LSTM to predict student engagement

A study conducted on Lernnavi, an educational web application

Dragos Albastroiu Nathan Chettrit Hongyi Shi

***Abstract*—This project focuses on predicting user engagement in terms of activity on Lernnavi, a Swiss educational web application which provides readings and exercises for high school students. By accurately predicting user engagement, targeted interventions could be developed to improve retention and user experience. The key research questions of our project include predicting engagement based on early interaction patterns, and factors influencing engagement.**

I. Introduction

The ability to predict user engagement has significant implications for the design and development of educational platforms such as Lernnavi. By understanding how users engage with the app, it could be possible to develop targeted interventions to increase user retention and improve the overall user experience. For example, if a student is predicted to disengage from the app after the first 4 weeks, it would be then possible to provide personalized recommendations or support to encourage the student to continue using the platform. Alternatively, if a student is predicted to remain active, Lernnavi could continue to provide engaging content and challenges to maintain their interest and motivation.

A. Research Questions

Given the importance and potential benefits of predicting user engagement on Lernnavi, this project aims to answer the following research questions:

1) Can we accurately predict a user's engagement based on their interaction patterns with the application during the first few weeks?
2) How does engagement differ across users? Are there any discernible patterns or trends in the levels or types of engagement?
3) What are the key factors that influence user engagement on Lernnavi?
4) What are the characteristics or behaviors of users who remain consistently active on the platform?

These questions guide our research and subsequent analysis. By answering them, we aim to provide valuable insights that can be used to enhance user engagement and the overall experience on Lernnavi.

II. Data Description and Exploratory Analysis

A. Dataset

During the initial phase of our project, our primary focus was to find a suitable model that could effectively predict user engagement on Lernnavi. To achieve this, we utilized datasets from a controlled study proposed by Lernnavi. While those datasets were relatively small, containing 12 weeks of data for 300 students, it provided valuable insights and a solid foundation for our experimentation. With this controlled datasets, we were able to carefully test and compare various models, experimenting with different algorithms and configurations. Once we chose the appropriate model, we fine-tuned the model's parameters to achieve optimal performance and mitigate the risk of overfitting. This iterative process allowed us to select a model that exhibited promising results and showed good generalization capabilities.

In the next phase of the project, we leveraged the full datasets from Lernnavi. Those datasets encompassed a significantly larger user base of 30929 users and spanned approximately 90 weeks of data. Utilizing this extensive datasets allowed us to create a more robust and accurate prediction model.

B. Data extraction

The data provided by Lernnavi is made of several datasets. We selected attributes from each of them that we consider relevant and/or could be potential predictors to user engagement to build a new dataset on which we will train our model.

Our approach to data extraction was designed to capture a wide range of user behavior and interactions in Lernnavi, allowing us to develop a comprehensive model for predicting user engagement. By selecting relevant and meaningful attributes, we aimed to maximize the predictive power of our machine learning model and provide valuable insights into how users engage with the platform over time. By doing this selection, we lost some data points since we did not use all the attributes we had in our disposal. For example, from the "event" dataset, we only used 9/13 attributes; from the "transactions" dataset, we used 9 out of 21 attributes; from the "documents" dataset, only 5/10 attributes etc. In the end, this selection process resulted in the utilization of 29 out of 59 attributes, which we deemed most relevant for our analysis and prediction of user engagement.

As a consequence of the attribute selection process, some users data was inevitably lost, particularly for those users who did not participate or engage in any of the specific attributes we had chosen. These users were thus not included in the analysis. This ensures that the dataset used for training and prediction consists only of users who demonstrated meaningful engagement with the attributes we deemed significant for our model.

The newly composed dataset contains 22201 users and has a total of 113 features, within which 3 are categorical features. Among those features, we created a synthetic feature *time_took* which is the difference between the *start_time* and the *commit_time* of each transaction. Before merging the provided datasets, a few features had to be renamed, such as the id of the topic trees or the translated topics.

