditional time-series forecasting relies on linear models ARIMA or exponential smoothing, which frequently fail to meet the demands of increasingly complex real-world generative operating environments. In recent years, deep learning has risen to prominence as a result of its excellent non-linear feature characterization capabilities combined with time series prediction findings.

In this paper, we will use a temporal model and machine learning models to solve a classic time series problem. ”How well do Xgboost, SVR and Bi-LSTM perform in the study of multivariate time series regression problems?” is the research question for this work.

The structure of the paper is as follows: We begin with the data pre-processing method , which explains how we handled with our data and used it to generate various features, as well as the several models we'll be using in this section.Some scientific literature supports are also given here. Next is the results and discussion section 4, where we examine and compare the results we obtained from various models. Finally, a summary of what we accomplished in this project are given in conclusion.

2 Method

2.1 Problem definition

There are total 376,912 records from 27 persons between 26-02-2014 to 08-06-2014, containing sensory data regarding the user's behavior. Our goal is to create a prediction model that anticipates the user's average mood for the following day. First and foremost, we must group these records by dates and users. This issue can be viewed as a regression issue. We determine the average mood as the target response for each person per day, utilizing other information from prior days as features. The attributes can be obtained from the user's behaviours in previous

7 days (assuming that the user's mood varies with the day of the week, 7 is a good time span. However, since a large time window will cause sample deduction in the LSTM model, we only employ 5 days of data in the Bi-LSTM model).

2.2 Feature engineering

Attributes generation When it comes to feature selection, we've taken a bottom-up approach. To put it another way, we brainstorm a big number of relevant features before deciding on the most crucial ones to test. Initially, we develop the four types of attributes listed below.
1.Aggregate the history
The raw data is directly obtained from the mobile application as a record or log and cannot be used in machine learning models. Rolling time window aggregation is a popular method of feature conversion. Based on the research and comprehension of the data, the time window is optional. In this scenario, we'll use three sorts of time windows:

- a)The most common time window is one day, which generates aggregated values for each day's data. Since our labels are also aggregated on a daily basis, saying we need to predict the average mood value for a particular day, this aggregation method is reasonable.
- b)As stated in problem definition part, data from the last 7 days is used to predict future labels. Therefore, the last 7 days can also be used as a time window to generate summary information of all data from the last 7 days.
- c)According to our understanding of the data, playing with the smartphone at night could be a sign of insomnia and a bad mood. We therefore artificially defined 0-6am as the time of night and calculated the aggregated values for night for each day.

2.Statistical features
As these are all features obtained from the mobile app, most of them have obvious statistical significance, such as the length of use of communication software, the length of use of gaming software, and the length of use of the screen, some basic statistical values can be calculated. For example, we can calculate the total number of hours used per day for the past n days, the average number of hours used per day, the number of times the screen was used, and the maximum number of hours used per day (due to missing data or the fact that some people do not use certain types of apps, most of the minimum values are zero, so there is no point in counting the minimum values). It is also possible to calculate the overall statistics for the last n days, the average, the maximum and the sum.
3.Periodical features
In general time-series data models, cyclical seasonal features are often likely to perform well. For example some of the labels may change seasonally or behave differently at the beginning of the month than the time at the end. The data in this dataset is range from 27th ,Feb to 8th, June. We have not developed corresponding seasonal and monthly features due to the short period. Only a

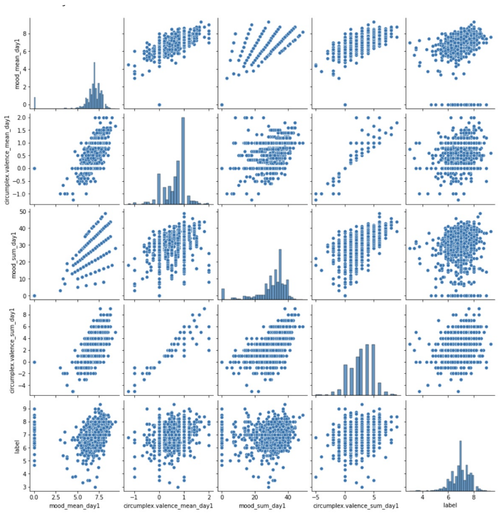

Fig. 1. Distribution of data

week-related feature "weekday", which indicates the day of the week are used as our feature.

