CS-421 MLBD - Milestone 7 Uclu Masa

Aybars Yazıcı, İlker Gül, Can Kirimca
Department of Computer Science, EPFL, Switzerland

***Abstract*—In this study, we apply time series prediction to understand users' learning paths on the e-learning platform, lernnavi. Using data from prior weeks, we predict a user's mastery level for the following week. Our methods include extracting "hidden" mastery levels, crafting timeseries data, and introducing engagement-based features. To improve data quality, we exclude inactive users. We utilize Linear Regression and Recurrent Neural Network models for prediction, with tailored training strategies for each. Our findings hold potential for enhancing personalized learning experiences, underscoring the significance of user engagement data for predicting educational outcomes.**

I. Introduction

Time series forecasting is a critical and compelling task due to its potential in predicting future events based on historical data patterns and trends. In this study, we aim to leverage these forecasting methods within an educational context. Specifically, we seek to investigate whether user engagement patterns on an e-learning platform, lernnavi, can predict their future progress and mastery of content. By examining the interaction logs from weeks $1...N-1$, we believe that we could potentially predict a user's skill level in week N. This endeavor bears significance as it could aid in tailoring the learning journey to each individual learner based on their predicted progress, leading to more personalized and effective learning experiences.

Consequently, our research question is: *Can we apply time series forecasting to the lernnavi data from weeks* $1...N-1$ *to predict a user's level of mastery in week* N*?*

However, the richness and complexity of learning behaviors suggest that simply examining past interactions may not fully encapsulate a user's learning trajectory. Therefore, we propose to augment this study by considering additional data features such as number of questions, number of views, percentage of correctly solved questions and user interactions with the application. We additionally try to group the students into clusters based on their behavior patterns. These added features will allow us to refine our predictions and could reveal hidden patterns in how users interact with the platform and consequently improve their mastery over time.

Moreover, integrating a comparative analysis of different forecasting models could be another facet to improve this study. By assessing the accuracy of various predictive models, we can ensure that our conclusions are drawn from the most effective forecasting method, strengthening the reliability of our results.

Thus, our research question evolves: *Can we apply time series forecasting to the multidimensional lernnavi data from weeks* $1...N-1$*, enriched with additional data features, to more accurately predict a user's mastery level in week* N*? And which predictive model will provide the most reliable results?*

II. Data Description and Exploratory Analysis

In this section we explain all of our pre-processing step the code of which can be found in the file $lernnavi\_preprocess.py$.

A. Parsing the mastery level

Our first job was to parse the mastery level from the data given to us, as it was not available as a feature out of the box but rather required some tinkering. The "hidden" mastery level was available as a JSON object serialized to a string in the tracking_data column, in the ACCEPT_PROGRESS, REJECT_PROGRESS and NAVIGATE_DASHBOARD events. We have decided not to use the PROGRESS events but rather the NAVIGATE_DASHBOARD events as we have found that the mastery levels retrieved from the former actions are a lot less in number compared to the latter. There is also the diligence level right next to the mastery level, which we also grab while parsing the mastery level. The function ***load_mastery_array*** iterates over all the rows that have the NAVIGATE_DASHBOARD event and returns an an array where each element is the parsed row, that contains the mastery level, the diligence level, the user_id, topic and the timestamp.

B. Creating the timeseries data

All of the subsections below are done by the function ***get_mastery_dfs***

1) Creating the initial data: Now that we had the mastery level of each student at specific time intervals. (See II-A) We'll be creating one dataframe for each of the topics. We had to create a data in the following format: (user_id — week — mastery)

To create the week column, we parse the timestamps to DateTime and compare it to the earliest event done by that specific user. But because a user can check their mastery level/i.e. navigate to the dashboard multiple times a week, we consider the mastery level for that user for that week to be the max mastery level he/she has attained. This is also

	user_id	weeks_since_first_transaction
0	387604	[10, 11, 12, 13, 14, 15, 16, 17, 19, 23, 24, 2...
1	387605	[4, 6, 13, 14, 22, 26]
2	387615	[0, 1, 2, 4, 55]
3	387643	[39, 40, 41, 42, 46, 47, 52]
4	387644	[0, 1, 2, 3, 5, 9, 10, 11, 12, 14, 15, 16, 18,...
...	...	...
17395	431987	[0]
17396	431989	[0]
17397	431991	[0]
17398	431999	[0]
17399	432020	[0]

17400 rows × 2 columns

Figure 1. Weeks missing for users: obtained by grouping the data by user_id

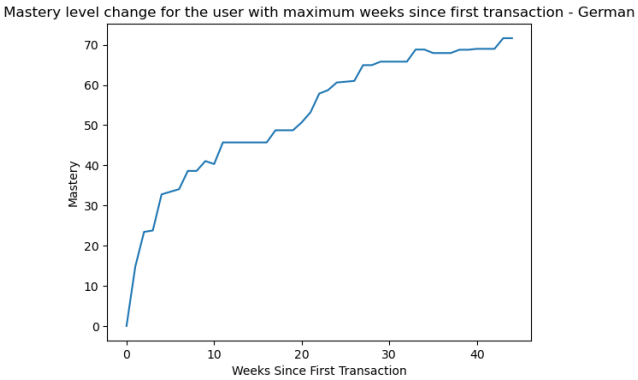

Figure 2. Mastery level change across weeks for the user with the most amount of weeks for the german dataframe

the case for the diligence level. We also notice that there are only 3 unique titles/topics that encompasses all the questions solved, which are

- **Orthografie**: (6,4)
- **Mathematik**: (34981, 4)
- **Deutsch**: (44648, 4)

Seeing that Orthografie only has 6 rows we decided to completely drop it and only consider Mathematics and German as our topics. Thus we drop the rows for Orthografie and split the DataFrame into two, one for german one for mathematics.