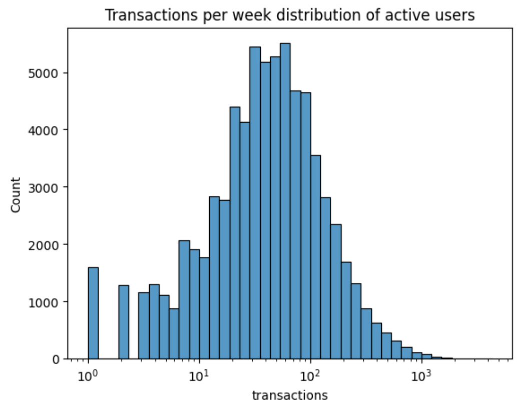

Figure 1: Distribution of the full dataset

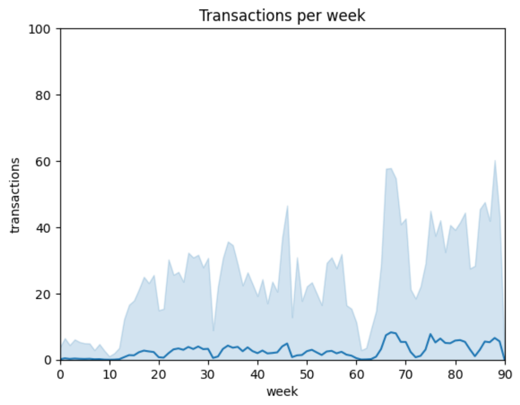

Figure 2: Transactions per week, full dataset

C. Data processing

We processed the categorical attributes by creating dummy variables ; it involves creating new columns for each category in the original attribute, where the value in each column is either 0 or 1, depending on whether the original value matches the corresponding category. Moreover, we performed normalization on the attributes to ensure that they are on the same scale and have equal weight in the model. In addition, all the null values have been set to 0.

We also chose to group the data by week because it provides a meaningful and manageable time frame to analyze user engagement. During the "group by" procedure, we sum up all the features we previously selected. We also added a new attribute : the number of transactions (i.e., the number of questions and answers solved by the user). We consider this new attribute as a baseline for the value we want our model to predict; indeed, the number of transactions is a critical indicator of user engagement since it directly reflects the user's interaction with the platform. By aggregating the data by user and week, we can compare each user's engagement between different weeks and identify any patterns or trends. We can observe the transactions per week a) in Figure 1 for its distribution and, b) in Figure 2 for each week the mean of transaction for all students and a representation of the standard deviation as the area around the mean.

D. Categorization of engagement

In order to gain a better understanding of a student's engagement compared to others, we classified the number of transactions per week per user into an integer value between 0 and 6. This allowed us to compare user engagement on a scale similar to a grade, where a higher value indicated greater engagement. We used different methods to classify the number of transactions, including the quantiles, normal distribution, and transaction/mean ratio (graded on an exponential scale).

Users who did 0 transaction on a whole week get by default 0 as engagement rank.

For the quantiles based classification, we omitted

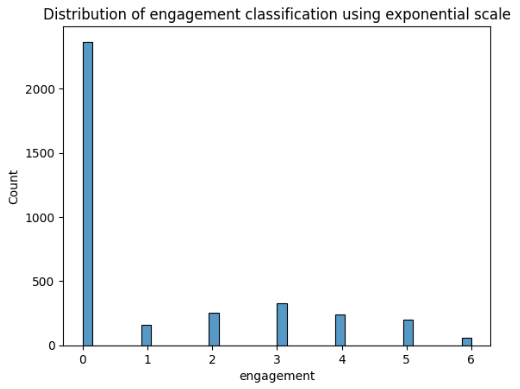

Figure 3: Target classification repartition using exponential scaling

all the 0 week transactions to compute the quantiles as in case of high number of 0 transactions this would mess with the distribution of 1-rank and eventually 2-rank.

