Assignment 1 Data Mining Techniques Data Mining Practice and Theory

Maya Ozbayoglu[1*][2794753], Dingming Liu[1*][2815739], and Junming Ye[1*][2803441]

Vrije Universiteit Amsterdam

Understanding the different factors that influence the users' reported mood based on historical data as well as available sensory data about the behavior of the user and their reported moods can help improve different kinds of resources for people suffering from various mental health illnesses, such as depression. This paper shows the various steps that were taken in order to predict users' mood for the next day. In our feature engineering approach, we eliminate the attributes that do not seem to influence the user's mood as well as add relevant features built on historical data using the window-sliding technique. Then, we design four models – two based on classification and two on numerical prediction. For each model, we used different hyperparameter optimization methods: Random Search for LSTM and Grid Search for Random Forest. Our models for classification consisted of a Random Forest Classifier and for numerical prediction: an LSTM neural network. Our results show that the most important feature in predicting the user's mood for the target day is their reported mood from the day before. Hence, our approach of adding time-specific features into our model proved to be helpful.

1 Task 1: Data Preparation

1.1 Task 1A: Exploratory Data Analysis

Our group decided to complete the advanced task for which the dataset was provided. It originates from the domain of mental health and includes records of different kinds of sensory data about the behavior of the user - such as total arousal and valence scored by the user, call made, sms sent, screen time as well as time spent on individual applications, for ex. those related to communication or entertainment. Our dataset contains 376 912 rows (records) and 5 columns with the following features: id of the user, time in which the given variable was recorded, the variable and its corresponding value. The most important attribute of our dataset is the reported mood by the user on a scale of 1-10 in a given time as the goal of this assignment was to predict the average mood of the user for the next day. The ranges for each feature were given in the assignment description, therefore we will not mention them in this report.

The most data are available for the screen time – there are 96 578 records of different users' screen time, following records of time spent on builtin apps with 91 288 records and in the third place being data on time spent using communication apps equal to 74 276 records. The least amount of data is available for the following variables: time spent on weather apps - 255 records, game apps –

813 and lastly, finance and unknown apps equally – 939. Data availability for the reported mood lies somewhere in between being equal to 5641 records. What is interesting is that different users have different times of records, for ex. there is more data availability for user AS14.01 than for user AS14.02 suggesting that separating the data to make it user-specific by, for ex. creating a dictionary of dataframes with individual dataframes containing information about individual users. This way the mood prediction will also be more accurate as it will be based on data that concerns the given user.

In Figure 1 we can see that mood variability changes from user to user. Based on the plot it can be inferred that some users have a relatively stable mood in times of records – mostly oscillating around 7, whereas some demonstrate a high variance. Most of the users, however, tend to report a positive mood (above 6) for most of the time.

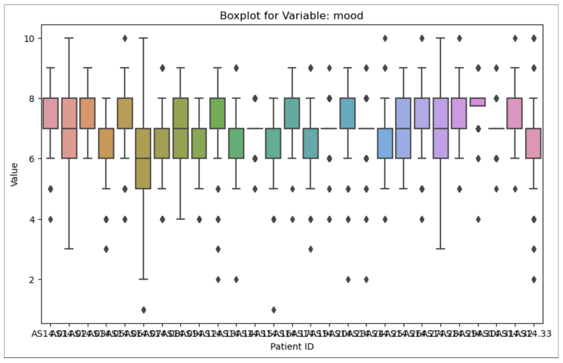


Fig. 1: Boxplot displaying the variance of the mood for each user

The barplots gave us an understanding of the mood variability of different users. However, it can also be interesting to look at mood variability over time. Figure 2 shows exactly that for the user AS14.15. The plot clearly demonstrates that the analyzed user had multiple stable (reported) mood periods of 7 and sudden peaks reaching 8 and valleys falling to 5. What is positive that there seem to be more instances where the user reports higher moods than lower ones. The graph does not display a clear pattern, thus making it hard to conclude any seasonality of mood reportings by the analyzed user.

