Assignment 1: DataMining Practice and Theory

Konstantinos Pasatas 2803568 Aristeidis Charizanis 2803717 Athanasios Katranis 2803183

Vrije Universiteit Amsterdam, De Boelelaan 1105, 1081 HV Amsterdam, Netherlands
Team 77
k.pasatas@student.vu.nl
a.charizanis@student.vu.nl
a.katranis@student.vu.nl

Introduction

This document outlines the group project for the Data Mining Techniques course conducted by Konstantinos Pasatas, Aristeidis Charizanis and Athtanasios Katranis at the Vrije Universiteit Amsterdam. The project utilizes a dataset from smartphone applications monitoring mental health, specifically focusing on predicting the next day's mood from historical sensory data and user interactions. Following the CRISP-DM methodology, this report documents the entire process, from data preparation to modeling, highlighting the use of various data mining techniques discussed throughout the course.

1 Data Preparation

1.1 TASK 1A: Exploratory Data Analysis and Basic Preparation

1. **Raw Data/Pivoted:** The dataset used for this assignment is collected from smartphone applications, customized to support mental health, focusing on depression. These applications track a variety of user behaviours throughout sensors, and also periodically collect user ratings of their mood. The dataset contains ID's, reflecting the user the measurement originated from, a timestamp (this implies that the data consist of time-series observations), detailing when the data was collected, and specific variables describing aspects of the user's interaction and mood. The dataset has a long format, which leads to the first preprocessing step of converting it to a wide format. This process involved utilizing the "id" and "time" columns as index, the "variable" column as column headers, and the "value" column as the cell values. This followed by resetting the index to flatten the DataFrame named Pivoted Data and restore "id" and "time" as regular columns. This new DataFrame consists of 358926 rows and 21 columns, of which 3 columns contain categorical variables, 2 columns binary variables, and 14 columns continuous variables.
   Examining the categorical variables of interest within the dataset (Fig.1), we found that the mood variable shows a concentration of entries with scores between 6 and 8 on a scale of 1 to 10. This indicates that the majority of participants were experiencing relatively positive mood states. Additionally, the variables circumplex.arousal and circumplex.valence, exhibit a distribution that spans from -2 to 2. Regarding circumplex.valence, it is worth mentioning that there is a clustering of data points within the range 0 to 1, suggesting a slope towards the most positive affective value.
   When we look at how people use smartphone apps, we notice a significant variation in the amount of time spent on different categories (as shown in Fig.2). The data reveals that the categories related to entertainment and social interaction have the highest usage times (Table 2). In contrast, applications related to finance, travel and weather are characterised by significantly fewer entries, signalling a lower frequency of use in these areas. Furthermore, we see that for appCat.buildin, an attribute measured in time, has negative values, which could be a sensor error, and had to be removed.
   Overall, the dataset exhibits a distribution with 'mood' ratings skewed towards the higher end, showing a tendency for participants to report positive emotional states. 'Circumplex.arousal' and 'circumplex.valence' scores are mostly concentrated in the positive quadrant, suggesting states of moderate arousal and positive emotional valence. In terms of smartphone usage, there is a clear disparity in app categories, with entertainment and social interaction apps outperforming those for finance, travel and weather. In addition, the majority of the variables have right-skewed distribution. These indications, combined with the high number of missing values (Table 1), suggest the presence of outliers, deviations from normality, and the need to examine the transformations.

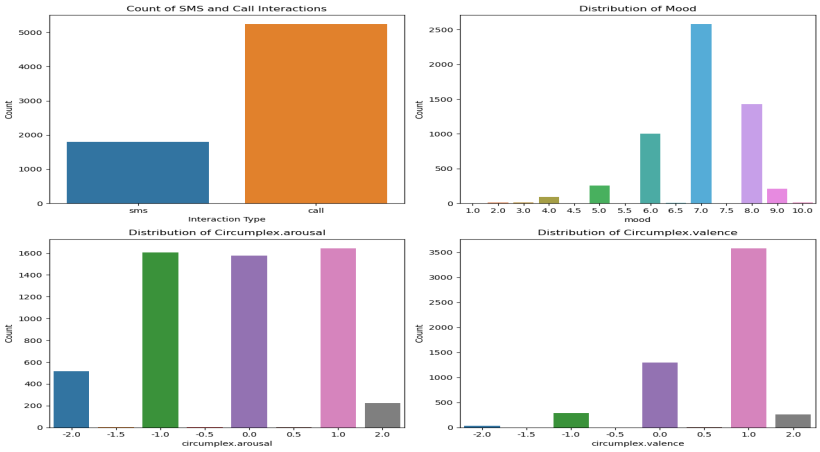

Fig. 1: Categorical data plot

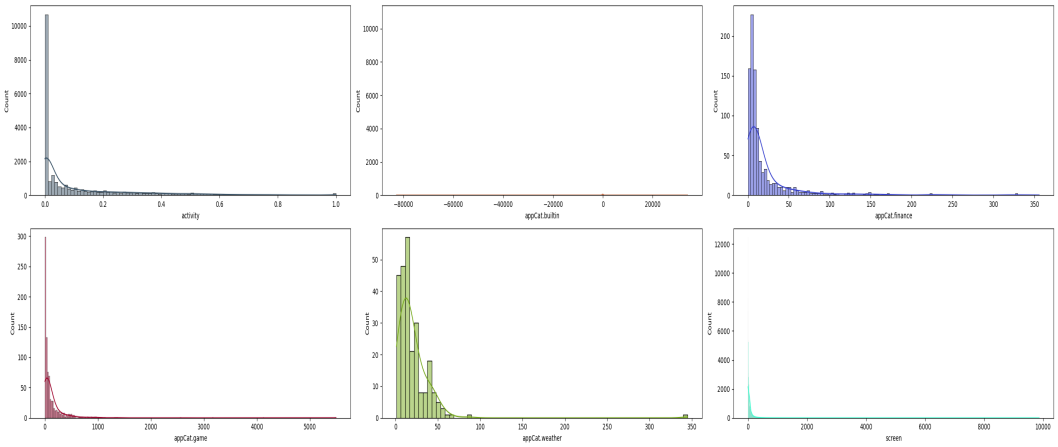

Fig. 2: Continuous data plot

2. **Aggregated Data:** Following the brief EDA on the raw and Pivoted dataset and the insights gained, we decided to proceed with data aggregation and the creation of a daily dataset. In our approach to aggregate the dataset into daily summaries, we considered the subjective nature of mood and app usage, which vary significantly across different users. For each user identified by their unique ID, we aggregated app usage durations and communication events using daily sums to capture total engagement per day. At the same time, we computed the daily mean for 'mood', 'circumplex.arousal', and 'circumplex.valence' based on timestamps, which reflected the average psychological state for each individual on any given day. Ultimately, a dataset named Daily_data was created which consists of 1972 rows and the same number of columns as the Pivoted Dataset. This approach, ensures that the individual variability in data is respected. Furthermore, this method not only made analysis easier but also addressed the challenge of handling sporadic missing values, thus enhancing the overall data quality. Influenced by similar methodologies in related research [1], this approach allowed us to analyze the daily patterns of mood while accommodating the personal nature of the data. As a result, this approach enabled us to reduce the number of missing values in the dataset, preventing potential bias from a high number of value imputation, and structured the dataset with the specific goal of predicting the mood of the next day. The difference in the amount of missing values is shown in Table 1. After aggregating our dataset into daily summaries, we continued to face challenges with missing values across several features. In order to address this, we chose to selectively discard certain features, guided by the prevalence of missing data and their demonstrated effect on mood, as described in the relevant literature [2][3]. Specifically, we utilized Table 1, where the 'Daily Data' column indicated the extent of missing values (higher numbers signify more missing data), and the 'Missing with Mood' column showed how often each variable's data was missing when mood data was also absent. Features that were frequently missing alongside mood data, and thus potentially less reliable for our analysis, were prioritized for exclusion. As a result, we dropped the following features due to their high missing values and lesser relevance to our study's focus on mood fluctuations: 'appCat.finance', 'appCat.game', 'appCat.office', 'appCat.travel', 'appCat.unknown', 'appCat.utilities', 'appCat.weather'. This approach helped us refine our dataset, focusing on more consistent variables to enhance our analysis of daily mood fluctuations.
3. **Insights:** The heatmaps for daily mood and activity scores (Fig3), alongside those of other variables in our analysis, showcase a synchronization in the trends of the data set. In general, most variables display similar patterns of user engagement, with a consistent presence of data for the central period and a notable absence of data at the beginning and end of the series. These gaps, evident in both self-reported mood and sensor-driven variables, may indicate periods of non-use or non-engagement with the app functionalities. With such uniform absence, especially in the mood data which requires active user input, implies that users may initially refrain from using the app's features or gradually reduce their interaction with them over time. You can find comprehensive details about the features of both the Pivoted and Daily datasets in Table 1,2.