4.Features related to trend

In time series features, the trend of the data may have a greater impact on the predicted value, and it is more difficult for ordinary machine learning to learn the trend of the data. Therefore, we have also developed some features related to trends. For example, taking screen time as an example, we can take the time spent looking at the screen in the past day and divide it by the total time spent looking at the screen in the past three days, if this value is greater than 1/3, it can indicate an increasing trend in the time spent looking at the screen. Similarly, we developed the feature of dividing sum of one day by sum of three days and three days by seven days to be used as trend features.

Attributes selection We generated a total of 1255 features. However, some features are not very relevant to the mood or have too many missing values. These features should be deleted. The linear correlation coefficient are used to rule out the unrelated features. Features with a correlation coefficient close to zero (-0.1-0.1) were also removed. Some important features can also be selected by data analysis in section 2.3 As the Xgboost model can also be used for feature selection on demand, the 192 selected features are used here for training.

2.3 Data analysis

Initial analysis Among the 19 initial variables, 15 variables record the user's usage time/number of times in different scenarios, and the remaining four are score variables, including the user's rating of mood, that is our target variable. The distribution of user entries, variables, mood value and time are shown in figures below.

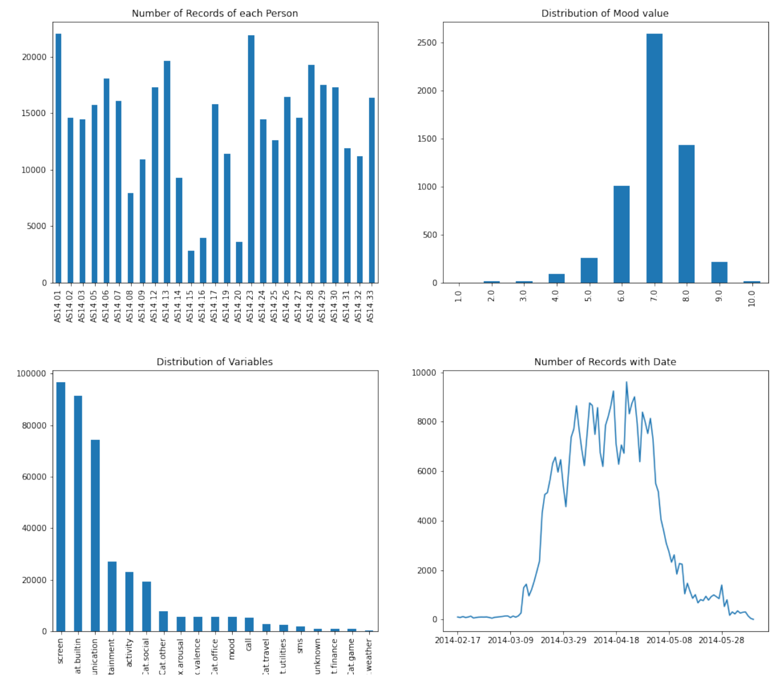

Fig. 2. Distribution of data

We go over the numerical details of each variable, looking for possible correlations between them. Here, we document some of the analyses we find valuable during data investigation.

Duration of screen activity (time) is in the range 0-374. For statistics, we divide the user's screen usage time into 10 segments, with 90% of the users' usage time being less than 150 seconds. Taking 150 as the cut-off point, the user's average mood increases slightly with the increase in usage time, and then begins to decline until 224. Although there is an increase tendency again(¿224), the amount of data is too small compared to the former segments, therefore no definite conclusion can be drawn.

Weekday0 represents Monday and Weekday6 represents Sunday. Users recorded almost equal number of app usage records per day, except on Tuesday because of a significant drop. We also found that the average value of Mood were relatively low from Monday to Wednesday, but it started to rise significantly after

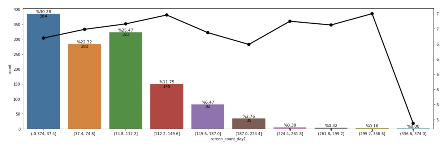

Fig. 3. Screen time Effect to Mood value

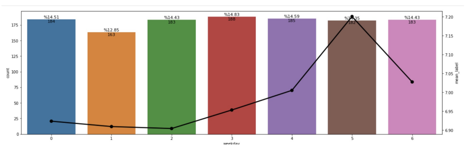

Fig. 4. Weekday distribution and effect

Wednesday with a peak on Saturday, and then dropped rapidly on Sunday and Monday in a row.

Likewise, we also looked at app-related variables. In terms of the correlation of these variables, they are not particularly clearly linked, but looking at the graphs we can see curves with slight trends, such as a small positive correlation between the average mood value and the time users spend on communication apps. The similar curves are shown for applications in the game, entertainment and travel categories.