Also, the current week engagement rank is being added to the input features, while the next week engagement is being used as label to predict.

III. The Proposed Approach

In our quest to predict user engagement on the Learnnavi platform, we've experimented with three different machine learning approaches: classification using an exponential scale and TimeSeriesForestClassifier from sktime, regression using LSTM (Long Short-Term Memory) to predict the fifth week of engagement based on the first four weeks, and finally regression using LSTM to predict the next four weeks (5th, 6th, 7th, 8th) based on data from the initial four weeks. Each of these methods has its merits and challenges, which we will discuss below.

A. Time Series Forest

Before diving into the more complex LSTM-based approaches, we began with a less intricate model, TimeSeriesForestClassifier, from the sktime library. Starting with a simpler model provided us with a more intuitive understanding of our data and allowed us to establish a baseline performance level for our predictions.

The first approach involves classifying engagement using an exponential scale and employing TimeSeriesForestClassifier from the sktime library. TimeSeriesForestClassifier is an ensemble method that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes of the individual trees [1]. Our choice to use an exponential scale for engagement classification stems from the understanding that user engagement tends to follow a normal distribution when eliminating inactive users.

TimeSeriesForestClassifier is an adaptation of the well-known Random Forest algorithm, suitable for time series data. Just like a regular Random Forest, the TimeSeriesForestClassifier consists of multiple decision trees, and its final prediction is based on the aggregate predictions of these individual trees. However, instead of splitting nodes based on individual features as in a traditional Random Forest, the TimeSeriesForestClassifier splits nodes based on features extracted from subsequences of the time series data, making it more suitable for temporal data [1].

The sktime library is designed specifically for time series machine learning, providing dedicated tools and data structures that streamline the preprocessing, feature extraction, and model training process for time series data. With sktime, we were able to conveniently convert our engagement data into a format that the TimeSeriesForestClassifier could interpret.

By opting for this method, we could leverage the intuitive and interpretable nature of decision trees while also taking advantage of the ensemble method's ability to reduce overfitting, resulting in a robust initial model. We used this as a stepping stone, helping us better understand our data and shape our more advanced LSTM approaches.

B. LSTM

Next, we employed a recurrent neural network (RNN) approach, more specifically Long Short-Term Memory (LSTM), to perform regression on the data. LSTM networks, a special kind of RNN, are designed to avoid the long-term dependency problem, allowing them to remember their inputs over a long period [2]. In the second approach, we trained an LSTM model to predict user engagement in the fifth week based on the data from the first four weeks. This single-week forward prediction approach allows us to assess the immediate future engagement of the user.

The presence of a large number of features in our dataset, totaling 113, makes it an ideal candidate for implementation using a neural network. Neural networks excel in handling high-dimensional data, primarily due to their capability of automatic feature extraction, effectively rendering manual intervention unnecessary. Especially with deep learning models like LSTM, complex and non-linear relationships between the multitude of inputs and outputs can be effectively modeled, allowing for the intricate patterns and dependencies among the 113 features to be learned. Additionally, the scalability and flexibility of neural networks allow them to manage increased complexity with ease as the dimensionality of the data increases. Unlike some traditional machine learning algorithms that can succumb to the "curse of dimensionality", neural networks are able to efficiently utilize the information abundance offered by large numbers of features. They achieve this by learning to assign appropriate weights to different features based on their relevance to the task at hand. Therefore, our choice to employ LSTM is aptly supported by the high-dimensional nature of our data.

1) Model Architecture: The model we worked with is relatively simple. It is made of two modules; a linear layer and LSTM layers, which can also be called a stacked LSTM layer. Indeed, our stacked LSTM contains 3 successive LSTM layer which increase considerably the depth of the model. It has been showed that the depth of such network is more important than the number of cells as it allows higher dimensionality of the time scale. [3] Between each LSTM layer, there is a dropout layer with a dropout probability of 0.1. The linear layer that follow the stacked LSTM is a fully connected layer that helps to changing the output dimension so it is easier to interpret and fit the target accordingly.