Lastly, what is essential in understanding our data better is to analyze how different attributes relate to each other. Figure 3, which is a heatmap of the correlation matrix of the reshaped data where columns standing for the different variables were added, demonstrates these relationships. Based on the heatmap, the highest correlation is between the screen variable and appCat.communication variable (84%). Hence, it can be inferred that users overall spend the most time on communication platforms. On an intuitive level this makes sense as the background of the users can be different – one can be more into gaming, one more

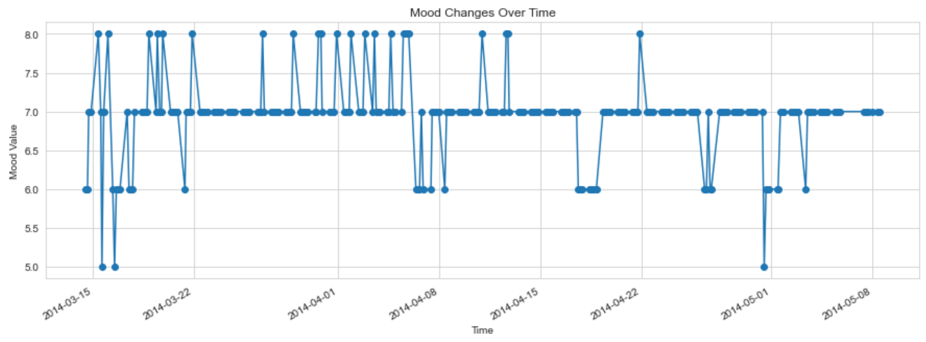


Fig. 2: Mood Changes Over Time

into finance – however, what most of us use on a daily basis are communication apps. Another correlation (of 50%) includes that of between appCat.builtin and screen time indicating that the users overall spend a big portion of their time on builtin apps as well. Another high correlation of 45% is between *circumplex.valence* and *mood*. For the rest of the features, there doesn't seem to be any significant correlations worth reporting. This in turn, means that most of the variables are independent of each other and do not directly translate into affecting another variable's score.

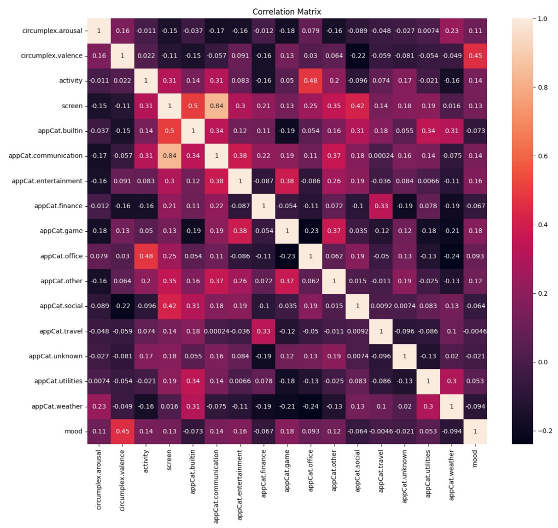


Fig. 3: Heat map showing the correlation matrix of all the variables

1.2 Data Cleaning

It can be observed that the dataset contains quite some noise, there is extreme or incorrect data, and some records are missing. Different ways were used to process or fill the data.

Imputation of NA The data analysis tool reveals that NA occurs only in variable *circumplex.arousal* and *circumplex.valence*, both of which range from -2 to 2. For the NA value, we apply the following formula to impute it:

$$v_T(t) = 0.6 * Mean(v_T) + 0.4 * Mean(Sum(v_{T\pm k}(t))), k = 1, 2 \tag{1}$$

while $v_T(t)$ represents missing values in period t on the T^{th} day, v_T represents the sum of variables on the T^{th} day.

This imputation method takes into account changes in sentiment on the day as well as utilizing trends in variables over time, allowing the estimate to more closely match the data distribution.