1.2 TASK 1B: DATA CLEANING

Daily Dataset Cleaning Following the Exploratory Data Analysis (EDA), it's evident that the dataset required comprehensive cleaning to fix issues such as missing values and outliers. This cleaning process was essential to ensure the reliability and accuracy of subsequent analyses, and models.

As mentioned earlier, a noticeable pattern was observed in the distribution of values across most variables, coinciding with the central calendars of our time-series data. This observed coherence in value distribution

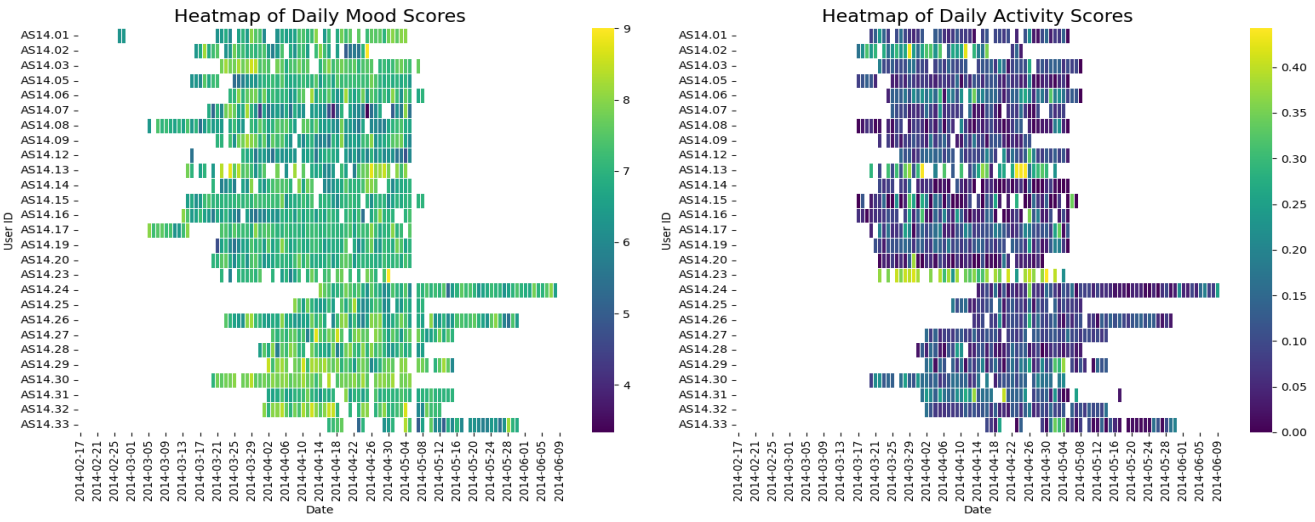

Fig. 3: Heatmap of Daily mood and activity scores for all users

Table 1: Comprehensive Data Summary and Comparison of Missing Data Proportions

Category	Pivoted Data (%)	Daily Data (%)	Missing with Mood (%)
Screen	73.09	766 (38.84)	666 (86.95)
Call	98.54	634 (32.15)	107 (16.88)
SMS	99.50	1238 (62.78)	399 (32.23)
Circumplex Arousal	98.44	705 (35.75)	705 (100.00)
Circumplex Valence	98.47	707 (35.85)	705 (99.72)
Activity	93.60	785 (39.81)	667 (84.97)
AppCat Builtin	74.57	778 (39.45)	667 (85.73)
AppCat Communication	79.31	790 (40.06)	669 (84.68)
AppCat Entertainment	92.44	1123 (56.95)	678 (60.37)
AppCat Finance	99.74	1765 (89.50)	704 (39.89)
AppCat Game	99.77	1779 (90.21)	701 (39.40)
AppCat Office	98.43	1696 (86.00)	699 (41.21)
AppCat Other	97.87	853 (43.26)	688 (80.66)
AppCat Social	94.67	985 (49.95)	679 (68.93)
AppCat Travel	99.21	1543 (78.25)	694 (44.98)
AppCat Unknown	99.74	1709 (86.66)	696 (40.73)
AppCat Utilities	99.31	1539 (78.04)	696 (45.22)
AppCat Weather	99.93	1859 (94.27)	703 (37.82)

across numerous variables, especially during the middle phase of our time series, guided our approach to data cleaning. Given this similarity in patterns, we made the decision to exclude the continuous initial and final days from our dataset. The rationale behind this choice stems from the consideration that imputing substantial gaps, particularly for a subjective measure like mood, could introduce significant biases. This subtraction helps to ensure that our analysis isn't influenced by artificially calculated values that may not accurately reflect the participants' true emotional states. As a result, and in order to ensure data integrity, we excluded entries prior to March 14, 2014, and after March 16, 2014.

1. **Outlier/Anomaly Detection:** As we delved into the task of anomaly detection within our time series dataset, we employed the Isolation Forest algorithm[4], drawing inspiration from a comprehensive study outlined in the survey paper[5]. Isolation Forest is particularly well-suited for this purpose due to its efficacy in identifying outliers by isolating anomalies instead of constructing a profile of normal observations. This model assumes that anomalies are few and distinct. Therefore, this method facilitates easier separation from the norm, leading to shorter paths in the tree structure of the algorithm. We focused on the subjectivity of mood as a variable, tailoring the identification of anomalies to the range of mood reports of each individual. This user-centred approach allowed us to avoid mislabeling outliers, where mood ratings that may be normal for one user could be considered abnormal for another one. Our adoption of Isolation Forest is aims to demonstrated balance between detection accuracy and computational efficiency. This makes it ideal for our dataset, which has a lot of variation and instances that don't follow the usual pattern. Additionally, we set the contamination parameter, which defines the proportion of outliers expected in the dataset, at 0.01. This value was determined to be optimal after several experimental iterations, ensuring that each

Table 2: Summary Statistics for Variables from the Pivoted and Daily Datasets

Variable	Pivoted Dataset					Daily Dataset				
	Count	Mean	Std Dev	Min	Max	Count	Mean	Std Dev	Min	Max
activity	22965	0.1	0.2	0.0	1.0	1091	0.1	0.1	0.0	0.4
appCat.builtin	91288	18.5	416.0	-82798.9	33960.2	1095	1249.0	1798.3	0.9	17416.3
appCat.communication	74276	43.3	128.9	0.0	9830.8	1084	2552.6	2169.1	1.1	10223.1
appCat.entertainment	27125	37.6	263.0	-0.0	32148.7	773	1077.2	1216.8	1.0	6324.9
appCat.finance	939	21.8	39.2	0.1	355.5	-	-	-	-	-
appCat.game	813	128.4	327.1	1.0	5491.8	-	-	-	-	-
appCat.office	5642	22.6	449.6	0.0	32708.8	-	-	-	-	-
appCat.other	7650	25.8	112.8	0.0	3892.0	1026	139.1	223.2	2.0	1896.2
appCat.social	19145	72.4	261.6	0.1	30000.9	903	1306.4	1462.3	0.7	7132.6
appCat.travel	2846	45.7	246.1	0.1	10452.6	-	-	-	-	-
appCat.unknown	939	45.6	119.4	0.1	2239.9	-	-	-	-	-
appCat.utilities	2487	18.5	61.0	0.2	1802.6	-	-	-	-	-
appCat.weather	255	20.1	24.9	1.0	344.9	-	-	-	-	-
call	5239	1.0	0.0	1.0	1.0	1267	3.8	3.1	1.0	16.0
circumplex.arousal	5582	-0.1	1.1	-2.0	2.0	1165	-0.1	0.6	-1.8	1.5
circumplex.valence	5474	0.7	0.7	-2.0	2.0	1163	0.7	0.4	-0.7	1.8
mood	5628	7.0	1.0	1.0	10.0	1165	7.0	0.7	3.3	9.0
screen	96578	75.3	253.8	0.0	9867.0	1107	5708.1	4262.8	0.5	19381.9
sms	1798	1.0	0.0	1.0	1.0	690	2.3	2.0	1.0	11.0