Moving forward, a specific trend with the data has caught our attention. Most of the users interact with the platform for the first few weeks(i.e. have multiple entries in the event table) then they stop visiting the website(i.e. there are not entries in the events table for that specific user) for a few weeks. Later they come back to continue to use the platform. Because of this we had to change our definition of a "week" per user. Now because of the fact that we have split the original dataframe into two, some users don't even have their first week as week 0. See Figure 1. Thus, in each topic dataframe we consider the first week of the user to be the first week he/she does an event related to that specific topic. And our week consideration is changed to be the active weeks(weeks that the user has done at least one kind of event) of a user. Meaning that in the case that a user solves some questions the first week then does not login to the platform for 5-6 weeks, and comes back, the 2nd week of that user will be considered the time where the user has returned.

After these preprocessing steps we have our data in the following shape: user — week — mastery — diligence. And we move onto expanding our feature space.

2) Adding features: To be able to predict the mastery level of users, we needed to expand our current dataframe with some features. We decided to use the following metrics each of them split between the German and Mathematics topics:

1) num questions: The number of questions solved by that user in that week.
2) percentage correct: The percentage of correctly solved questions by that user in that week. Partially correct results account as 0.5 correct.
3) num review: The number of review tasks
4) num view: The number of questions viewed

The splitting by topic was done by grabbing the document_id of the event from the documents table, this document id was then mapped to the topics_translated table through the topic_id which gave us the information whether this question solved/reviewed was about Mathematics or German.

One interesting thing we have noticed is that, we have lost some questions due to splitting them by topic, because there are in total 6282 questions for our case where the math column is NaN. This is explained more in detail in the notebook.

C. Pruning the data

1) Users with negligable data: In our new data frame there were users who have interacted with the platform for a really short amount of time, for instance 1 or 2 weeks. Because our task was to predict week N for each user by using the data from weeks 1...N-1. The users with only a few weeks would not be serving us much in our tasks. Thus we had to decided to to only consider users who have interacted with the platform for at least 6 weeks. This gave us 749 users in Math, and 1039 in German.

2) Users with no mastery: There was still an edge case we were yet to consider. There were some small amount of users who have obtained no mastery level during their time spent on the platform(even after 6 weeks). As we were concerned with predicting the mastery level, we have decided that these users would not be valuable to us, and thus have dropped them from the DataFrame. The count of these users were 11 for Mathematics and 19 for German.

D. Clustering of students

To further extend our data, we have looked into student's behaviours and seeked to cluster them utilizing Spectral Clustering [1]. For this we have written the *cluster.py* file. In which we have the following functions:

- **prepare_data**: gets one of the previously created dataframes, a min_week and a boolean value called *scale* to: (1) filter out students that have not interacted with the platform for at least min_weeks (2) create two lists: (a) user_ids: shape[n,1] (b) data: shape[n,min_week] where each column is the number of questions solved by that student at that week. (3) Scale the data array so that each of the n rows have 0 mean and unit std dev, if the *scale* boolean value is set to True. We have used min_week = 6 as previously explained
- **get_distance_matrix**: Calculates the euclidean distance between each student utilizing Dynamic Time warping, given a window size, window size = 0, corresponds to computing the standard euclidean distance.
- **get_affinity_matrix**: Calculates the similarity matrix S, given the distance matrix and γ.
- **get_adjacency**: Calculates the matrix W, given a connectivity method. We most of the time have used a fully connected method, meaning $S = W$
- **get_heuristics_spectral**: Given W and a K which is a set of of possible ks, cluster the data to k clusters for each value of $k \in K$, and plot the eigenvalue and silhouette score associated to each $k \in K$.
- **visualize_clusters**: After assigning each student a label, this function creates a plot for each unique label and plots each students data to their labels plot. See Figure 3

We then have utilized these functions to get eigenvalue and silhouette scores for difference values for: window size & γ See Code block 1. For both scaled = False and True. The reasoning behind scaling the data is simple, the number of questions solved by students vary greatly between each other, and with our clustering our main goal is the capture their behaviour throughout the weeks, i.e. the shape rather than the exact numbers. The plots we have obtained for Math and German datasets are under the images_math/ and images_german/ directories, respectively.

III. The Proposed Approach

A. Models

We used the following models to predict user performance:

- Linear Regression Variants
  - **Linear Regression**
  - **Lasso regression:** Linear Regression with L1 regularization technique.
  - **Ridge regression:** Linear Regression with L2 regularization technique.
- Recurrent Neural Network (RNN) Variants[2]
  - **Long Short-Term Memory (LSTM):** LSTM layer followed by two linear layers. Rectified linear unit (ReLU) as activation function and Adam as optimizer.
  - **Gated Recurrent Unit (GRU):** GRU [3] layer followed by two linear layers. Rectified linear unit (ReLU) as activation function and Adam as optimizer.

We implemented linear regression models using the *scikit-learn* library, while the LSTM and GRU models were developed using *PyTorch*. To increase the complexity of the LSTM and GRU models, two additional fully connected layers were added after their respective recurrent layers.