Clean of Outliers In the extreme value cleaning step, we first decided on two common ways of handling extreme values: the z-Score and the Interquartile Range method. Due to the non-normal distribution of all original data, as shown in the Figure 4, we ultimately decided to employ IQR method to remove outliers that are lower than lower_bound or higher than upper_bound, as depicted in the formula:

$$\begin{aligned} &IQR = Q3 - Q1 \\ &lower\_bound = Q1 - 1.5 * IQR \\ &upper\_bound = Q3 + 1.5 * IQR \end{aligned} \tag{2}$$

while $Q1$ is first quartile representing 25^{th} percentile of the data, and $Q3$ is third quartile representing the 75^{th} percentile of the data.

Collation of Data To facilitate the processing of the data and to better serve as an input to the model, our goal was to predict the user's Mood value for the coming day, so we decided to organize the data in the form of day-by-day records. It was observed that large swaths of records were missing at the beginning of the record for each user and consecutively long periods of missing data are not conducive to padding. Over-reliance on padding in the quest for data completeness makes it easy to destroy the original data distribution, therefore we decided to remove the records with more than 13 missing data and checked the legitimacy of the time series.

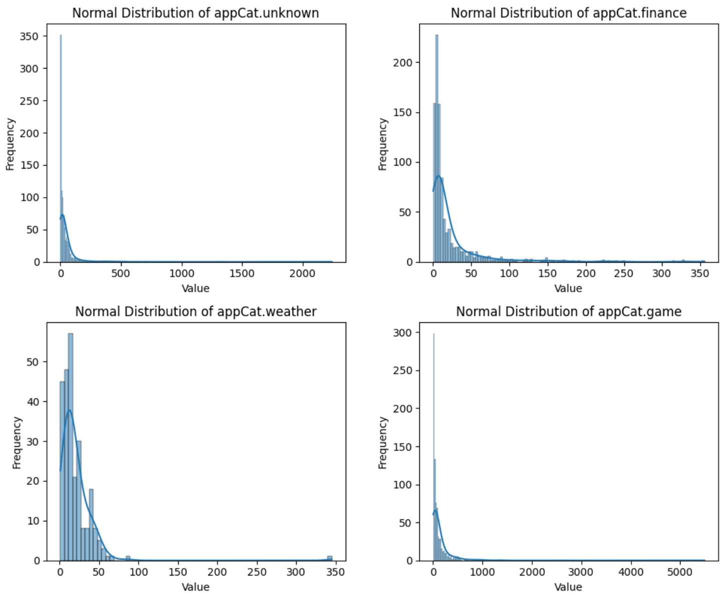


Fig. 4: Normal Distribution of Some Varibales

Imputation of Missing Value We compared three types of filling in the Fill Missing Values step, which are Mean imputation, k-Nearest Neighbors, and RandomForestRegressor.
Mean Imputation. All missing values are replaced with the mean of the respective feature, to supplement missing attribute values with the most probable value.
k-Nearest Neighbors. By utilizing either Euclidean distance or correlation analysis, the nearest K samples with missing data to a given sample are determined. Subsequently, a weighted average of these K values is computed to estimate the missing data for the respective sample.
RandomForestRegressor For a dataset with n features, if feature T contains missing values, we treat feature T as the target label. The remaining $n-1$ features, along with the original label, constitute a new feature matrix. We then utilize this new feature matrix to predict the missing values.

For the imputed dataset, we utilize heatmaps to observe whether the feature distribution of the imputed dataset matches that of the original data. As shown in the Figure 5, the feature distribution of the dataset imputed using RandomForestRegressor closely resembles that of the original data.