participant's unique behavioral patterns were accurately captured. With this approach, we minimized false positives and ensured that anomalies reflect true deviations from each user's normal behavioral patterns. The Isolation Forest algorithm detected outliers across various variables in the dataset, with the number of outliers per variable ranging from 8 to 13. Variables such as 'screen', 'circumplex.valence', and several app categories like 'builtin', 'communication', and 'other' had around 12 outliers each. Fewer outliers were detected in 'sms' and 'appCat.entertainment', indicating less deviation from typical usage patterns in these categories. This pattern suggests variability in user behavior across different functions and mood states. Given the nature of the mood variable, which is subjectively reported by users, we decided not to alter or remove the detected outliers. Our rationale is based in the understanding that mood is inherently personal and can vary greatly based on individual experiences. Removing or adjusting these outliers could introduce bias into the dataset, as it could hide important emotional events that are crucial to understanding user behaviour. Furthermore, with only 11 outlier values identified, which is a relatively small number compared to the overall dataset, the potential impact on the overall analysis is minimal. This approach ensures that we preserve the integrity and authenticity of the user-reported mood data, as supported by findings in the literature that discuss the implications of outlier handling on statistical conclusions [3]. For the remaining variables, which primarily derived from sensor inputs such as screen time and various app categories, our approach to managing outliers was nuanced. We retained outliers for any variable if, for the same user and the same day, outliers were also observed in the mood data. This decision was based on the rationale that such simultaneous outliers may reflect important events or circumstances affecting the user's mood score. Outliers in 'calls' and 'sms' were also preserved, as inaccuracies in these data types are unlikely given they represent direct user interactions, which are less prone to sensor errors. For the remaining sensor-based variables in which there were no outliers of simultaneous disposition, outliers were removed to enhance the consistency and reliability of the dataset. This selective retention of outliers ensures that the dataset more accurately reflects typical and atypical user behavior, facilitating a more robust identification of genuine patterns and trends.

2. **Imputing missing values:** To tackle the missing values in our time-series dataset, we explored two sophisticated imputation methods that are well-suited to the intricacies of sequential data:
   (a) **Interpolated and Backfill:** The combination of Linear Interpolation and Backward Fill for imputing missing values in time series leverages the inherent temporal order to provide accurate data estimations. The Linear intterpolation first fills in gaps by creating a linear function that connects the two nearest known data points surrounding the missing data [6]. Following this approach, the Backward fill, completes any remaining gaps that interpolation couldn't fill, especially at the beginning of the dataset, by extending the next known value backward. This dual strategy ensures data continuity and minimizes the introduction of bias, making it acceptable for time dependence datasets.
   (b) **K-Nearest Neighbors (KNN):** K-Nearest Neighbors (KNN) is a method that estimates missing values in a dataset by finding the nearest neighbors to a data point with missing entries, based on a similarity measure [7].The missing values are then imputed using the mean or the median from the nearest neighbors. At first we separated the non numeric columns 'id' and 'date' from the numeric ones

because KNN requires only numerical inputs. The KNN imputer was applied with the mean of its five nearest neighbors. The imputed numerical data was then transformed back into a DataFrame. Lastly, we recombined the non numeric data with the newly imputed data ensuring the indices were aligned.

Choosing the right imputation method for time series depends on the nature of the data and the specific patterns and dependencies inherent in it. After testing both the K-Nearest Neighbors (KNN) and the combination of Linear Interpolation and Backward Fill for imputing missing values in our dataset, we chose the KNN approach based on two critical observations. The Backward Fill method introduced a bias by extending an outlier value of approximately 9.2 across all preceding missing entries for user AS14.03, which was particularly problematic for analyses sensitive to outliers. The second reason was that the KNN method produced a more balanced and smoothly distributed dataset by leveraging inherent relationships, thus avoiding the extension of outliers and maintaining natural variability.

1.3 TASK 1C: FEATURE ENGINEERING

Having completed the data cleaning phase, the next step was to engage in feature engineering. This process is crucial for enhancing the predictive power of our machine learning models by creating more informative and relevant features from the cleaned dataset. Our approach to feature engineering consists of six steps, each designed to enhance the dataset for improved model performance. These steps are outlined below:

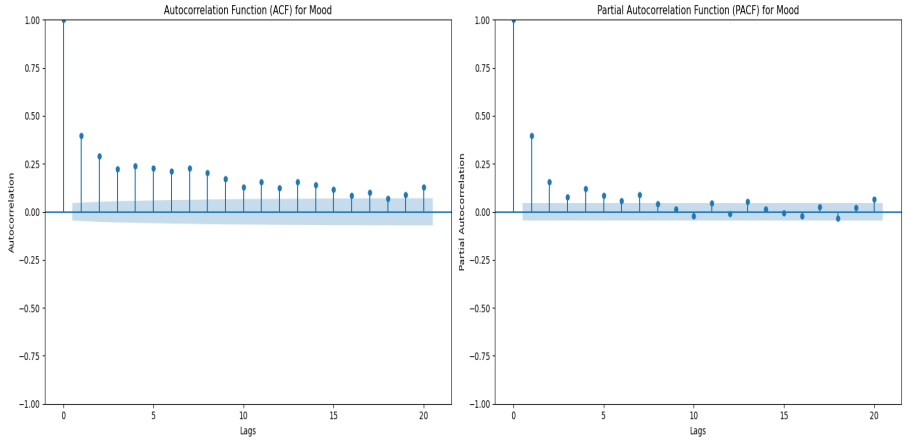

Fig. 4

1. **Creating the Target Attribute for Classification:** The first step in our feature engineering process was the creation of a target attribute for classification. We utilized the existing 'mood' feature to generate a new categorical variable named mood_category. This category was crated with three levels; Low, Medium, and High, each representing a range of daily mood scores. To ensure balance in the target variable, which is crucial for avoiding model bias towards any particular class [8], we set the thresholds for these categories such that each one contained approximately the same number of observations. This approach facilitates more accurate and equitable predictive performance across all mood states. Ultimately, the new feature consist of 446 Low values, 417 Medium values, and 409 High values.
2. **Transformation:** The second step in our feature engineering process involved transforming various dataset variables to optimize their distribution and scale, enhancing model performance. For features such as screen, and all app.Cat, which showed right-skewed distributions, we applied a logarithmic transformation after adding a small constant to manage zero values. In addition, we applied Min-Max Scaling to normalize their range, in order to manage skewness and stabilize the variance. For the more symmetrically distributed features such ascircumplex.arousal, circumplex.valence, and activity, we used Robust Scaling. This method is ideal for features that are already fairly well-distributed but still benefit from scaling to a common range. Furthermore, for count variables like call and sms, which have many small values and a few larger outliers, we applied Min-Max Scaling to bring them onto a 0 to 1 scale without transforming their nature as count data. These transformations are necessary in order to reduce potential model bias and improving the predictive accuracy, as outlined in Osborne's discussion on the effective use of data transformations to improve normality and manage various data distributions [9].
3. **Encoding Cyclical Features:** As our dataset is a time series, the third step in our feature engineering involved creating and encoding cycle features from the date variable to capture cyclical patterns. After

transforming the variables, we analyzed average mood variations by day, discovering that Fridays and weekends typically exhibit higher mood scores than other weekdays. This insight led us to introduce additional temporal features such as week, day, day_of_week, and week_of_year to explore broader temporal influences like seasonal changes and monthly variations. Additionally, we applied one-hot encoding to these features to treat each period distinctly, capturing unique patterns across different timescales without implying any ordinal relationships. Our goal with these cyclical and calendar-based features was to enable our models to better predict mood fluctuations by recognizing both weekly patterns and broader monthly and seasonal trends.