1) Training and Testing: We employed different strategies for training and testing the regression models and the RNN models. For the Ridge and Lasso Regression models, we used the first five weeks of user data for training and then tested their performance on predicting the sixth week. We experimented with various alpha values for both Ridge and Lasso Regression models and conducted a grid search to determine the optimal alpha value for each model.

On the other hand, for the LSTM and GRU models, we employed a distinct approach to split the dataset in order to train them effectively. Instead of using the same training and testing split as in the regression models, we split the dataset differently.In splitting the dataset for the LSTM and GRU models, we took user-level separation into account. This ensured that the users in the training dataset had all weekly entries up to and including week 6, while the test dataset contained the remaining users, who also had all weekly entries up to and including week 6.

By conducting the train-test split at the user level, we prevented data leakage between the training and testing sets. This approach not only facilitated a more realistic evaluation of the LSTM and GRU models' performance in predicting user performance for week 6 but also allowed us to capitalize on the temporal structure of the data.

Additionally, this methodology enabled us to harness the time-dependent nature of the LSTM and GRU models and train them more efficiently. We carried out training over 50 epochs to ensure adequate model convergence and reliable performance in predicting user outcomes for week 6.

```
import lernnavi_preprocess as pp
import cluster

# Load the preprocessed data
rows = pp.load_mastery_array()
mastery_df_german, mastery_df_math =
↪   pp.get_mastery_dfs(rows)

# Get the data ready for clustering
user_ids, data =
↪   cluster.prepare_data(students=mastery_df_german,
↪   min_week=6, scale = True)

# Compute the distance for each window size we'll be
↪   sweeping
windows = [0,1,2,3,4,5,6]
D_for_window = {
    window: cluster.get_distance_matrix(data,
    ↪   metric='dtw', window=window) if window != 0 else
    ↪   cluster.get_distance_matrix(data, metric='e')
    ↪   for window in windows
}
# Gamma and k values to sweep for
gammas = [0.01, 0.1, 0.5, 1] + np.arange(2, 10,
↪   0.5).tolist() + np.arange(10, 30, 1).tolist() +
↪   np.arange(30, 100, 10).tolist()
n_cluster_list = range(2, 30)

# Begin sweeping, swoosh swoosh :)
for window in vis.tqdm(windows, desc='Sweeping
↪   windows'):
    D = D_for_window[window]
    for gamma in vis.tqdm(gammas, desc='Sweeping
    ↪   gammas'):
        S = cluster.get_affinity_matrix(D, gamma)
        W = cluster.get_adjacency(S)
        df_labels = cluster.get_heuristics_spectral(W,
        ↪   n_cluster_list, save=True,
        ↪   outdir=f'Images/scaled/window={window}',
        ↪   filename=f'gamma_{gamma}.png')
        plt.close('all')
```

Listing 1: Grid search for clustering hyperparameters

Note: Our choice to concentrate on predicting mastery levels for week 6 was driven by the availability of a greater number of users in both training and testing sets, which allowed for a more robust evaluation of our models' performances. Having a larger dataset is essential for achieving precise results. It should be noted, though, that these models can be easily modified to forecast mastery levels for alternative target weeks. By altering the target week and verifying that enough data exists for that week, the same techniques and models can be employed to generate accurate predictions for various time frames

B. Clustering the students

After obtaining the plots of eigenvalue and silhouette scores for different values of window size & γ, we analyzed these plots to find the optimal k (number of clusters) for specific values of window size & γ. Even though the optimal k turned out to be 3 (See Figure 2) for the vast majority of window size & γ pairs, we also experimented on k=2 and k=4 to observe the effect of k on model performances. We also experimented with window sizes 0 (which corresponds to the Euclidean Distance), 1, and 3 for Dynamic Time Warping. Finally, we picked γ values of 0.1, 2, and 5. Then, we clustered the students for each combination of these parameters and obtained a total of 3^3=27 clustered DataFrames for both Mathematics and German, a grand total of 54 DataFrames. Finally, we trained our models on each of these clustered DataFrames and obtained the scores, which we discuss in the Experimental Evaluation section.

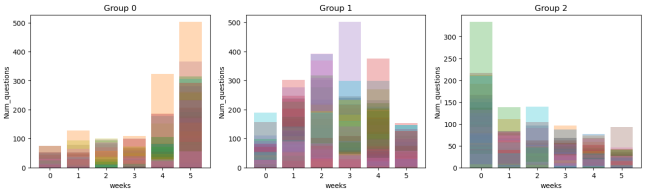

Figure 3. Visualization of clustering using k=3 γ=5, window size=1, *scale*=True

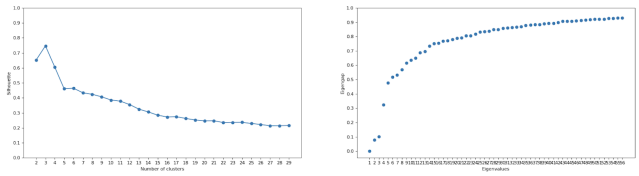

Figure 4. Eigenvalue and Silhouette scores of k values for γ=5, window size=1, *scale*=True

IV. Experimental Evaluation

A. Results before applying clustering

After training each model without adding a cluster attachment, we calculated the MSE, RMSE and MAE values for each model on test data and recorded the results in Tables 1-4. After experimenting with different models, we concluded that LSTM and GRU perform significantly better on our data than Linear Regression variants, with LSTM slightly outperforming GRU. The main reason is that LSTM and GRU are much more complex models and can capture temporal patterns and dependencies in our data much better than Linear Regression, which leads to significantly improved prediction performance. Among regression models, Ridge Regression with an alpha of 1000 performs slightly better than Linear Regression, while Lasso Regression lags behind.