1.3 Task 1C: Feature Engineering

For our feature engineering component, we decided to follow two approaches. Firstly, in order to understand the results better as well as improve the model, we

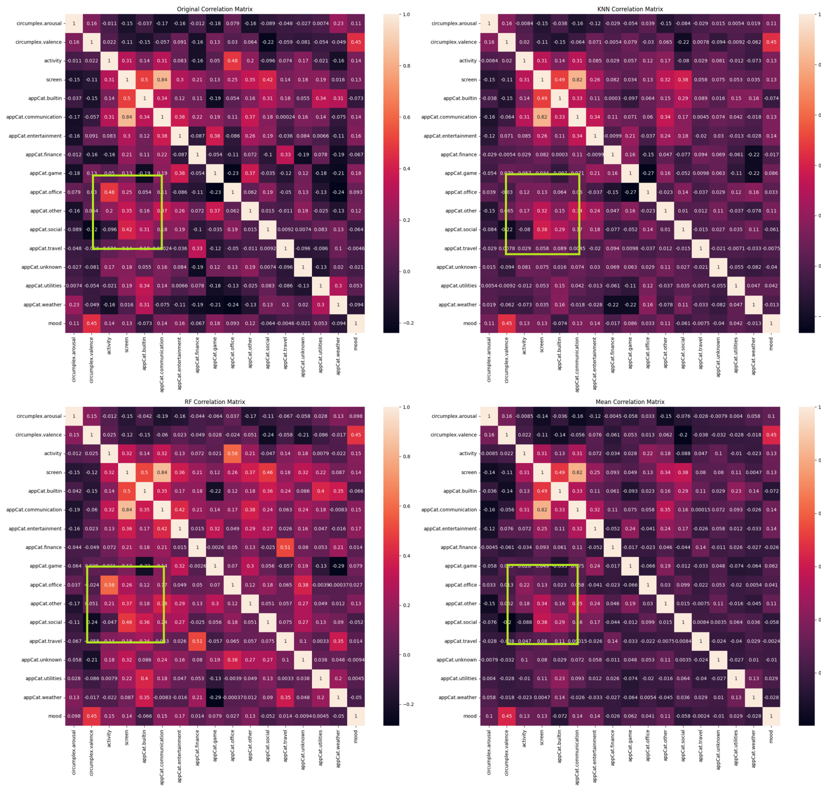


Fig. 5: Heatmaps of 4 Imputation

decided to drop the following features: 'screen', 'sms', 'call', 'appCat.unknown', 'appCat.office' and 'appCat.finance'. These choices were based on the heatmap as well as feature importance when predicting the mood from random forest (shown in Table 1) based on the insignificance of the features in relation to the mood variable. It is worth pointing out that we de decided to drop screen time as it constituted the cumulative time spent on all apps and for our more detailed analysis it is more important to understand which apps directly relate to changes in user's mood while also eliminating the ones that do not seem to have a significant impact. Not all variables serve the purpose of equally contributing to predicting our target variable, therefore not all variables will be a part of the features for our model.

Our second approach included adding additional features specifically related to a time-series data that we are dealing with. As was suggested in the description of the assignment, we decided to use a time frame of 5 days in order to predict the reported mood by the user on the 6th day. This was done using the window sliding technique where values from other columns corresponding to the next days were taken. Moreover, for an easier understanding as well as for

Table 1: Table of Top 5 Features' Importance

Feature	**Importance**
circumplex.valence(t)	0.2
mood(t-1)	0.04
mood(t-5)	0.04
mood(t-4)	0.04
mood(t-2)	0.04

The most important two factors in predicting the mood for the next day turned out to be the valence of the current day and the mood of the user in the previous day.

visualization purposes, the 'time' column was changed to "initial mood date" which corresponds to the average mood reported by a given user on that day and another column of "target mood date" was added that corresponds to the date of the mood on the target, 6th day. Thanks to this, later on when analyzing the feature importance of our models, we will be able to assess which factors played a more important role in determining the mood of the user on the target day, i.e. are the reported moods from the previous days better predictors of the mood on the target day than the other seemingly important attributes, such as circumplex.valence score or time spent on communication apps. This approach is a widely used one in research concerning time-series data.