4. **Lagged Features:** The next step of the Feature Engineering process was to create lagged features. Lagged features involve shifting the values of a time series data backward in time to capture temporal dependencies within the dataset. By incorporating lagged features, we enable the predictive model to recognize and utilize the relationships between past and present data points. In order to create effective lagged features, we leveraged domain knowledge, as well as an understanding of the seasonality and trend of the data, to determine the appropriate lag intervals and variables to be used[10]. We developed lagged features for mood, arousal, and valence, as well as for app categories such as communication, entertainment, and social, all ranging from 1 to 7 days. Additionally, in creating these lagged features, we carefully ensured to prevent data leakage, securing that each feature used for training was derived strictly from past data, thus preserving the integrity of our predictive model. Furthermore, we conducted a decomposition analysis to confirm the cyclical nature of these variables, as illustrated in Fig.6. This helped us to justify our lag selection, capturing both short-term impacts and longer-term weekly trends important for understanding the dynamics of mood and app usage.
5. **Rolling Window** To further analyze the effects of mood over time, we implemented a rolling window. This approach involved creating features like mood_mean_1d - mood_mean_3d - mood_mean_7d, mood_median_1d -mood_median_3d - mood_median_7d, and mood_std_1d - mood_std_3d - mood_std_7d, based on the trends and seasonality observed in our data. These features calculate the mean, median, and standard deviation of mood over one up to seven days.
6. **Data Preparation for Train:** The final stage in Feature Engineering involved preparing the dataset for model training. This involved removing rows with empty values that resulted from generating lagged features and applying the Rolling window technique to the dataset. We decided not to impute these values, as it could introduce some bias in the data set. The next step after preparing the dataset was the feature selection. In this stage, we employed autocorrelation and partial autocorrelation techniques. These methods are useful for time series data, as they help to identify the relationship of a variable with itself over intervals of time. Autocorrelation measures how the values of the dataset are related to themselves over previous time periods, which assists in understanding the inherent serial correlation in time series data. Partial autocorrelation, on the other hand, offers insight into the correlation of the data with its own past values. This can indicate which lags are most important when forecasting future values of the series, thereby distinguishing the most impactful features from those that are collinear or irrelevant. This approach ensures that the model is trained on features that carry significant and direct implications for the outcome variable. Autocorrelation and Partial autocorrelation foe mood variable can be seen in Fig.4. As a final step in our feature selection process, we conducted a feature importance analysis (see Fig.9c) Then we utilized the dropdown method to reduce the dimensionality of the data and experimented through trial and error to identify the best combinations.

2 CLASSIFICATION

2.1 Random Forest:

The RandomForestClassifier is utilized for mood_category prediction based on features derived from mood data measurements. It constructs multiple decision trees during training and outputs the mode of the classes of these trees[11]. This method is well known for its robustness in handling large datasets with higher dimensionality. As a result, it provides insights into feature importance and maintains a balance between bias and variance.

1. **Training Procedure:** The RandomForestClassifier is trained using a TimeSeriesSplit cross-validation strategy to ensure robustness and generalize ability to unseen data. This approach involves splitting the dataset into sequential folds, where each fold retains the temporal order of the data. During each iteration, one fold is used for testing while the remaining folds are used for training. This process is repeated for all folds, allowing the model to be trained and evaluated across multiple time periods. During each iteration, the model is trained on a subset of the data using the selected features from the Feature Engineering process. The RandomForestClassifier constructs a set of decision trees with 100 estimators to learn underlying patterns in the training data, ensuring robustness with random_state=42. Once trained, the model is ready for prediction. TimeSeriesSplit cross-validation ensures preservation of temporal structure during training, enhancing its applicability to real-world scenarios.

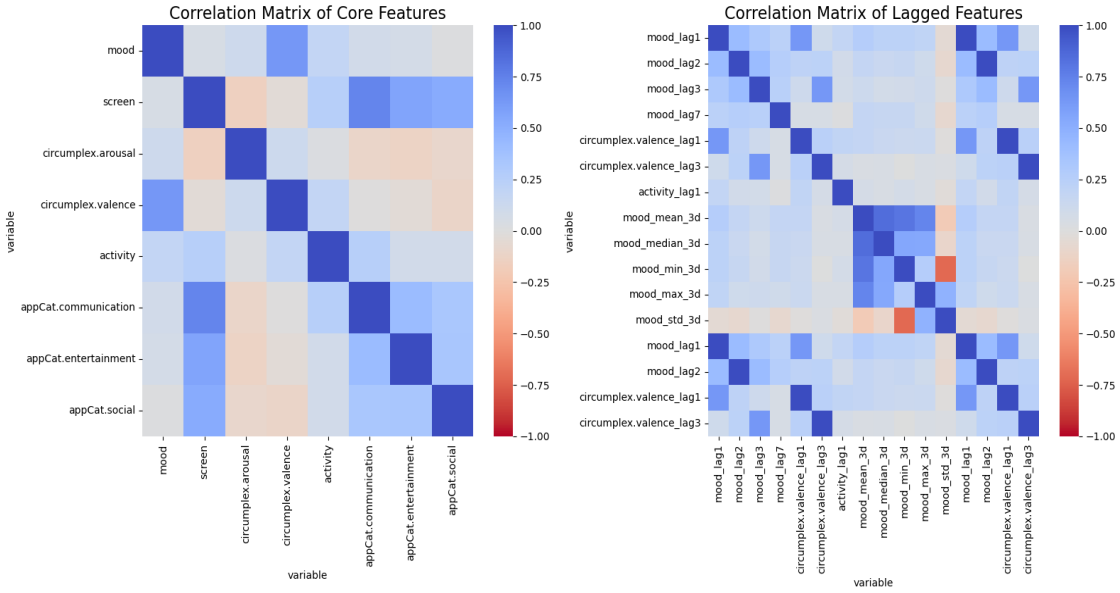

Fig. 5

2. **Hyperparameter Tuning:** For hyperparameter optimization, we employed RandomizedSearchCV, which explores large hyperparameter spaces using a probabilistic approach. Designed to run for 100 iterations, it sampled hyperparameter combinations from a specified distribution, which identified the optimal combination of hyperparameters. In table 3(a), the ranges and optimal parameters utilized in training the final model are displayed.

3. **Evaluation:** The first measurement of the performance, was focused on the accuracy metric, as it provides a clear assessment of the model's overall correctness in predicting mood categories. Accuracy is suitable in this context because it gives equal weight to true positives, true negatives, false positives, and false negatives. However, it's essential to acknowledge that accuracy alone cannot be sufficient, especially if certain types of errors are more critical than others. Therefore, as a second measure, we used precision, recall, and F1 score for a comprehensive evaluation of its predictive capabilities. Precision, indicates the ratio of true positive predictions to the total number of positive predictions, while recall calculates the ratio of true positive predictions to the total number of actual positive instances. Lastly, the F1 score is the harmonic mean of precision and recall, providing a balanced measure between the two metrics. For the visualization of the performance, we used the confusion matrix, a graphical representation, showing the model's classification performance by comparing true and predicted mood_categories. Additionally, we implemented an expanding window validation method, which adjusts the training size and evaluates the model's performance over time. As a result, we visualized how the model's accuracy evolves with varying amounts of training data. With this visual aid, we were able to examine the adaptability to changing conditions. This indicates the model's capability to detect seasonal patterns within the dataset as it undergoes training on various subsets of the data. Therefore, large plot fluctuations indicate the model's sensitivity to training data size changes, suggesting issues such as data sensitivity or instability.