We use the Adam optimizer which is an adaptive optimizer which gives more convenience regarding model parameters.

C. LSTM extended

In our third approach, we extended the horizon of our prediction and used the LSTM model to forecast engagement over the next four weeks (5th, 6th, 7th, and 8th weeks) based on data from the initial four weeks. This model provides a broader view of user engagement trends, offering potentially valuable insights for designing long-term interventions.

In both LSTM-based approaches, we leveraged the PyTorch library, a popular open-source machine learning framework that accelerates the path from research prototyping to production deployment. PyTorch's flexible and intuitive interface enabled us to quickly and efficiently implement our models, offering dynamic computation graphs and a rich set of functionalities [4].

At the core of our methodology is the Long Short-Term Memory (LSTM) model, a type of recurrent neural network that's well-suited for time series data. Unlike standard feedforward neural networks, LSTM has feedback connections, allowing it to process sequences of data. In essence, an LSTM network is capable of learning to recognize patterns over time, making it a powerful tool for predicting future data points in a sequence based on historical data.

Moreover, LSTM units include a 'memory cell' that can maintain information in memory for extended periods, which is crucial when dealing with data where past information is relevant to the future prediction. This is controlled by three distinct gates within the LSTM unit – the input gate, the forget gate, and the output gate. The input gate determines how much of the new information should be stored in the cell. The forget gate decides what information should be discarded from the cell, and the output gate controls how much of the information in the cell should be used to compute the unit's output [2].

Our LSTM models were trained for a total of 500 epochs. This number of epochs was chosen to balance between achieving a satisfactory level of performance and avoiding overfitting to our training data. Throughout the training process, we employed a combination of L1 loss and Mean Squared Error (MSE) loss. L1 loss, or absolute error, is less sensitive to outliers and can prevent the model from overfitting. On the other hand, MSE loss is more sensitive to outliers, making it more suitable for achieving higher accuracy when the data is normally distributed. By combining these two loss functions, we aim to leverage the advantages of both to optimize our model's performance.

Each of these approaches has been carefully selected and tailored to address our research questions and predict user engagement effectively. By comparing their results, we aim to find the most reliable method for predicting user engagement on Learnnavi and enhancing the overall user experience.

IV. Experimental Evaluation

Before training our model, we split the dataset into a training set, a test set and a validation set with a ratio of 0.7, 0.2, 0.1 respectively. Thus, we evaluated the accuracy of our model using the respective loss function of each of our approach on the validation set.