2 Task 2: Classification

2.1 Task 2A: Application of Classification Algorithms

Classification Approaches In our project, we employed two distinct classification approaches to tackle the prediction of mood categories derived from the dataset: Random Forest and LSTM (Long Short-Term Memory) networks.

Data Preparation Before training the model, I analyzed the distribution of the data to ensure a more balanced dataset for training purposes. To achieve this, I categorized the 'mood' into three distinct classes: 'low' for mood scores from 1 to 6, 'medium' for a score of 7, and 'high' for scores ranging from 8 to 10. This categorization helps in creating a more evenly distributed dataset. Consequently, the mood prediction problem was transformed into a three-class classification issue, which facilitates more balanced training and potentially improves the model's performance across diverse mood states.

Random Forest One approach we have chosen to use the Random Forest classifier to predict mood categories derived from the dataset. Random Forest is an ensemble learning method that operates by constructing a multitude of decision trees during training and outputs the class that is the mode of the classes of the individual trees. It was selected for its robustness and effectiveness in handling both binary and multi-class classification problems.

Long Short-Term Memory Neural Network Another approach we've chosen is the Long Short-Term Memory (LSTM) network, which is a type of recurrent neural network (RNN) suitable for sequence prediction problems. In this case, LSTM can capture time dependencies in mood variations based on sensory data from previous days, making it well-suited for predicting the next day's mood.

The model's architecture begins with an input layer configured based on the number of training data features. It features densely connected layers using the ReLU function for non-linearity, each followed by a Dropout layer to enhance learning and generalization. This pattern continues throughout the model, culminating in a final densely connected layer equipped with the Softmax function, making it well-suited for multi-class classification tasks. In addition, the model uses the Adam optimizer, and compiles with the categorical crossentropy loss function, suitable for multi-class challenges, and evaluates performance based on accuracy.

Hyperparameter optimization The process of hyperparameter optimization for both Random Forest and LSTM models involves systematically tuning the parameters that control the training process and model structure to achieve the best possible accuracy on a given dataset. Below is how the optimization was approached for each model type.

Random Forest In terms of random forest, optimization was performed using GridSearchCV, which is a method that performs an exhaustive search over specified parameter values for an estimator. The parameters tuned were:

- n_estimators: The number of trees in the forest, with values tested being 50, 100, and 200.
- max_depth: The maximum depth of the trees, tested for None (unlimited depth), 10, 20, and 30.

The objective was to find the combination of parameters that results in the best cross-validated performance on the training dataset. And the best parameters found were the combination of {*max_depth*: 10, *n_estimators*: 100}.

LSTM In optimizing the LSTM model, various hyperparameters were adjusted to enhance generalization and prevent overfitting. Critical parameters such as the number of LSTM units (ranging from 32 to 512), dropout rates (between 10% and 50%), and learning rate (from 0.0001 to 0.01) were fine-tuned. These adjustments help capture data relationships, regulate neuron activity during training, and control convergence speed. Additionally, different batch sizes ensured efficient data processing per epoch. To further combat overfitting, a significant number of training epochs were set, complemented by early stopping.

Hyperparameter search was conducted using Random Search through Keras Tuner, which samples subsets of parameters randomly, providing a faster and more computationally efficient alternative to exhaustive searches.

The Evaluation Setup The effectiveness of each model was evaluated using several key metrics, including accuracy, precision, recall, F1 score, and ROC-AUC. These metrics provide insights into various aspects of model performance. In addition, we introduced a Naive Bayes classifier as a benchmark to provide a comparative baseline.

Accuracy serves as a critical indicator of a model's ability to predict mood categories correctly. However, it may lose effectiveness in datasets with imbalanced classes commonly found in real-world scenarios. Therefore, we also evaluate precision and recall: precision is crucial for avoiding false alarms, especially in sensitive situations like medical interventions, while recall ensures the identification of all true positives, particularly vital in critical scenarios such as depression detection. The F1 score combines precision and recall. Furthermore, the ROC-AUC assesses the model's performance across various classification thresholds, crucial for determining the optimal balance between sensitivity (true positive rate) and specificity (false positive rate). Additionally, we use Naive Bayes as a benchmark to establish a baseline for comparing more complex models. Known for its efficiency in handling large datasets, Naive Bayes serves as a reference point to underscore the advancements made by more sophisticated models in mood classification.