4. **Description of results:** The Random Forest model demonstrated a noteworthy performance in classifying mood categories, with optimized hyperparameters resulting in high precision, recall, and F1 scores (all above 0.89 for the different mood categories). In addition, the average accuracy across all sequential folds in the time series data was approximately 81.11%. Furthermore, the model's robustness was evident from the confusion matrix, which showed a high number of true positives and minimal misclassifications. To examine the model's capability in the context of our time series dataset's properties we employed expanding window validation to chart performance trends. This method revealed the stable performance and adaptability of the model over successive time periods, as there was only a small variation. (approx. from 78 to 82). As a result, the evaluation showed that the model can manage the time-structured inherent in mood prediction from time series data. However, the significant drop in the model's performance at the index around 400 and Fold number equal to 3, suggests a difficulty in capturing the underlying patterns at that point in the time series. For this period, a closer examination might provide more insights, in order to address the cause of this anomaly in the data.

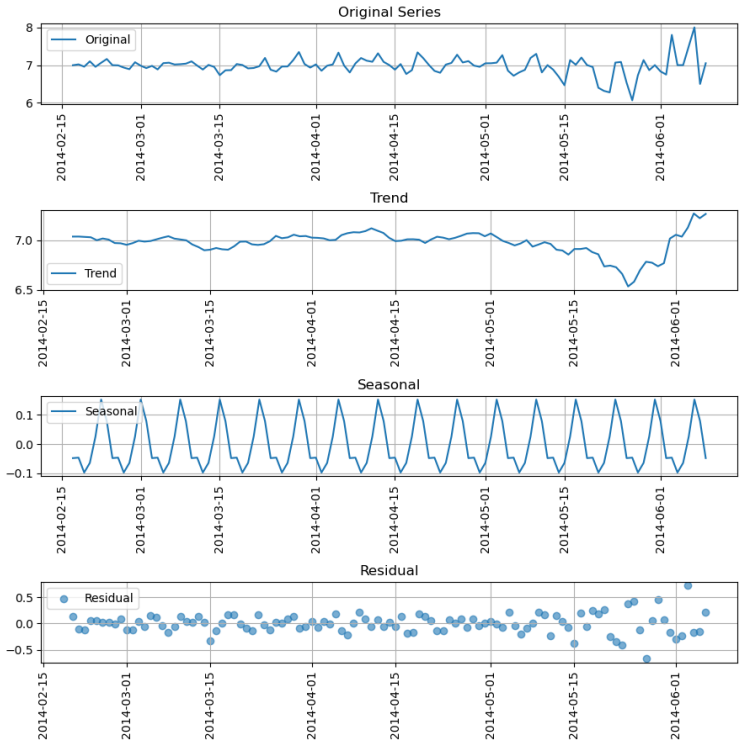

Fig. 6

Table 3: Hyperparameter Tuning Results and Classification Report (Random Forest)

(a) Hyperparameter Tuning Results

Variable	**Best Parameter**	**Parameter Range**
n_estimators	600	`np.arange(100, 1001, 100)`
max_features	'log2'	`['auto', 'sqrt', 'log2']`
max_depth	20	`np.arange(10, 101, 10)`
min_samples_split	10	`[2, 5, 10]`
min_samples_leaf	2	`[1, 2, 4]`
bootstrap	True	`[True, False]`

(b) Classification Report

Class	**Precision**	**Recall**	**F1-Score**
High	0.91	0.96	0.93
Low	0.94	0.93	0.94
Medium	0.92	0.89	0.90
Accuracy		0.93	
Macro Avg		0.93	
Weighted Avg		0.93	

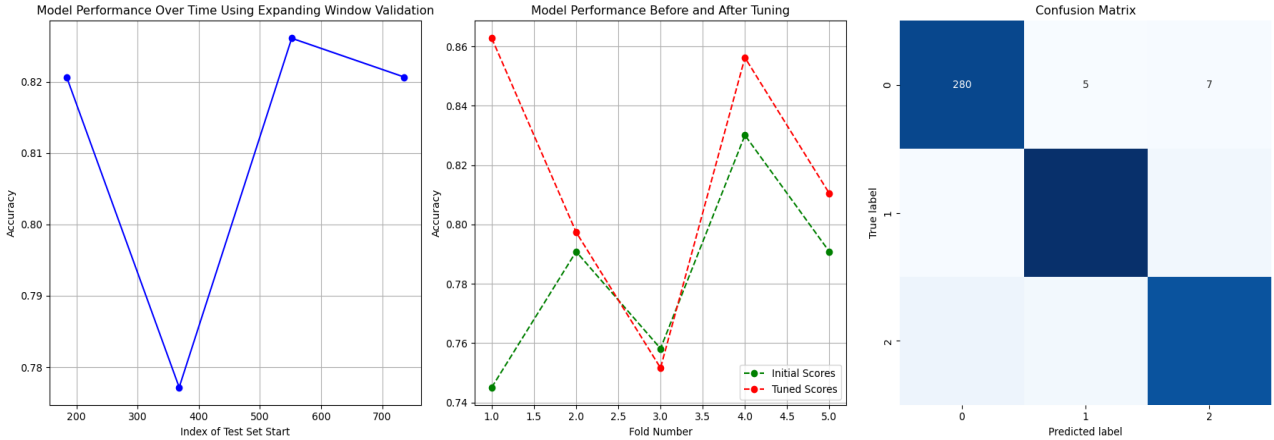

Fig. 7

2.2 LSTM:

The Long Short-Term Memory (LSTM) model is applied for mood_category prediction based on features derived from time-series data related to mood and behavior. LSTMs are particularly effective for sequence prediction problems due to their ability to maintain information in memory for long periods, which is essential in our case

for this time-series analysis[12]. This makes them suitable for capturing temporal dynamics and dependencies in the data, therefore providing insights into how past mood states influence future states.

1. **Training Procedure:** The LSTM model is trained using a TimeSeriesSplit cross-validation strategy to ensure it generalizes well to unseen data. This method is particularly apt for sequence data, as it preserves the chronological order. During training, the model is exposed to different subsets of data across multiple time windows, which simulates a real-world scenario where future predictions are based on past data. The model configuration includes multiple LSTM layers to learn various levels of abstraction in the data. Each LSTM layer processes the input sequence, with the final layer outputting a sequence that serves as the input to a dense layer for mood prediction. To prevent overfitting, we have added dropout layers are between LSTM layers.
2. **Hyperparameter Tuning:** For hyperparameter optimization, we utilized Keras Tuner's RandomSearch to explore the hyperparameter space. This way we test various combinations of model configurations over a specified number of trials, aiming to optimize the balance between training accuracy and generalization to unseen data. The optimal hyperparameters are presented in Table 4(b) . Optimal hyperparameters identified include two LSTM layers with 160 units each and 0 dropout rate, suggesting that minimizing dropout helped stabilize the learning process. The learning rate was finely tuned to approximately 0.001015, optimizing the speed and stability of convergence.
3. **Evaluation:** The primary evaluation metric used is accuracy, which measures the model's ability to correctly predict mood categories across all classes. Given the nature of the data, we also focus on precision, recall, and F1 score to evaluate the model's performance comprehensively. These metrics help assess the quality of predictions in terms of both the relevance and retrieval of true positive results. Precision measures the proportion of correct positive predictions made out of all positive predictions, which helps in understanding the model's exactness. Recall, on the other hand, focuses on the model's ability to identify all relevant instances, expressed as the ratio of correct positive predictions to the total actual positives. The F1 score, being the harmonic mean of precision and recall, serves as a balanced metric that considers both the precision and the recall. Visualizations play a crucial role in conveying the LSTM model's performance. We utilize the models accuracy and loss in the train and validation dataset along with a confusion matrix to graphically represent the classification outcomes, comparing actual to predicted categories. This visualization is instrumental in identifying the model's strengths and weaknesses in class-specific predictions.
4. **Description of results:** Based on the evaluation metrics and hyperparameter tuning, the LSTM model demonstrates varied performance across different categories, achieving a final test accuracy of approximately 71%. The precision metrics indicate that the model is most reliable when predicting the Low class, with a precision of 0.77, meaning that predictions for the Low category are correct about 77% of the time. However, the model exhibits less precision for the Medium class at 0.60, suggesting a higher incidence of false positives for this category. The High class shows a precision of 0.69. Recall scores shows the model's effectiveness at identifying instances across classes. The model is most effective at recognizing instances in the Medium class, with a recall of 0.78, indicating it captures a substantial proportion of actual Medium cases, albeit missing some. The recalls for the High and Low classes are 0.69 and 0.61 respectively. The F1-scores, which balance precision and recall, are fairly consistent across the classes, averaging around 0.68-0.69. This indicates a moderate balance between precision and recall but also highlights areas for improvement, particularly in enhancing both metrics to increase these scores. The overall accuracy of the model stands at 0.71, indicating that the model correctly predicts the category about 71% of the time. Both macro and weighted averages are similarly aligned at around 0.69 for both precision and recall, reflecting a consistent performance across classes without significant bias towards any single class. To conclude, the LSTM model shows competence in certain areas, especially in predicting the Low category, the performance across categories is uneven, and there is significant room for improvement. This underscores the importance of continued model tuning and possibly exploring additional features or data to enhance model reliability and accuracy. The consistency of macro and weighted averages also suggests that the model treats all classes with comparable fairness, but enhancing overall precision and recall could lead to better performance metrics.