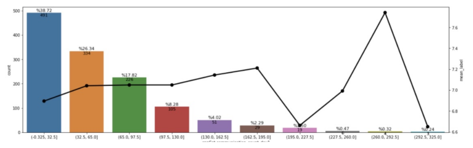

Fig. 5. Communication distribution and effect

2.4 Model

Support Vector Regression Support vector regression(SVR) is a robust machine learning techique which is capable of interpreting linear and nonlinear patterns and strong generalization ability.[3] Different from the traditional statistical learning theory, the SVM is based on the principle of structural risk minimization, which can effectively solve the problem of overfitting, and so has a prospect for temporal data forecasting[6]. By reading the relevant literature, we find a way to separates time series into linear part and nonlinear part, then predicts the two parts respectively, and finally integrates the two parts to forecast.[4]

In our SVR implementation, we divide our features into two parts. One is the stable part, which contains long-term trends, such as the average value of various app usage within a week; the other is the unstable part, which includes changes in mood values and activity scores. The SVR model predicts the two parts separately, and finally integrates the two parts to obtain the prediction result, which improves the prediction efficiency to a certain extent.

XGBoost Besides SVR, another machine learning technique we use is XGBoost. XGBoost is an efficient implementation of gradient boosting for classification and regression problems. It performs well in a wide range of predictive modeling tasks[1], appears frequently in kaggle competitions, and has also been widely employed in enterprises in recent years. In this case, we implement our XGBoost model to forecast user's mood value by transforming the time series dataset into a supervised learning dataset using a sliding-window representation. A significant advantage of using XGBoost is that it has great feature extraction capabilities. The importance of XGBoost output variables can be used to guide the analysis process and possibly help train more complex models such as RNN, LSTM, etc.[7]

Bidirectional Long Short-Term Memory The TensorFlow frame is used to implement the Bi-LSTM network. Compared with normal LSTM network, the Bi-LSTM network allowing any neural network to store sequence information in both ways, either backward or forward[5]. Based on previous work, Bi-LSTM has shown a clear advantage in multivariate time series regression problem[2]. As shown in Figure 5, the model used in the project contains two Bi-LSTM layers with hidden layers of size 400 and 300 respectively, and multiple dense layers to produce the output of size 1.

Benchmark In order to provide a compared model, we implement a benchmark model which predicts the average mood of the next day through the value of the last day, meaning $mood_d = mood_{d-1}$. We generate the test set with the same way of machine learning and LSTM for giving a better performance.

3 Experiment

3.1 Sample generation

We construct the three dimension variables and response tuples for Bi-LSTM models with the history window set to 5 and the horizon set to 1 per user. This means that the model will forecast the mood of the following day based on the qualities of the previous five days. The number of records may be reduced due to the presence of a slicing window. The sample is the same as we defined in section 2 for machine learning models.

3.2 Missing Values

Many missing values can be found in the dataset created by section 2.3. Missing values might be of two types. To begin with, the records of various users may have different attributes. This might be due to a failure to record the necessary information. All of the models, however, need that their input format be consistent. As a result, we generate a synthesis of all features. We populate the missing attribute columns for each user with all zeros, assuming that zero represents no measurement of that attribute.

Second, some of the average attributes value in one day for each user is NaN. We can't just remove records with NaN values since a high percentage of records have missing values, and removing them will result in temporal gaps and the loss of important data. In Bi-LSTM models, we use a mix of backward and forward fill to interpolate the missing value with the nearest not-null value. This is due to the Bi-LSTM model's requirement for temporal data continuity and as few outliers as feasible. The missing data for XGB are all filled with zeros since XGB has high robustness and zero can represent no activity that day.

3.3 Train,validation,test data set split

Furthermore, the data is divided into training, validation, and test sets using a 6:2:2 ratio to prevent over-fitting and conduct parameter adjustment. The data comprises of time series data from several users, and the amount of data from a single user is insufficient to train for the Bi-LSTM model. We can't aggregate users' data by days since their time spans overlap, as shown in section 3.3. As a result, each user's training, validation, and testing are created following the way described in section 3.1 and then concatenated in order.

3.4 Normalization

The Bi-LSTM places a higher importance on attributes with larger values. As a result, the merged dataset is normalized using MinMaxScaler, a tool that can scale data between 0 and 1. Furthermore, as compared to StandardScaler, the MinMaxScaler performs better in training and testing. This is most likely due to a lack of negative values in the dataset. Because XGB is made up of decision trees, there is no need to conduct any normalization on the dataset.