Results, Interpretation and Rationale Table 2 reveals that the LSTM model outperformed the Random Forest and Naive Bayes classifiers, achieving the highest accuracy at 72.9%. It also demonstrated superior precision at 75.3%, effectively identifying true positive mood states with fewer false positives. However, its recall rate at 63.6% indicates challenges in capturing all relevant instances, but it still maintained a respectable F1 score of 66.2%, showing a balanced trade-off between precision and recall. The ROC-AUC of 87.6% highlights the LSTM model's strong discriminative ability across various thresholds, making it highly reliable for applications requiring precise mood classification.

In comparison, the Random Forest model recorded slightly lower metrics with an accuracy of 70.1% and a ROC-AUC of 77.6%. Despite its precision being nearly comparable at 73.3%, the model's recall dropped to 58.2%, pointing to difficulties in consistently identifying all positive cases. These challenges are reflected in a lower F1 score of 61.6% and may be attributed to its inability to capture the dynamic temporal relationships as effectively as the LSTM model.

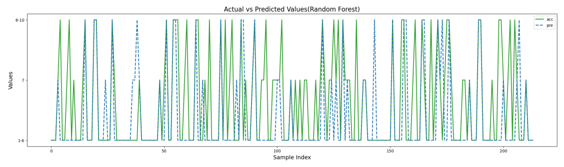
The Naive Bayes classifier, serving as a benchmark, showed the lowest performance across all metrics. It achieved an accuracy of 64.9%, a precision of 60.8%, and a recall of 61.1%, with an F1 score of 60.8%, indicating moderate effectiveness in balancing precision and recall but at a level below the other models. Its ROC-AUC score of 73.7%, while respectable, underscores its limitations in effectively distinguishing between different mood states compared to the more complex models.

These results underscore the LSTM model's proficiency in handling sequential data and maintaining context, essential for accurate mood state prediction influenced by past emotional states and behaviors. This capability, coupled with its high ROC-AUC score, confirms its utility in environments where nuanced mood differentiation is crucial. The lower performance of the Random Forest and Naive Bayes models highlights the importance of model selection based on the specific requirements of the dataset and the complexity of the task at hand.

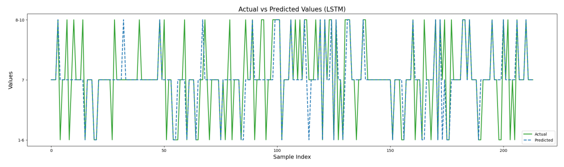
Table 2: Performance Comparison of Classification Models

Model	Accuracy	Precision	Recall	F1 Score	ROC-AUC
LSTM	72.9%	75.3%	63.6%	66.2%	87.6%
Random Forest	70.1%	73.3%	58.2%	61.6%	77.6%
Naive Bayes	64.9%	60.8%	61.1%	60.8%	73.7%

Fig. 6 illustrates the comparison between the predicted values and the actual values in the test set by the random forest and LSTM models.



(a) Actual vs Predicted Values (RF)



(b) Actual vs Predicted Values (LSTM)

Fig. 6: Comparison of Actual vs Predicted Values

2.2 Task 2B: Winning Classification Algorithms

The "ICR - Identifying Age-Related Conditions" competition, held from May 11, 2023 to August 11, 2023, utilized a dataset comprising over fifty anonymized health characteristics associated with three age-related conditions. Additionally, a supplemental metadata file exclusive to the training set offered further insights into the condition types. Submissions were evaluated using balanced logarithmic loss, ensuring that the importance of each category was approximately equal in determining the final scores.