	MSE	RMSE	MAE
Table 1: Test Losses of Regression Models For Math			
Linear Regression	60.25	7.76	5.34
Ridge Regression	60.11	7.75	5.28
Lasso Regression	59.96	7.72	5.18
Table 2: Test Losses of Regression Models For German			
Linear Regression	90.98	9.53	7.06
Ridge Regression	91.15	9.54	7.03
Lasso Regression	99.24	9.96	6.93
Table 3: Test Losses of LSTM and GRU For Math			
LSTM	25.13	5.01	2.64
GRU	23.30	4.82	2.73
Baseline	392.01	19.80	15.23
Table 4: Test Losses of LSTM and GRU For German			
LSTM	23.31	4.82	3.23
GRU	21.21	4.60	2.96
Baseline	598.29	24.46	20.29

B. Regression Evaluation with Clustering

We then have introduced an additional feature, **cluster**, which is determined through the application of the parameters **gamma** and **window**. We optimize these parameters by evaluating two metrics, the silhouette and Eigen scores as explained in subsection II-D. The silhouette score gives us an idea of how well each data point fits within its cluster compared to neighbouring clusters, while the Eigen scores indicate the amount of variance each principal component in the data accounts for so that we were seeking the most effective way to create well-separated, cohesive clusters in our data.

We also wanted to see whether adding this cluster attribute will improve our evaluation scores or will not have positive impact on our evalutation. Initially we will see the results for Math dataset with different gamma(γ), k, and window(w) combinations

As you can see the results from Table I there is slight increase in performance of our regression models when we add the additional **cluster** attribute to the model. When we check with different combinations of gamma(γ), k, and window(w) we were able to decrease below 59. Now we will check the scores for the German Dataset with different gamma(γ), k, and window(w) combinations

Different from the results we have in Math Dataset, when we introduce the cluster attribute to our model for German Dataset there is no major increase in performance of our regression models. By a majority, we see that the performance of the model became slightly worse.

C. LSTM Evaluation with Clustering

Similarly, we wanted to also see whether the performance of our LSTM model will improve after introducing the **cluster** feature or not. In the following you will see evaluation with 27 different results via using combinations of gamma(γ), k, and window(w) values

Table I
REGRESSION MODELS' EVALUATION SCORES FOR **MATH DATASET**

Parameters	Regression	MSE	RMSE	MAE
γ = 0.1, k = 2 w = 0	Linear	58.553	7.652	5.326
	Ridge	58.553	7.652	5.326
γ = 0.1, k = 2 w = 1	Linear	58.523	7.650	5.327
	Ridge	58.523	7.650	5.327
γ = 0.1, k = 2 w = 3	Linear	58.476	7.647	5.330
	Ridge	58.476	7.647	5.330
γ = 0.1, k = 3 w = 0	Linear	58.521	7.649	5.327
	Ridge	58.521	7.649	5.327
γ = 0.1, k = 3 w = 1	Linear	58.312	7.636	5.332
	Ridge	58.344	7.638	5.284
γ = 0.1, k = 3 w = 3	Linear	58.519	7.649	5.327
	Ridge	58.494	7.648	5.281
γ = 0.1, k = 4 w = 0	Linear	58.552	7.651	5.326
	Ridge	58.525	7.650	5.280
γ = 0.1, k = 4 w = 1	Linear	58.534	7.650	5.328
	Ridge	58.508	7.649	5.282
γ = 0.1, k = 4 w = 3	Linear	58.446	7.645	5.331
	Ridge	58.432	7.644	5.284
γ = 2, k = 2 w = 0	Linear	58.540	7.651	5.326
	Ridge	58.540	7.651	5.326
γ = 2, k = 2 w = 1	Linear	58.532	7.650	5.327
	Ridge	58.532	7.650	5.327
γ = 2, k = 2 w = 3	Linear	58.541	7.651	5.326
	Ridge	58.541	7.651	5.326
γ = 2, k = 3 w = 0	Linear	58.525	7.650	5.327
	Ridge	58.525	7.650	5.327
γ = 2, k = 3 w = 1	Linear	58.505	7.648	5.328
	Ridge	58.490	7.647	5.281
γ = 2, k = 3 w = 3	Linear	58.526	7.650	5.328
	Ridge	58.499	7.648	5.281
γ = 2, k = 4 w = 0	Linear	58.637	7.657	5.325
	Ridge	58.590	7.654	5.279
γ = 2, k = 4 w = 1	Linear	58.488	7.647	5.328
	Ridge	58.472	7.646	5.281
γ = 2, k = 4 w = 3	Linear	58.536	7.650	5.325
	Ridge	58.508	7.649	5.279
γ = 5, k = 2 w = 0	Linear	58.391	7.641	5.331
	Ridge	58.463	7.646	5.282
γ = 5, k = 2 w = 1	Linear	58.518	7.649	5.327
	Ridge	58.518	7.649	5.327
γ = 5, k = 2 w = 3	Linear	58.535	7.650	5.326
	Ridge	58.535	7.650	5.326
γ = 5, k = 3 w = 0	Linear	58.580	7.653	5.325
	Ridge	58.580	7.653	5.325
γ = 5, k = 3 w = 1	Linear	58.525	7.650	5.328
	Ridge	58.499	7.648	5.281
γ = 5, k = 3 w = 3	Linear	58.525	7.650	5.327
	Ridge	58.499	7.648	5.281
γ = 5, k = 4 w = 0	Linear	58.529	7.650	5.329
	Ridge	58.504	7.648	5.282
γ = 5, k = 4 w = 1	Linear	58.653	7.658	5.324
	Ridge	58.605	7.655	5.279
γ = 5, k = 4 w = 3	Linear	58.281	7.634	5.322
	Ridge	58.295	7.635	5.277