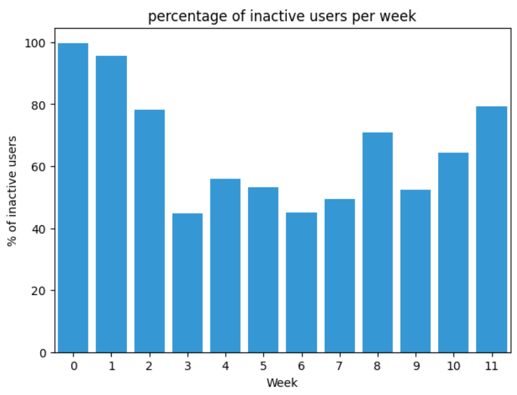

Figure 4: Percentage of inactive users per week

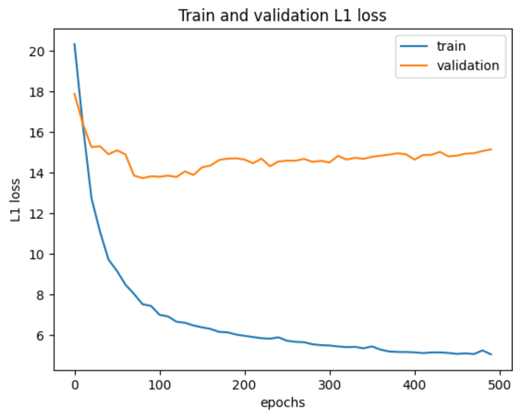

Figure 5: One week ahead train and validation loss

A. *Time Series Forest*

For the first approach which uses the TimeSeriesForestClassifier from the sktime library, we evaluated the model accuracy using the accuracy_score method from the same library. This method computes is specifically designed for evaluating classifiers accuracy. It compares the target sample and the predicted sample and computes how many entries match exactly over the whole given sample. The model manages to achieve an accuracy score of **0.9732** on our dataset which is pretty impressive.

We also evaluated the same model with the balanced_accuracy_score method from sktime library. This method works as the previous one, but takes into consideration the possible imbalancement of the dataset ; if each target class does not have the same number of entries, it will balance it by computing the mean accuracy score over each individual class. The model manages to achieve a balanced accuracy score of **0.9087** on our dataset which is a bit lower than the previous accuracy score. We can deduce that either our dataset suffers from a lack of balance, or that the labeling we applied to our dataset privileges some classes over the other. Indeed, in the statistics of the exponential scale labeling 3, which is the one used here, the number of data points classified to the class 0, very low engagement, is heavily predominant.

Regarding the exponential scaling, we take user weekly transactions and divide it by the weekly mean transaction of all users, than we classify user engagement according to the quotient, according to the scale 0.5, 1, 2, 4, 8. We tried classifying user engagement using different methods beside the one previously mentioned, including one inspired by normal distribution and one that makes use of quantiles. However, in all our attempts, the number of users classified as 0, the lowest engagement, was predominant. We could have enforced even stricter scaling on the lowest engagement classes to achieve a more balanced repartition, however looking into the dataset, on all weeks there was actually a high percentage of inactive users 4. Solutions here could have been to balance the dataset manually after attributions of the labels or the removal of weekly inactive users. However, considering how unbalanced our classes are partitionned, only losing around 7% of accuracy seems reasonable.

B. *LSTM to predict one week ahead*

In this approach, we used the L1 loss, also known as the Mean Absolute Error (MAE). After 500 Epochs, the model achieves a L1 loss of **14.1692**. We can see clear sign of overfitting as the best loss was reached after 160 Epochs, where a loss of **13.7898** was evaluated over the validation set 5. Notice that the loss did not increase considerably even if the training was ran for a lot more time than necessary, showing how efficiently the L1 loss deals with potential overfitting.

The training loss and the validation loss crosses each other after around 10-20 Epochs. Such a quick convergence can be sign that our model is too complex for the task it is asked to do. Indeed, our LSTM model is implemented with a stack of 3 LSTM layers. Increasing LSTM layers helps with dealing with the temporal complexity. However, here, the temporality of our dataset is relatively trivial; predicting user engagement week by week is a linear temporality problem. Reducing the number of layers to two, or even one, could be interesting.

Figure 6 shows some one week ahead predictions of user engagement using the trained model. Despite