The winner of the competition, Room 722, utilized a sophisticated model called WeightedEns, a custom weighted ensemble that integrated multiple classification algorithms. This model included XGBoost, known for its efficiency and flexibility in gradient boosting, widely recognized across various data science competitions and practical applications. Additionally, it featured TabPFN, an ensemble configuration based on PyTorch that leverages GPU acceleration, significantly enhancing the model's processing speed and learning capabilities.

The winning approach in the competition employed key strategies for preprocessing and model optimization. Firstly, missing values were filled with the median, and categorical variables were converted to binary format. Time data was utilized for feature engineering, enhancing the model's ability to handle temporal variations. Additionally, time features from the training data were extended in the test dataset to simulate future scenarios for predictions. Predicted probabilities were adjusted according to the competition's balanced logarithmic loss function, optimizing the model's sensitivity for categories with smaller sample sizes.

The approach also featured advanced model integration and specialized optimization techniques. By integrating various algorithms, the model captured complex patterns while reducing overfitting risk. During prediction, weighted probabilities were recalculated based on the number of predicted instances per category, tailored to optimize performance for the competition's criteria. Unlike traditional models, this approach directly optimized for balanced logarithmic loss, showcasing its tailored effectiveness within the competition's evaluative framework.

3 Task 3: ASSOCIATION RULES

The concept of association rules was first introduced by R. Agrawal et al.[2] as rules that describe the (potential) relationships that exist between data items (attributes, variables) in a database.

Although the Apriori algorithm is capable of generating association rules effectively, it suffers from two main deficiencies: 1. The algorithm tends to produce excessive redundant rules; 2. Efficiency issues arise as the algorithm requires scanning the entire database for each frequent itemset discovered.

The FP-Growth algorithm proposed by J. Han et al.[1] offers higher efficiency and speed compared to the Apriori algorithm. It achieves this by employing two passes over the database and constructing a compact data structure known as the "FP-tree" (Frequent Pattern Tree), thereby avoiding generating a large number of candidate itemsets. The specific steps are outlined as follows:

1. **Construct the FP-Tree:** Scanning the database to compute the frequency of each item to identify the frequent items and sort frequent items in descending order of frequency to prioritize the search. Then, create the tree's root, labeled as 'null'. For each transaction in the dataset, sort the items by the descending frequency and create a branch. If a branch already exists, increment the count of each node; otherwise, create a new node.
2. **Mine the FP-Tree:** For each item in the item frequency table, create a conditional FP-tree, which is a tree constructed by considering only the

prefix paths leading to the item and from each conditional FP-tree, extract the item's frequent patterns by traversing the tree.

3. **Generate Association Rules:** Use the frequent patterns to generate association rules that satisfy the minimum confidence level and evaluate the rules generated to ensure they are meaningful and useful for the desired application.

Advantages **Efficiency** is one of the most significant advantages of the FP-Growth algorithm. Due to its compact data structure (FP-tree) and two-pass database scanning, the algorithm can efficiently find all frequent itemsets within a shorter period. **Memory utilizatio**n is optimized through the utilization of the FP-tree, where the FP-Growth algorithm reduces storage requirements by compressing transactional data and retaining only essential information. **Scalability** refers to the algorithm's ability to effectively handle large-scale datasets. Typically, the FP-Growth algorithm can effortlessly manage substantial amounts of data.

Disadvantages **Initialization cost** may be high, if there are a large number of items in the transaction database and they are unevenly distributed, constructing the initial FP-tree may consume considerable time. **Not applicable to all data types** indicates that the FP-Growth algorithm is primarily designed for transactional data and may not be suitable for other types of data structures or patterns. **Parameter sensitivity** refers to the possibility of algorithm performance being influenced by parameters such as the support threshold.

4 Task 4: Numerical Prediction

Linear Regression, a staple in statistical modeling and machine learning, predicts a continuous variable using one or more predictors, assuming a linear relationship. It adjusts model coefficients to minimize the residuals between observed and predicted values.