Before adding the **cluster** feature to our model we were getting **25.13 MSE** score from the model. However, after applying the **cluster** feature to our model as you can see in the Table III our MSE scores varies between 29 to 35. Thus, we can conclude that applying the **cluster** feature worsen the

Table II
REGRESSION MODELS' EVALUATION SCORES FOR **GERMAN DATASET**

Parameters	Regression	MSE	RMSE	MAE
γ = 0.1 k = 2 w = 0	Linear	91.102	9.544	7.043
	Ridge	91.102	9.544	7.043
γ = 0.1 k = 2 w = 1	Linear	91.348	9.557	7.052
	Ridge	91.399	9.560	7.014
γ = 0.1 k = 2 w = 3	Linear	91.252	9.552	7.049
	Ridge	91.268	9.553	7.009
γ = 0.1 k = 3 w = 0	Linear	90.950	9.536	7.019
	Ridge	91.129	9.546	6.992
γ = 0.1 k = 3 w = 1	Linear	91.086	9.543	7.022
	Ridge	91.236	9.551	6.994
γ = 0.1 k = 3 w = 3	Linear	91.178	9.548	7.032
	Ridge	91.178	9.548	7.032
γ = 0.1 k = 4 w = 0	Linear	91.195	9.549	7.028
	Ridge	91.420	9.561	7.002
γ = 0.1 k = 4 w = 1	Linear	91.142	9.546	7.049
	Ridge	91.351	9.557	7.018
γ = 0.1 k = 4 w = 3	Linear	91.095	9.544	7.047
	Ridge	91.310	9.555	7.015
γ = 2, k = 2 w = 0	Linear	91.087	9.544	7.044
	Ridge	91.087	9.544	7.044
γ = 2, k = 2 w = 1	Linear	91.135	9.546	7.045
	Ridge	91.135	9.546	7.045
γ = 2, k = 2 w = 3	Linear	91.178	9.549	7.045
	Ridge	91.264	9.553	7.008
γ = 2, k = 3 w = 0	Linear	91.171	9.548	7.042
	Ridge	91.354	9.558	7.011
γ = 2, k = 3 w = 1	Linear	90.910	9.535	7.052
	Ridge	91.154	9.547	7.015
γ = 2, k = 3 w = 3	Linear	91.181	9.549	7.042
	Ridge	91.416	9.561	7.013
γ = 2, k = 4 w = 0	Linear	91.227	9.551	7.051
	Ridge	91.431	9.562	7.020
γ = 2, k = 4 w = 1	Linear	91.071	9.543	7.045
	Ridge	91.325	9.556	7.015
γ = 2, k = 4 w = 3	Linear	91.382	9.559	7.051
	Ridge	91.568	9.569	7.019
γ = 5, k = 2 w = 0	Linear	91.095	9.544	7.043
	Ridge	91.095	9.544	7.043
γ = 5, k = 2 w = 1	Linear	91.075	9.543	7.047
	Ridge	91.341	9.557	7.014
γ = 5, k = 2 w = 3	Linear	91.198	9.550	7.049
	Ridge	91.309	9.556	7.012
γ = 5, k = 3 w = 0	Linear	91.119	9.546	7.053
	Ridge	91.331	9.557	7.018
γ = 5, k = 3 w = 1	Linear	90.930	9.536	7.050
	Ridge	91.181	9.549	7.014
γ = 5, k = 3 w = 3	Linear	91.298	9.555	7.050
	Ridge	91.445	9.563	7.017
γ = 5, k = 4 w = 0	Linear	91.249	9.552	7.040
	Ridge	91.478	9.564	7.012
γ = 5, k = 4 w = 1	Linear	91.246	9.552	7.043
	Ridge	91.479	9.564	7.015
γ = 5, k = 4 w = 3	Linear	91.412	9.561	7.053
	Ridge	91.592	9.570	7.021

performance of our LSTM model.

Although we believe we have a robust data in terms of seeing the effects of the attributes on mastery label prediction, we are leveraging an LSTM model trained on a dataset comprising 809 individuals and subsequently testing it on another set of 205 individuals. Even though the dataset's size is fairly considerable, it may impose some restrictions