2.3 TASK 2B: WINNING CLASSIFICATION ALGORITHMS

The competition was "Toxic Comment Classification Challenge" which was hosted by Jigsaw and Google as part of the Conversation AI team initiative. The competition started on December 19, 2017, and ended on March 21, 2018. The dataset consisted of comments from Wikipedia's talk page edits. It required participants to build a model capable of identifying multiple types of toxicity, such as toxic, severe toxic, obscene, threat, insult, and identity hate, in online comments.The models were evaluated using the mean column-wise ROC AUC, focusing on predicting probabilities for each type of toxicity per comment. The winners of the competition used the pre-trained word embeddings and they also implemented robust machine learning techniques. Their classification system was an ensemble of bidirectional GRU (BiGRU) layers followed by dense layers designed to output

probabilities across multiple categories of toxic behavior. The winning approach had a strategy that emphasized the embedding layer's importance. By using diverse pre-trained embeddings from sources like FastText and Glove, they provided their model with a huge foundation of language understanding. This was very crucial for the classification task at hand, where comments needed to be accurately categorized into different forms of toxicity. Classification was achieved through their machine learning pipeline that applied BiGRU layers, a type of RNN favored for its ability to capture context from both directions in text sequences. The BiGRU layers were helping to understand the sequence of words and their impact on how toxic they are. The output from the RNNs was then passed through dense layers, which was the final classification step, providing a probability for each type of toxicity. What set the winning approach apart was their application of train/test-time augmentation (TTA) using translations, which boosted their models' ability to understand and predict toxicity in comments. They translated the English comments into French, German, and Spanish and then back to English to augment the dataset and improve the model's generalization capabilities. Furthermore, their use of rough-bore pseudo-labeling (PL), where they added their ensemble's labeled test samples to the training set and retrained the model, in which they said that it helped address any distribution differences between the training and test datasets. Their validation and stacking framework also distinguished their method from others. They used a rigorous cross-validation strategy and stacked their models using LightGBM, fine-tuned through extensive Bayesian optimization. It improved the performance but also it insured against overfitting, something that is very common in such machine learning competitions. In contrast to other more complicated or architecture-focused approaches, the winner's solution shows that basic machine learning techniques, when carefully validated together with strong, pre-trained embeddings, can give better performance even in a very hard NLP task.For more information regarding the competition you can visit the Toxic Comment Classification Challenge — Kaggle.

Table 4: Classification Report and Hyperparameter Tuning Results (LSTM)

(a) Classification Report

Class	**Precision**	**Recall**	**F1-Score**
High	0.69	0.69	0.69
Low	0.77	0.61	0.68
Medium	0.60	0.78	0.68
Accuracy		0.71	
Macro Avg		0.69	
Weighted Avg		0.69	

(b) Hyperparameter Tuning Results

Variable	**Best Parameter**
Number of LSTM Layers	2
Units in LSTM Layers	160
Dropout Rate	0.0
Learning Rate of the Optimizer	0.0010153958908885

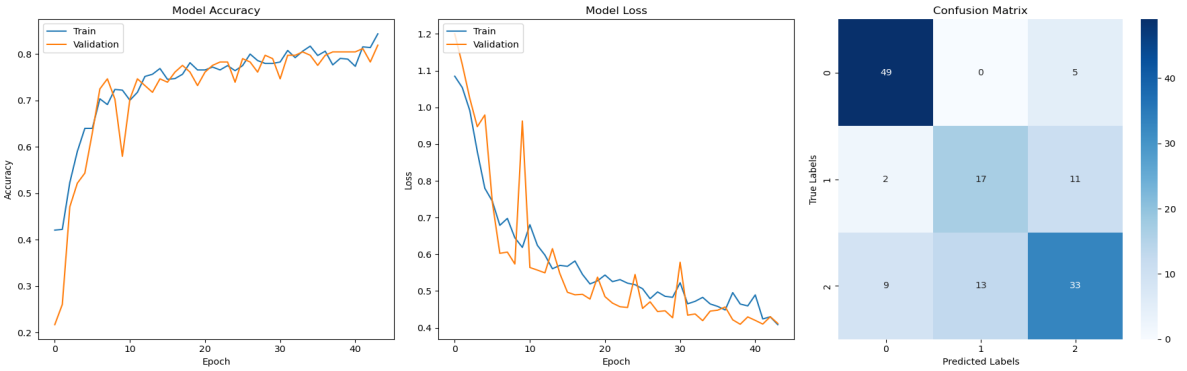

Fig. 8

3 ASSOCIATION RULES

Association rules are a highly effective data mining technique used to identify connections and patterns among items in large datasets, often applied in market basket analysis. The Apriori algorithm is notable for generating association rules that are crucial in shaping decision-making and marketing strategies. Innovations to enhance the Apriori and similar methods often involve grouping products into higher-level categories. For instance, different types of pizzas might be categorized under a general "pizza" category. This method can use clustering techniques, such as Agglomerative Hierarchical Clustering, to group similar items, thereby simplifying complex data sets before applying the Apriori algorithm [13].

Advantages of this approach include:

1. **Reduced Complexity:** By grouping items like various types of pizzas into a single "pizza" category, the algorithm processes fewer, broader item categories. This simplification can lead to faster processing times and reduced computational load.
2. **Increased Relevance of Rules:** This grouping allows for the generation of more universally applicable rules. Instead of generating specific rules for each type of pizza, a general rule applicable to all pizzas can enhance utility across different types.
3. **Improved Manageability:** It becomes easier to manage and update rules when they are generated for broader categories rather than specific items, making the algorithm more scalable and adaptable.

Disadvantages of this approach include:

1. **Loss of Detail:** While grouping products into broader categories simplifies data analysis, it may result in the loss of specific insights that could be obtained from individual item analysis. For example, specific buying patterns for Pizza Margherita versus Pizza Quattro Formaggio might be overlooked.
2. **Potential for Overgeneralization:** Applying the same rules to all items in a broad category might not always be appropriate as different items within the category could have different consumer purchasing behaviors.
3. **Challenges in Category Definition:** Deciding how to group products can be subjective and challenging. Incorrect categorization can lead to less effective association rules, which might misrepresent actual purchasing trends.

In relation to our dataset, which features complex and multidimensional data including variables like mood, activity, screen time, and various app usage categories, Agglomerative Hierarchical Clustering can be beneficial. This method groups similar days or user behaviors, reducing complexity before applying association rule mining. Although Agglomerative Hierarchical Clustering isn't inherently designed for time-series data, it can be adapted to consider sequential days in the clustering criteria, aiding in the identification of consistent mood patterns or similar app usage behaviors—key for understanding mental health dynamics [14]. The customizable nature of this clustering technique allows for varying levels of detail in data analysis, which is particularly valuable when dealing with diverse behavioral data.

Practical considerations include the resource-intensive nature of Agglomerative Hierarchical Clustering, particularly with large datasets. However, the structure of our dataset, primarily comprising daily measurements, may make this approach feasible. Consideration of the available computational resources and potential optimization strategies is crucial. The clusters formed can be visually represented through dendrograms, enhancing the interpretability of the data and offering valuable insights into the organization of different behavioral patterns before applying the Apriori algorithm.