Random Forest Regression, an ensemble method using multiple decision trees, averages their predictions. Effective for complex, non-linear datasets, it doesn't presuppose any data distribution.

For Linear Regression, tuning involves adjusting regularization in Ridge or Lasso variants to balance complexity and generalization. In Random Forest, tuning includes settings like the number of trees, tree depth, and minimum samples per node to manage complexity and avoid overfitting. Both models use tools like scikit-learn's GridSearchCV and RandomizedSearchCV for hyperparameter optimization with cross-validation.

Performance evaluation used an 80-20 train-test split, assessing models via Mean Squared Error (MSE) and Mean Absolute Error (MAE). Linear Regression posted lower errors (MSE: 0.229, MAE: 0.354) than Random Forest (MSE: 0.257, MAE: 0.366).

The lower error rates of Linear Regression might also indicate that this model was able to avoid overfitting more effectively than Random Forest, which can sometimes overfit especially with smaller data sets or without proper tuning of its hyperparameters. This scenario highlights the importance of model selection based on the specific characteristics of the data and the problem at hand. Choosing the simpler model, Linear Regression, was beneficial in this case due to its adequacy for the task, demonstrating that more complex models are not always superior, especially when the underlying data relationships are not overly complex.

5 Task5: Evaluation

5.1 Task 5a: Characteristic of Evaluation Metrics

The Mean Squared Error (MSE) is a measure of the average squared difference between the observed actual outcomes and the outcomes predicted by the model. The formula for MSE is:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3}$$

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

The Mean Absolute Error (MAE) is the average of the absolute differences between the predicted and the actual values. The formula for MAE is:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{4}$$

where $|y_i - \hat{y}_i|$ denotes the absolute value of the difference between the actual and the predicted values.

MSE is preferred when accurately predicting extreme mood swings is critical, as it emphasizes larger errors in predictions. This is essential in clinical settings where significant deviations could indicate critical changes in a patient's emotional state. MAE, offering a direct average of error magnitudes, is less sensitive to outliers, which could be beneficial for stable patients where predictions of mood are generally consistent, avoiding overreaction to minor fluctuations.

In a scenario where the prediction errors are uniform across all observations, i.e., $|y_i - \hat{y}_i| = k$ for a constant k and for every observation i, the MSE and MAE would yield the same result. This uniformity means that every prediction is off by the same amount; thus, squaring the error or taking its absolute value will not alter the aggregated measure of the errors. This could occur, for example, in a perfectly linear dataset where a regression algorithm predicts values that are consistently above or below the actual values by the same amount.

5.2 Task 5b: Impact of Evaluation Metrics

In assessing patient mood predictions, the linear regression model showed an MSE of 0.2293 against an MAE of 0.3535, suggesting the model does not produce large deviations in predictions, as MSE values close to or less than MAE imply a lack of substantial outliers. This could mean the model is reliable for consistently gauging patient mood without significant unpredictable swings.

On the other hand, the random forest regression model presented a slightly higher MSE of 0.2567 compared to its MAE of 0.3662. This slight increase in MSE suggests that while the model may occasionally predict more extreme mood changes, these are not significantly enough to raise concerns over the model's capacity to predict accurately. It implies the model could potentially capture more nuanced patterns in mood shifts but is not prone to extreme prediction errors.

The comparison between the two models indicates that neither has a considerable edge in terms of the presence of outliers or large errors. However, the Random Forest Regression's marginally higher MSE and MAE may reflect its more complex nature, capturing subtle dynamics in the data at the cost of slightly larger average errors.

References

1. Han, J., Pei, J., Yin, Y., Mao, R.: Mining frequent patterns without candidate generation: A frequent-pattern tree approach. Data mining and knowledge discovery **8**, 53–87 (2004)
2. Srikant, R., Agrawal, R.: Mining generalized association rules (1995)