Table III
LSTM EVALUATION SCORES FOR **GERMAN DATASET**

Parameters	MSE	Root MSE	MAE
γ = 0.1 k = 2 w = 0	30.3996	5.5135	3.2858
γ = 0.1 k = 2 w = 1	33.0091	5.7453	3.5419
γ = 0.1 k = 2 w = 3	30.6469	5.5359	3.6702
γ = 0.1 k = 3 w = 0	31.4635	5.6092	3.4042
γ = 0.1 k = 3 w = 1	32.0287	5.6593	3.4989
γ = 0.1 k = 3 w = 3	31.9077	5.6486	3.5747
γ = 0.1 k = 4 w = 0	32.5120	5.7019	3.5349
γ = 0.1 k = 4 w = 1	30.1923	5.4947	3.3586
γ = 0.1 k = 4 w = 3	31.3703	5.6009	3.4117
γ = 2 k = 2 w = 0	34.9444	5.9113	3.4346
γ = 2 k = 2 w = 1	33.1254	5.7554	3.4071
γ = 2 k = 2 w = 3	31.1688	5.5829	3.4693
γ = 2 k = 3 w = 0	31.6267	5.6237	4.0277
γ = 2 k = 3 w = 1	34.3086	5.8573	3.4755
γ = 2 k = 3 w = 3	32.1765	5.6724	3.7132
γ = 2 k = 4 w = 0	30.5426	5.5265	3.3891
γ = 2 k = 4 w = 1	31.2809	5.5929	3.4384
γ = 2 k = 4 w = 3	30.5825	5.5301	3.2700
γ = 5 k = 2 w = 0	29.8226	5.4610	3.5348
γ = 5 k = 2 w = 1	32.7521	5.7229	3.4334
γ = 5 k = 2 w = 3	34.7235	5.8926	4.4001
γ = 5 k = 3 w = 0	30.3630	5.5102	3.3337
γ = 5 k = 3 w = 1	33.7689	5.8111	4.0484
γ = 5 k = 3 w = 3	32.5908	5.7088	3.3397
γ = 5 k = 4 w = 0	33.8081	5.8144	3.6728
γ = 5 k = 4 w = 1	30.4767	5.5205	3.5977
γ = 5 k = 4 w = 3	30.1226	5.4884	3.4150

when we attempt to include a 'cluster' feature, which relies on silhouette and Eigen scores for optimization. Given that efficient tuning of this clustering feature demands a considerable volume of data, the comparatively smaller size of our current dataset may limit the successful application and overall performance of our LSTM model.

The observed discrepancy in model performance when the **cluster** feature is applied to **German** and **Math** datasets could be attributed to a few key factors. Firstly, the inherent characteristics and distribution of data within each dataset might be fundamentally different. Attributes that strongly influence the mastery label prediction in one dataset might not have the same effect in the other. Secondly, the quality and relevance of the clustering achieved through silhouette and Eigen score optimization may differ between the datasets, impacting the effectiveness of the **cluster** feature.

Contrarily, when we implemented the **cluster** feature into our math dataset, there was a noticeable improvement in our model's performance as it is visible in **Table IV**. It's evident that the application of this feature has a distinctive impact, enhancing the efficacy of our LSTM model when dealing with mathematical data. Now we should also check whether it is going to be a similar case for our GRU model for German and Math Datasets as well.

D. GRU Evaluation with Clustering

Moving forward, we are now set to explore the application of the GRU (Gated Recurrent Unit) model to our dataset. Given the results we've observed with the LSTM model,

Table IV
LSTM EVALUATION SCORES FOR **MATH DATASET**

Parameters	MSE	Root MSE	MAE
γ = 0.1, k = 2, window = 0	16.051	4.0063	2.6860
γ = 0.1, k = 2, window = 1	13.477	3.6711	2.6281
γ = 0.1, k = 2, window = 3	23.0299	4.7989	3.2424
γ = 0.1, k = 3, window = 0	17.6408	4.2001	2.4971
γ = 0.1, k = 3, window = 1	15.1596	3.8935	2.5835
γ = 0.1, k = 3, window = 3	21.9962	4.6900	2.8989
γ = 0.1, k = 4, window = 0	20.2894	4.5044	2.7150
γ = 0.1, k = 4, window = 1	18.2627	4.2735	2.9465
γ = 0.1, k = 4, window = 3	15.1817	3.8964	2.6010
γ = 2, k = 2, window = 0	13.1381	3.6246	2.4193
γ = 2, k = 2, window = 1	15.8887	3.9861	2.4729
γ = 2, k = 2, window = 3	11.8777	3.4464	2.2752
γ = 2, k = 3, window = 0	15.4327	3.9284	2.6147
γ = 2, k = 3, window = 1	13.0998	3.6194	2.3680
γ = 2, k = 3, window = 3	11.7564	3.4288	2.4136
γ = 2, k = 4, window = 0	14.3745	3.7914	2.3440
γ = 2, k = 4, window = 1	12.9069	3.5926	2.2888
γ = 2, k = 4, window = 3	16.2984	4.0371	2.4172
γ = 5, k = 2, window = 0	11.0628	3.3261	2.2069
γ = 5, k = 2, window = 1	11.7682	3.4305	2.2858
γ = 5, k = 2, window = 3	18.0000	4.2426	2.5448
γ = 5, k = 3, window = 0	14.9357	3.8647	2.6252
γ = 5, k = 3, window = 1	12.8958	3.5911	2.2660
γ = 5, k = 3, window = 3	13.3097	3.6482	2.4184
γ = 5, k = 4, window = 0	16.2455	4.0306	2.4733
γ = 5, k = 4, window = 1	15.1486	3.8921	2.4147
γ = 5, k = 4, window = 3	10.8130	3.2883	2.2117