3.5 Parameter tuning

Before training, the data is shuffled and sliced using the batch size for Bi-LSTM models. The parameters we chose for the Bi-LSTM models after grid search are shown in table 1. Meanwhile, because the Bi-LSTM model is trained using gradient descent, early stopping is used to avoid overfitting. The early stopping patience is set to 10, which means that the training process will be terminated if the validation loss begins to grow in 10 epochs.

4 Result and discussion

4.1 Evaluation

The change in training loss and validation loss during the training process using the Bi-LSTM model is shown in Fig. 7. To predict using the value of the

Parameters	value
batch size	4
buffer size	8
loss function	msle
optimizer	adam
activation function	tanh
epoch	100
steps per epoch	10

Table 1. parameters of the Bi-LSTM models

Parameters	value
tree depth	3
colsample	0.8
subsample	0.8
eta	0.1
early stopping	50

Table 2. parameters of the XGB models

test set created in section 3.3, the Bi-LSTM model and benchmark model are used. The mean squared error, mean absolute error, and square root of the mean squared error are used to study the performance of these models further, and the results are displayed in Table 3 (all these metrics are computed with all data being normalized).

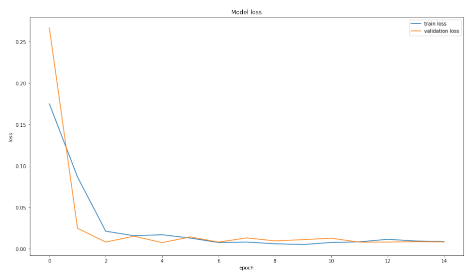

Fig. 6. Train and validation loss curves for LSTM.

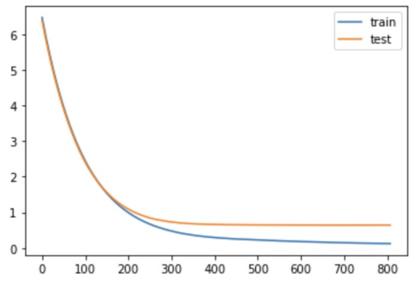

Fig. 7. Train and validation loss curves for xgb.

4.2 Result

During the training of Bi-LSTM models, the model converges swiftly after 2 or 3 epochs, despite the models stopping extremely early and having modest steps each epoch. For the dataset of this research, this demonstrates the Bi-LSTM's high learning capabilities. The final train loss and validation loss are both less than 0.01, indicating a satisfactory training effect for the model. The difference between the train loss and the validation loss, on the other hand, is not substantial, indicating that the model is not overfitted in the end.

The loss function's value decreased consistently while the training set and test set for XGB were processed. The XGB training curve is much smoother than

Metrics	Bi-LSTM	Benchmark	XGB	SVR
MSE	0.413	0.560	0.434	0.445
MAE	0.486	0.561	0.471	0.510
RMSE	0.643	0.748	0.651	0.699

Table 3. Performance comparison between XGB, Bi-LSTM and benchmark using metrics

Models	p-value
Bi-LSTM vs Benchmark	0.1193
XGB vs Benchmark	0.4175
SVR vs Benchmark	0.4262

Table 4. p-values of T-test between the benchmark, Bi-LSTM, XGB and SVR

the Bi-LSTM training curve. This is due to the fact that Bi-LSTM learns via iteration, whereas XGB is taught by adding decision trees to reduce residuals.

In terms of the test set, the Bi-LSTM projected values, the benchmark model predicted values, and the actual values all follow a similar pattern. The Bi-LSTM forecasts, on the other hand, are more consistent than those of the real value and benchmark models. This is likely related to the fact that we train the model using the entire dataset, yet the range of fluctuation in a user's mood might be rather wide.

Both Bi-LSTM and XGB beat benchmark's results, as shown in Table 3, with Bi-LSTM being the best of the three models we constructed. This demonstrates that when dealing with multivariate time series regression problems, the current temporal model has an edge over traditional machine learning methods. Two-sample t-tests are used to analyze the differences in model performance further. Table 4 shows the test results for the difference in mean squared error of various models. It is clear that the performance of Bi-LSTM and benchmark models are not significantly different. It's possible that this is due to a lack of data. Due to inadequate training and test sets, the model does not match the data very well. The t-test findings may also be less accurate due to the minimal number of data in the test set.