4 NUMERICAL PREDICTION

Random Forest

1. **Model Setup and Training:** For our regression model, we dropped the 'mood_category' column and instead used the 'mood' numeric column as our target variable. This adaptation focuses on predicting actual mood scores, using the RandomForestRegressor for its robust handling of complex, feature-rich time series data. The Random Forest Regressor predicts the continuous numerical value of mood by averaging the outputs of individual trees in the forest. Unlike a classifier, which outputs class labels, the regressor focuses on estimating a precise value to minimize prediction error. The training was conducted on 80% of the data, reserved chronologically to preserve the temporal order critical for time series analysis.
2. **Hyperparameter Tuning:** The methodology used for hyperparameter tuning in this regression task, mirrored the one from the classification task. In these scenarios, RandomizedSearchCV was employed with a 5-fold cross-validation setup, performing 50 model fits to optimize the hyperparameters effectively. Details on the parameter ranges, initial settings, and optimized values are provided in Table 5.

LSTM

1. **Model Setup and Training:** We utilize a Long Short-Term Memory (LSTM) network to predict continuous mood scores, focusing on the numeric column 'mood' as our target variable. The LSTM model was trained using 80% of the data, with the sequence reserved chronologically to preserve temporal dependencies critical for accurate time series forecasting.
2. **Hyperparameter Tuning:** We employed Keras Tuner's RandomSearch to systematically explore the model configuration space. This method tested various combinations of LSTM layers, units, and dropout rates over a specified number of trials, aiming to find the optimal balance that minimizes validation loss while preventing overfitting. The search was conducted across 10 trials, with each trial running for up to 50 epochs, monitored by early stopping to interrupt training if the validation loss ceased to improve. The best hyperparameters can be found in Table 5.

Table 5: Hyperparameter Tuning Ranges, Initial, and Optimal Values for Different Models

Parameter	**Range**	**Initial/Best**	**Parameter**	**Range**	**Initial/Best**
No. of Estimators	100 to 1000 (100 steps)	100 / 700	No. of LSTM Layers	1 to 3	1 / 2
Min Samples Split	2, 5, 10	2 / 5	Units per Layer	32 to 128 (32 steps)	32 / 160
Min Samples Leaf	1, 2, 4	1 / 1	Dropout Rate	0.1 to 0.5 (0.05 steps)	0.25 / 0.25
Max Depth	10 to 110 (10 steps), None	None / 60	Learning Rate	0.001 to 0.01 (log scale)	0.001 / 0.0010153
Bootstrap	True, False	True / True			

Comparison Classification - Regression: Regression and classification tasks serve different purposes: regression for predicting exact, continuous outcomes and classification to determine the category to which an input belongs. The differences also extend to how models are trained, tuned, evaluated, and the interpretation of their performance. In regression tasks, the model is designed to output continuous values, aiming for as precise numerical predictions as possible. In contrast, classification tasks are concerned with assigning discrete labels to data points, effectively categorizing the inputs into distinct classes or groups. When dealing with regression, the target variable is typically a single continuous quantity—in this case, the 'mood' column—which the model attempts to predict. The model must capture the nuances within the data, as even minor changes can greatly impact the continuous outcome. On the other hand, for classification, the target involves categorical outcomes. Here, the focus is to distinguish between different mood states, ensuring that each instance is correctly categorized. The training of regression models emphasizes the reduction of prediction errors, quantified by loss functions like the Mean Squared Error (MSE). The smaller the difference between predicted and actual values, the better the model performs. In classification, the training process aims to enhance the accuracy of categorization, with performance metrics such as accuracy, precision, recall, and F1 score measuring the model's success. These metrics offer a comprehensive view of the model's performance by accounting for the overall correctness and the quality of positive predictions. Evaluating a regression model involves measuring how closely the predicted values align with actual values across a continuous spectrum. Conversely, classification models are evaluated on their ability to distinguish between different categories effectively. Tools such as confusion matrices are mandatory in visualizing and understanding class-specific performance. The sensitivity to data nuances also varies. Regression models are particularly sensitive to outliers since the aim is to predict exact values, necessitating careful data normalization. Classification models may be more robust to outliers since the prediction is categorical. Visualization techniques are tailored to the type of task. For regression, plots comparing actual versus predicted values are crucial in assessing the model's accuracy. For classification, plots of accuracy and loss over training epochs, as well as confusion matrices, provide clarity on the model's success in correctly labeling classes and identifying any patterns of misclassification.

5 EVALUATION

TASK 5A: Characteristics of Evaluation Metrics: MSE and MAE

Predictive modeling requires robust evaluation metrics to accurately assess model performance. Among the most commonly used metrics are the Mean Squared Error (MSE) and Mean Absolute Error (MAE), each providing unique insights into prediction accuracy. This subsection explores their formulas, implications, and practical use cases, informed by recent studies MAE and RMSE — Which Metric is Better?[15].

Mathematical Formulation

Mean Squared Error (MSE): MSE measures the average of the squares of the errors, which quantifies the variance between the predicted and actual values. It is defined as follows:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{1}$$

where y_i represents the actual values, $\hat{y}_i$ represents the predicted values, and n is the number of observations.

Mean Absolute Error (MAE): MAE calculates the mean of the absolute differences between predicted and actual values, providing a linear score that reflects the true average model prediction error. The formula for MAE is:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2}$$

Comparative Analysis

1. **Sensitivity to Outliers** MSE is particularly sensitive to outliers as the squaring of error terms can significantly affect the metric when large deviations exist. This property makes it suitable for applications where large errors are especially undesirable. Conversely, MAE is more robust against outliers, as it treats all deviations linearly, making it ideal for datasets where outliers are expected but should not dominate the error metric MAE and RMSE — Which Metric is Better?[15].
2. **Differentiability and Optimization** The differentiability of MSE with respect to the predictions makes it better option for gradient-based optimization methods, facilitating efficient parameter updates. MSE's property of having a unique global minimum assists in optimization by reducing the likelihood of encountering local minima. MAE's non-differentiability at zero can cause challenges, especially with gradient descent techniques, as it may require additional computational heuristics to handle [15].
3. **Practical Applications** Choosing between MSE and MAE depends on the specific requirements of the modeling task and data characteristics. MSE is often preferred in financial and scientific computing where accurate and consistent predictions are critical, and large errors can have substantial consequences. MAE is typically selected in marketing and system monitoring, where the focus is on maintaining fairness across predictions and minimizing the impact of anomalies MAE and RMSE — Which Metric is Better?.
4. **Example: Identical Results in Specific Scenarios** In a hypothetical dataset where all observations lie exactly on a predicted linear regression line, both MSE and MAE would yield identical results, as each prediction perfectly matches the actual data. This example illustrates that under certain conditions, such as perfectly linear data, the choice between MSE and MAE might not impact the model's evaluation outcome.

Note: In practice, such conditions are rare, and the choice of metric should consider the broader implications of each metric's characteristics.

TASK 5B: IMPACT OF EVALUATION METRICS

Random Forest

1. **Robust Error Metrics** With a MAE of 0.23192 and an MSE of 0.09880, the Random Forest model confirms its ability to make close predictions with a minimal rate of error. These metrics suggest that for most predictions, the model is likely to be within a quarter of a point of the actual value, on average, which is particularly strong performance for many applications.
2. **Effective Feature Utilization** Its ability to handle a large set of features effectively is evident in the low error rates. The model identifies which features are most informative, likely contributing to a more accurate and robust model by avoiding overreliance on noisy or irrelevant data.
3. **Adaptability to Complex Patterns** The model's success in this area suggests that it can uncover and utilize complex patterns and interactions within the dataset that are not easily detected, a strength that is often crucial for datasets with complex underlying structures.
4. **Generalization Strength** By building multiple decision trees and averaging their predictions, Random Forest is less prone to overfitting the unusual characteristics of the training data, demonstrating a strong ability to generalize effectively to new, unseen data.