Table V
GRU EVALUATION SCORES FOR **GERMAN DATASET**

Parameters	MSE	Root MSE	MAE
γ = 0.1, k = 2, window = 0	31.3319	5.5975	3.6975
γ = 0.1, k = 2, window = 0	31.3842	5.6022	3.5792
γ = 0.1, k = 2, window = 0	33.5626	5.7933	3.6272
γ = 0.1, k = 2, window = 0	32.4280	5.6946	3.3157
γ = 0.1, k = 2, window = 0	33.6764	5.8031	3.5740
γ = 0.1, k = 2, window = 0	33.2668	5.7677	3.3148
γ = 0.1, k = 2, window = 0	32.0569	5.6619	3.5810
γ = 0.1, k = 2, window = 0	33.1190	5.7549	3.4724
γ = 0.1, k = 2, window = 0	31.1899	5.5848	3.4520
γ = 2, k = 2, window = 0	35.3856	5.9486	3.2925
γ = 2, k = 2, window = 1	33.7572	5.8101	3.8752
γ = 2, k = 2, window = 3	34.0932	5.8389	3.4449
γ = 2, k = 3, window = 0	33.8908	5.8216	3.3743
γ = 2, k = 3, window = 1	31.7088	5.6311	3.7396
γ = 2, k = 3, window = 3	33.3187	5.7722	3.4424
γ = 2, k = 4, window = 0	33.1708	5.7594	3.7782
γ = 2, k = 4, window = 1	31.8344	5.6422	3.3022
γ = 2, k = 4, window = 3	32.8824	5.7343	3.3481
γ = 5, k = 2, window = 0	33.2773	5.7686	3.3657
γ = 5, k = 2, window = 1	33.0045	5.7450	3.2440
γ = 5, k = 2, window = 3	33.1609	5.7585	3.3872
γ = 5, k = 3, window = 0	32.3400	5.6868	3.7805
γ = 5, k = 3, window = 1	32.9032	5.7361	3.6834
γ = 5, k = 3, window = 3	32.7424	5.7221	3.4347
γ = 5, k = 4, window = 0	32.2199	5.6763	3.3220
γ = 5, k = 4, window = 1	33.7253	5.8073	3.3353
γ = 5, k = 4, window = 3	32.5162	5.7023	3.5164

our aim is to investigate if the GRU model will demonstrate similar performance changes when the **cluster** feature is applied. This exploration will provide valuable insights into the versatility and effectiveness of our clustering approach across different neural network architectures.

Similar to our LSTM model, before adding the **cluster** feature to our model we were getting **25.13 MSE** score from the model. However, after applying the **cluster** feature to our model as you can see in the Table V our MSE scores varies between 30 to 35. Thus, we can conclude that applying the **cluster** feature worsen the performance of our GRU model.

The application of the **cluster** feature to both our LSTM and GRU models has led to an unexpected increase in the Mean Squared Error (MSE) score. This suggests that the performance of both models has worsened after the inclusion of this feature. This decline in performance may be due to the nature of the **cluster** feature itself, which relies on silhouette and Eigen scores for optimization. In essence, these scores are utilized to identify meaningful groups or 'clusters' in our data. However, it's possible that our data doesn't inherently possess such cohesive and distinctive groups, making the addition of the cluster feature detrimental. Additionally, both LSTM and GRU models are adept at identifying and learning intricate patterns in sequential data. Introducing a cluster feature might add unnecessary complexity to the models, hindering their ability to learn effectively from the inherent patterns of the data. Ultimately, this can result in a less accurate model, as observed in our increased MSE scores.

Now, we are going to check the performance change for GRU model for **Math** dataset. By checking the results in Table VI, we might realize some hidden pattern that influences the effect of the **cluster** feature.

Upon careful analysis of our results, it can be conclusively stated that the incorporation of the **cluster** feature proved to be advantageous for the math dataset. The performance of both LSTM and GRU models significantly improved with the introduction of this feature. This observation underlines the contextual effectiveness of the **cluster** feature, indicating its potential to enhance model performance given the right conditions and data characteristics.

V. DISCUSSION AND IMPLICATIONS

After experimenting with different models, we concluded that LSTM and GRU perform significantly better on our data than Linear Regression variants, with GRU slightly outperforming LSTM. The main reason is that LSTM and GRU are much more complex models and can capture temporal patterns and dependencies in our data much better than Linear Regression, which leads to significantly improved prediction performance. Among regression models, Ridge Regression with an alpha of 1000 performs slightly better than Linear Regression, while Lasso Regression lags behind.

Our analysis reveals that while LSTM and GRU models exhibit satisfactory performance in predicting mastery levels for the German and Math datasets, there remains room for improvement. To enhance the effectiveness of these models, we propose two potential future directions. Firstly, data augmentation could be employed, as both LSTM and GRU