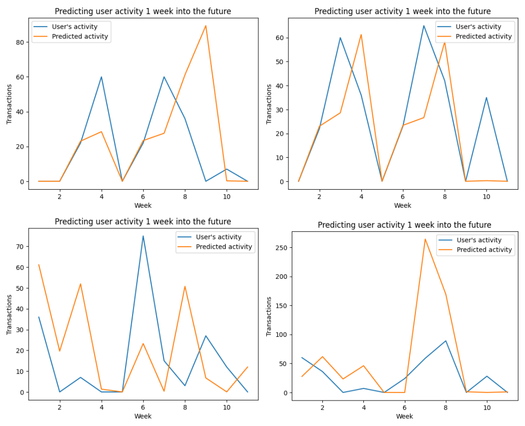

Figure 6: 4 samples of one weeks ahead user engagement predictions

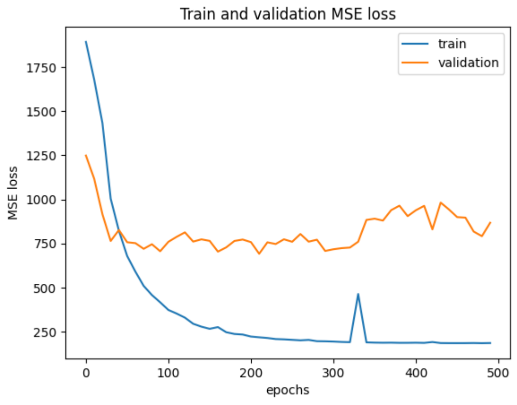

Figure 7: Four weeks ahead train and validation loss

the quick convergence mentioned previously, the prediction follows the trend of the target engagement even though it is far from being accurate. We can see it managed to predict quite consistently the decreases of engagement, however it struggles at predicting correctly the increases, often predicting a smaller increase than it should.

Remember that here, we are predicting only one week ahead. This is particularly visible on the figures as it often shows a delay of one week on engagement peaks.

C. LSTM to predict four weeks ahead

Here, we evaluated the model using the Mean Squared Error (MSE) loss. After 500 Epochs, the model achieves a MSE loss of **867.5751**. The best loss was reached after around 160 Epochs, similarly as with the L1 loss, where a MSE loss of **703.6511** was evaluated over the validation set.

The training loss and the validation loss crosses each other after around 40 Epochs. The convergence is slightly slower than previously with the L1 loss, however this convergence 7 is still hasty once again showing that the model is too complex for its needs.

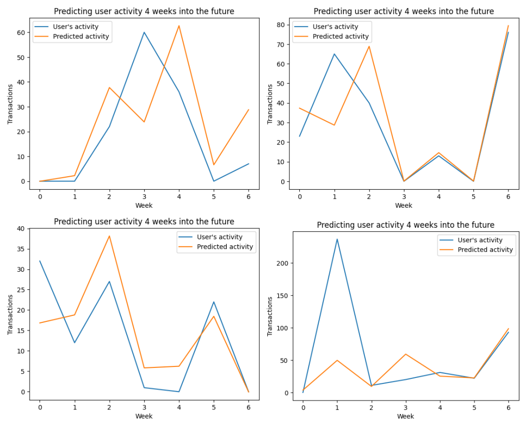

Figure 8: 4 samples of four weeks ahead user engagement predictions

Figure 8 shows some four weeks ahead predictions of user engagement using the trained model. Here, compared to the one week ahead predictions, we can see less frequent one week delay in engagement peaks. The model also manages to predict quite accurately the drops of engagement, while still struggling at predicting engagement increases. However, overall, the predictions look better than the 1 week ahead predictions. Indeed, we can observe that there are more predictions overlapping with the target.

V. Discussion and Implications

The Time Serie Forest classifier works specially well with our labeling method despite an unbalanced dataset. However, we faced some difficulties to compare it with our LSTM model since we did not manage to fit LSTM for classifier and made it works exclusively on regression. This being said, even though the loss throughout trainings do not look as expected and converge very fast, the predictions still show potential, in particular the four weeks ahead ones. By reducing its complexity to slow down the loss convergence point, we may have been able to train for more significant Epochs and eventually reach a better accuracy.

The four weeks ahead predictions working better could be related to our model complexity. Indeed, four weeks ahead is further away than only predicting a single week so it is obviously a more difficult task and requires better tools to achieve. In contrario, predicting only a week ahead is much easier, and since we are predicting weekly engagement, the time dimensionality is rather a linear temporality which might not require three LSTM layers. While the temporality of four weeks ahead predictions is still linear, there are much more unknowns. Indeed, while predicting the 5th week is equivalent as predicting one week ahead, when it comes to predicting the 6th week, it already becomes more difficult since the newly predicted 5th week output may not be accurate, and so it is for week 7th and 8th. With the increase of unknowns, a more complex model seems more adapted.