LSTM

1. **Sensitivity to Data Characteristics** The LSTM's MAE 0.22412 and MSE 0.29205 indicate a performance that is sensitive to the dataset's characteristics, particularly the sequence and timing of events. The model's higher error rates may derive from the data not having strong or consistent temporal patterns for the LSTM to capture effectively.
2. **Dependency on Model Configuration** LSTMs require careful tuning, including the number of layers, the number of neurons per layer, and the dropout rate, among others. The performance of the model suggests that it may benefit from further tuning to better capture the dataset's temporal structure.
3. **Risk of Overfitting** Given the complexity of the LSTM architecture, there is a risk that the model could become too specialized to the training data, capturing noise rather than the underlying pattern, leading to poor generalization.

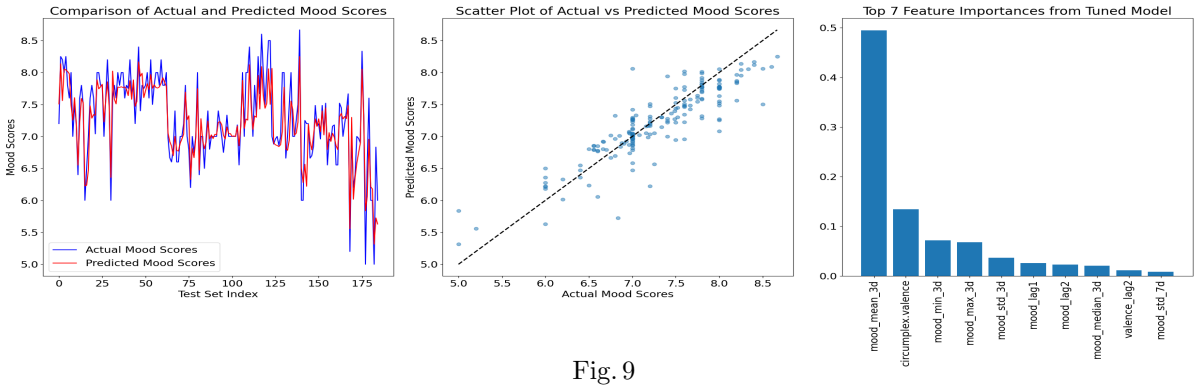

Fig. 9

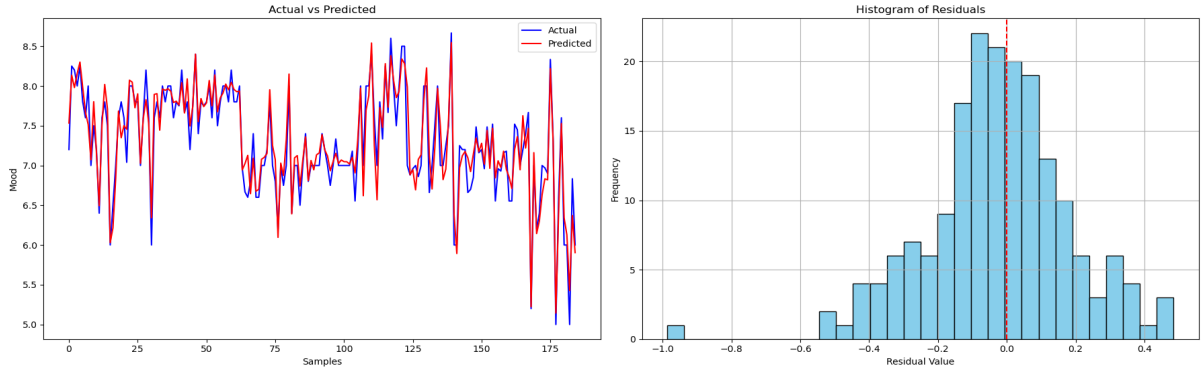

Fig. 10

Visual Analysis Insights

1. **LSTM Residual Patterns** The analysis of residuals from the LSTM model reveals a slight skew in predictions, indicating a systematic underestimation in certain cases. While the centering of residuals around zero suggests no bias in the predictions overall, the presence of outliers points to specific instances where the model's predictions diverge significantly from the actual values.
2. **Random Forest Consistency** The line and scatter plots show that the Random Forest predictions are generally consistent and closely aligned with the actual values, a sign of the model's reliable performance. Even though some prediction errors exist, particularly with extreme values, the overall trend shows a strong correlation between the predicted and actual values.
3. **Feature Importance Analysis** The bar plot of feature importances highlights the most influential variables according to the Random Forest model. This suggests a certain level of dependency on a few key features, which, while effective for prediction, could be an area to investigate to avoid potential overfitting to specific aspects of the training data.

Implications

1. **Model Selection** When deciding between Random Forest and LSTM models, the dataset's features and the nature of the prediction task must be carefully considered. The Random Forest model's lower error rates and strong performance across diverse features makes it suitable for datasets with complex, non-linear relationships, whereas the LSTM model may be the tool of choice for time-series data where the temporal sequence is predictive of the outcome.
2. **Practical Application** For practical applications, such as in health-related mood predictions or stock market forecasting, the choice of the model can significantly impact the effectiveness and efficiency of the predictions. Random Forest offers a balance between accuracy and computational efficiency, making it a practical choice for many applications, while LSTM models are better for tasks where the sequence of data points is crucial.
3. **Iterative Improvement** Both models can benefit from an iterative improvement process. For Random Forest, this might involve exploring ways to diversify the reliance on features to enhance model robustness. For the LSTM model, improvements might focus on better capturing temporal relationships and addressing outliers to improve prediction accuracy.

Bibliography

[1] J Asselbergs, J Ruwaard, M Ejdys, N Schrader, M Sijbrandij, and H Riper. Mobile phone-based unobtrusive ecological momentary assessment of day-to-day mood: An explorative study. *Journal of Medical Internet Research*, 18(3):e72, 2016.

[2] Khadija Zanna, Sayde King, Tempestt Neal, and Shaun Canavan. Studying the impact of mood on identifying smartphone users. *Preprint*, June 2019. Available online at ResearchGate.

[3] Thomas V. Pollet and Leander van der Meij. To remove or not to remove: the impact of outlier handling on significance testing in testosterone data. *Adaptive Human Behavior and Physiology*, 3(1):43–60, 3 2017.

[4] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008.

[5] Mohammad Braei and Sebastian Wagner. Anomaly detection in univariate time-series: A survey on the state-of-the-art. *Preprint posted to ResearchGate*, April 2020. Technische Universität Darmstadt, Telecooperation Group.

[6] Mathieu Lepot, Jean-Baptiste Aubin, and Franz H.L.R. Clemens. Interpolation in time series: An introductive overview of existing methods, their performance criteria and uncertainty assessment. *Water*, 9:796, 2017.

[7] Parisa Saeipourdizaj, Parvin Sarbakhsh, and Akbar and Gholampour. Application of imputation methods for missing values of pm10 and o3 data: Interpolation, moving average and k-nearest neighbor methods. *Environmental Health Engineering and Management Journal*, 8(3), 2021.

[8] Priyanka Banerjee, Frederic O. Dehnbostel, and Robert Preissner. Prediction is a balancing act: Importance of sampling methods to balance sensitivity and specificity of predictive models based on imbalanced chemical data sets. *Frontiers in Chemistry*, 6:362, 2018.

[9] Jason Osborne. Notes on the use of data transformations. *Practical Assessment, Research and Evaluation*, 9:42–50, 01 2005.

[10] John Culnan, Damian Y. Romero Diaz, and Steven Bethard. Exploring transformers and time lag features for predicting changes in mood over time. In *Proceedings of the Eighth Workshop on Computational Linguistics and Clinical Psychology*, pages 226–231, Tucson, AZ, USA, 2022. Association for Computational Linguistics.

[11] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 10 2001.

[12] Konrad Zołna and Bartłomiej Romański. User modeling using lstm networks. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 5025–5026, 2020.

[13] Athman Bouguettaya, Qi Yu, Xumin Liu, Xiangmin Zhou, and Andy Song. Efficient agglomerative hierarchical clustering. *Expert Systems with Applications*, 42(5):2785–2797, 2015.

[14] William H. E. Day and Herbert Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification*, 1(1):7–24, 12 1984.

[15] T. Chai and R. R. Draxler. Root mean square error (rmse) or mean absolute error (mae)? – arguments against avoiding rmse in the literature. *Geoscientific Model Development*, 7(3):1247–1250, 2014.