Table VI
GRU EVALUATION SCORES FOR **MATH DATASET**

Parameters	MSE	Root MSE	MAE
γ = 0.1, k = 2, window = 0	10.9322	3.3064	2.5392
γ = 0.1, k = 2, window = 1	11.9900	3.4627	2.4516
γ = 0.1, k = 2, window = 3	19.9300	4.4643	2.7843
γ = 0.1, k = 3, window = 0	17.1069	4.1361	2.5201
γ = 0.1, k = 3, window = 1	13.9666	3.7372	2.4178
γ = 0.1, k = 3, window = 3	11.3414	3.3677	2.5333
γ = 0.1, k = 4, window = 0	11.1462	3.3386	2.3965
γ = 0.1, k = 4, window = 1	17.4297	4.1749	2.3427
γ = 0.1, k = 4, window = 3	17.3867	4.1697	2.6674
γ = 2, k = 2, window = 0	16.6404	4.0793	3.2012
γ = 2, k = 2, window = 1	14.2271	3.7719	2.2881
γ = 2, k = 2, window = 3	11.9805	3.4613	2.4258
γ = 2, k = 3, window = 0	16.2298	4.0286	2.5455
γ = 2, k = 3, window = 1	16.8165	4.1008	2.3832
γ = 2, k = 3, window = 3	14.5362	3.8126	2.3042
γ = 2, k = 4, window = 0	13.8625	3.7232	2.4479
γ = 2, k = 4, window = 1	13.7841	3.7127	2.6025
γ = 2, k = 4, window = 3	15.4154	3.9262	2.5956
γ = 5, k = 2, window = 0	17.3276	4.1626	2.3984
γ = 5, k = 2, window = 1	15.2665	3.9072	2.8176
γ = 5, k = 2, window = 3	11.9991	3.3165	2.2832
γ = 5, k = 3, window = 0	13.3318	3.6513	2.4521
γ = 5, k = 3, window = 1	11.6530	3.4137	2.6107
γ = 5, k = 3, window = 3	30.8084	5.5505	3.2978
γ = 5, k = 4, window = 0	23.3402	4.8312	2.7381
γ = 5, k = 4, window = 1	15.2460	3.9046	2.5804
γ = 5, k = 4, window = 3	16.4889	4.0607	2.4990

models tend to perform better with larger datasets. Secondly, extracting additional features that provide more meaningful insights into the dataset could also prove beneficial.

In order to gauge the relative performance of the LSTM and GRU models, we compared their results with a baseline score for both the German and Math datasets. The baseline score was computed by assuming a scenario where all predictions made by the model were 0, and subsequently calculating the corresponding MSE, RMSE, and MAE scores. This comparison offers valuable context for evaluating the effectiveness of our LSTM and GRU models.

Now, **clustering** is a machine learning technique that involves grouping similar instances on the basis of certain features, with the idea being that instances in the same group, or cluster, are more similar to each other than to those in other clusters. In our model, this clustering feature is expected to provide additional insights or characteristics that could potentially improve the precision of our regression predictions.

To create these clusters, we used a technique that requires specifying a gamma value and a window value. The gamma value is a parameter that helps determine how different two instances should be to belong to different clusters, while the window value is used to control the time-dependence of the data, a feature that might be crucial if the mastery value tends to vary with time.

We tested three different gamma(γ) values (0.1, 2, and 5) and three different window values (0, 1, and 3). Furthermore, we deduced that the number of clusters (k) could be either 2, 3, or 4. This resulted in 27 different combinations of gamma values, window values, and numbers of clusters, which we tested separately on the German dataset and the Math dataset.

For the Math dataset, the inclusion of the cluster feature yielded noticeable improvements in our evaluations. The model was more precise in predicting the mastery values, indicating that the feature was able to capture some latent relationships or patterns in the Math dataset. This could be due to several reasons, such as inherent characteristics of the users that are available in the dataset, which could have distinct groupings that are well-represented by the clusters.

However, when we turned our attention to the German dataset, the story was quite different. The introduction of the cluster feature didn't bring about the same enhancement in the evaluation. There could be a multitude of reasons for this. One possibility is that the features in the German language dataset might not have clear groupings, or these groupings do not contribute significantly to the mastery value.

It's also plausible that our chosen parameters (gamma and window values) aren't as effective for the German dataset as they are for the Math dataset. This highlights the importance of understanding the unique characteristics of each dataset, as techniques that work well for one might not necessarily translate to another.

VI. CONCLUSIONS AND FUTURE WORKS

Our study focused on time series prediction on mastery levels for the users of the e-learning platform, lernnavi. For this purpose, we preprocessed the user data by partitioning the user activity based on users' active weeks to enable time-series analysis. We then extracted the mastery and diligence levels of users as well as other features such as the number of questions solved and the percentage of correct answers. We used a variety of models, namely Regression (Linear, Lasso, and Ridge Regression) and RNN variants (LSTM and GRU) to generate our predictions. Our findings revealed that LSTM and GRU significantly outperform Regression models at predicting user mastery levels due to their ability to capture temporal patterns better than Regression models. As an effort to enhance the performances of our models, we clustered the students based on their behavioral patterns. We experimented with a range of different values for the clustering parameters γ, k, and window size. However, clustering the students in this manner did not lead to the expected improvement in the overall performance of our models, but only yielded enhancements over the Math dataset for LSTM and GRU models, disclosing the contextual effectiveness of adding the cluster feature.

To further improve the results obtained in this study, the student behavior can be analyzed and predicted for specific topics of Math and German, enabling even more specialized predictions based on the topic. Our study was not conducted on such a topic-by-topic basis due to the insufficiency of data

within the scope of each individual topic. Given a larger dataset, a topic-by-topic analysis could provide significant improvements in mastery level prediction. Another way to extend this study is to examine the behavior of the students who work on both subjects (Mathematics and German) instead of completely separating the two subjects and analyzing them independently. Such an effort would enable the examination of students' mastery level improvement when they focus on two subjects simultaneously, and its comparison with the case where the student focuses only on a single subject.

REFERENCES

[1] U. von Luxburg, "A tutorial on spectral clustering," 2007.

[2] V. Subramanian, *Deep Learning with PyTorch: A practical approach to building neural network models using PyTorch*. Packt Publishing Ltd, 2018.

[3] F. Fleuret, "Deep learning 12.2. lstm and gru," 2020.