4.3 Important features

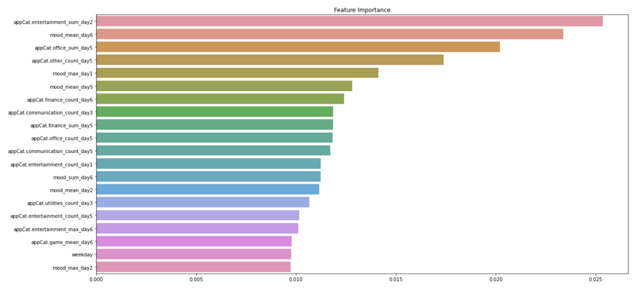

Fig. 8. Feature importance ranking generated by XGB

The top 20 relevant features picked by XGB are shown in Figure 8. These features are divided into three groups. The first is the weekday feature, which, as we said in section 2.3, since mood seems to shift depending on the day of the week. In addition, the figure has six mood-related elements, illustrating that mood data is stationary and is impacted by previous moods The remaining features include the use of various applications, such as entertainment apps, which could have a significant impact on mood changes.

5 conclusion

We pre-process the data, modify the data display format, and create a multivariate time series data in this project. Each record contains a variety of data about a user on a specific day. We discovered that this is a multivariate time-series regression problem after analyzing the data. As a result, we picked Bi-LSTM as a temporal model representation and XGB as a standard machine learning technique representative, respectively. The benchmark model that predicts the mood on the following day just like the previous day is also attributable to the strong smoothness of the time series data in this project.

Missing value interpolation, normalization, and train-test splitting are all performed on the pre-processed data for the Bi-LSTM model. The model is then trained using customized parameters. Both training and validation loss is less than 0.01 percent (result got after normalization). The essential time series features are created for XGB, the obtained data is used for training, and appropriate parameters and features are picked to obtain the final model.

To be able to test the performance of the individual models, different models were predicted using the same test set. The results are also compared using multiple metrics. On the test set, Bi-LSTM had the best performance, followed by XGB, both of which were better than benchmark. Proving that both are somewhat effective to some extent, but the effect is not significant. This could possibly be due to the smaller amount of data.

Different models were forecasted using the same test set to examine the performance of the distinct models. Multiple measures are used to compare the findings. Bi-LSTM had the best performance on the test set, followed by XGB, both of which outperformed the benchmark. demonstrating that both are effective, although the impact is not significant. This is probably due to the reduced amount of data.

The Bi-LSTM, on the other hand, is a more complicated model that can extract relevant information from raw data without the need for significant feature engineering. XGB, on the other hand, is a more typical tree model whose success is largely dependent on feature quality and necessitates complex feature engineering. Its training, on the other hand, is simpler and uses less computer resources.

Due to the minimal amount of data in this study, additional research will be conducted. As a result, Bi-LSTM did not produce satisfactory training outcomes. For large-scale time-series data, this Bi-LSTM model may be better suited.

References

1. Raza Abid Abbasi, Nadeem Javaid, Muhammad Nauman Javid Ghuman, Zahoor Ali Khan, Shujat Ur Rehman, et al. Short term load forecasting using xgboost. In *Workshops of the International Conference on Advanced Information Networking and Applications*, pages 1120–1131. Springer, 2019.
2. Shengdong Du, Tianrui Li, Yan Yang, and Shi-Jinn Horng. Multivariate time series forecasting via attention-based encoder–decoder framework. *Neurocomputing*, 388:269–279, 2020.
3. R Amaro e Silva, LCC Teixeira da Silva, and MC Brito. Support vector regression for spatio-temporal pv forecasting.
4. Kunhui Lin, Qiang Lin, Changle Zhou, and Junfeng Yao. Time series prediction based on linear regression and svr. In *Third International Conference on Natural Computation (ICNC 2007)*, volume 1, pages 688–691. IEEE Computer Society, 2007.
5. Farah Shahid, Aneela Zameer, and Muhammad Muneeb. Predictions for covid-19 with deep learning models of lstm, gru and bi-lstm. *Chaos, Solitons & Fractals*, 140:110212, 2020.
6. Saurabh Suradhaniwar, Soumyashree Kar, Surya S Durbha, and Adinarayana Jagarlapudi. Time series forecasting of univariate agrometeorological data: A comparative performance evaluation via one-step and multi-step ahead forecasting strategies. *Sensors*, 21(7):2430, 2021.
7. Naiju Zhai, Peifu Yao, and Xiaofeng Zhou. Multivariate time series forecast in industrial process based on xgboost and gru. In *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, volume 9, pages 1397–1400. IEEE, 2020.