More generally, LSTM seems to work for predicting user engagement, not as accurately as we wished in our first research question in case of our model, but it shows optimist results. It is especially efficient to predict user engagement drops, which actually, as mentioned in our introduction, is a non-negligible feature as it could helps in providing more resources to those future low engagement users and providing them from dropping the program. Whereas, it is often less important to help students who are not in difficulty. This obviously does not brush out the lack of accuracy of our models which could have been improved as mentioned previously.

An explanation on why our models detect especially well the engagement drops could be the intrinsic low engagement of our dataset. If we look back at Figure 4, on average, there are more than 65% of inactive users per week, meaning users who did not do any transaction and thus number of transaction is 0. This is a huge amount of inactive users, however it is a variable stats from user to user. Indeed, by simple looking at the example in Figure 6 and 8, we can see that those users have quite a few weeks where they are inactive. We can conclude that the 65% of inactive users are in majority not the same each week, and thus, that most users have aggressive drops of engagement once a while before bouncing back the week right after, which can be an answer to our second research question. This pattern being very frequent in our dataset, our models which have been trained on it will then be way more inclined at detecting engagement drops.

Looking a bit more into our dataset, there is actually no user in our dataset who was consistent through all the weeks. Every user has some weeks without any transactions, the minimum of inactive week per user being 4 which is already quite a lot consdering our dataset contains only 12 weeks. The median of inactive weeks per user is 9 9. This being said, we could redefine our meaning of "consistent

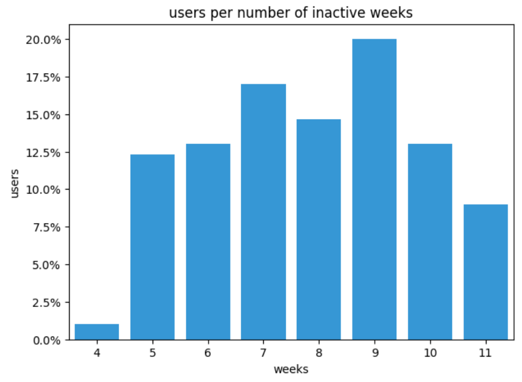

Figure 9: Users per number of inactive weeks

user" and consider them as being users who have 4 or less inactive weeks. Still, this would not be much as this represent roughly 1% of the total users of our dataset. Finally, finding a relevant definition of "consistent user" is harder than we could think and is thus still an open question considering our dataset.

VI. Conclusions and Future Works

Despite a lack of accuracy in the final predictions, the ones generated by the four weeks ahead model seems quite reasonable. We can conclude that LSTM is a relevant machine learning model to predict users engagement. However, there are still several improvements that could have been made to complement our work.

First, regarding the regression model itself, a decrease in the model complexity could have been interesting to try be removing some of the 3 LSTM layers. This way, the convergence point might have been reached much later which may have help in achieving a finer grained accuracy. Also, this could have helped training the full dataset more easily and faster.

Then, still related to the LSTM model, turning it into a classifier in order to fit one of the labeling methods that take the overall weekly users engagement into account could have been interesting. As seen, users engagement throughout the weeks is very variable. There could be weeks were there are not much works to do compared to the other, and so even a few transactions could have meant a high engagement. To turn our model into a classifier, it would have been necessary to add layers in our model, such as a Softmax activation layer in order to allow multi label classification inputs, and change the loss function accordingly, using a Cross Entropy loss or a Hinge loss as example which fit better classifiers.

Some additional steps could have been required such as one-hot encode the labels and update the activation layers accordingly to fit the tensors shape correctly.

References

[1] Markus Löning et al. *sktime: A Unified Interface for Machine Learning with Time Series*. 2019. arXiv: 1909.07872 [cs.LG].

[2] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[3] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. *Speech Recognition with Deep Recurrent Neural Networks*. 2013. arXiv: 1303.5778 [cs.NE].

[4] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.