Taaldiertje: Using Support Vector Machine learning to develop student planning skills

Cnossen, D.
d.cnossen@student.tue.nl
1503383

Muyrers, T.H.C.
t.h.c.muyrers@student.tue.nl
0907928

Stappers, D.R.C.
d.r.c.stappers@student.tue.nl
1256114

Toebosch, R.H.A.
r.h.a.toebosch@student.tue.nl
1235294

Eindhoven University of Technology
Eindhoven, The Netherlands

ABSTRACT

As we currently live in an information age, it is essential to prepare the next generation for a professional life characterised by continuous learning. It is important to teach students at an early age how they develop themselves to become effective planners and learners (Christie & Kumar, 2018). However, this is a daunting task to assign to the primary school teachers, who already deal with a high workload (Hagen, 2019; TNO, 2019). In order to relieve some of this workload, Taaldiertje was developed, with the aim of helping children to learn how to effectively plan and learn word lists. Taaldiertje's XAI contains two Support Vector Regression (SVR) models, each using the same 7 inputs and generating one of two outputs: a prediction of the score improvement or a prediction of the amount of time an exercise would take. These two outputs are fed into a comparative algorithm, and results in recommendations for the student, which are presented to the student through a web application running on NodeJS. The AI has been validated using the LeaveOneOut method in order to determine the R-squared error values for both outputs. Taaldiertje is currently not yet a single system, but both the AI and the web application are functioning and ready for merging. Once the systems are combined, Taaldiertje will be ready for deployment, and validation with the target user group could start.

Keywords

Machine Learning, Explainable Artificial Intelligence, Planning, Language, Language Learning, Education, Interactive Machine Learning, Primary School, Reducing Work Load Teacher, Web Application, Self-Regulated Learning, Life Long Learning

1. INTRODUCTION

1.1 Problem analysis

We currently live in an information age, and in order to keep up with the constant flow of information thrown at us, we need to develop skills that help us become lifelong learners (Christie & Kumar, 2018). In order to prepare the next generation for what is essentially a life characterised by continuous learning, it is important to teach children how to become effective planners and learners from a young age.

Yet, monitoring individual planning skills of each and every student would be a daunting task. The task could be assigned to the primary school teachers, however quality is limited by the busy work schedules of teachers: a questionnaire under eight groups of people actively in education show that the introduction of 'passend onderwijs,' while having many benefits, has the downside of increased work pressure on teachers (Dienst Uitvoering Onderwijs & Algemeen Dagblad, 2015). The survey included 1668 participants of which 955 were active in primary school education. The participants concluded that the work pressure with the introduction of this new format of education has further increased which means that teachers lack the necessary time for personalized guidance for all students.

This increased work pressure amongst other factors contributes to the significant increase in burn-out symptoms among primary school teachers. Research performed by the TNO for the Dutch ministry of Education states that 22,1% of all educational workers have burn-out symptoms compared to 15,6% among other workers. This research also concludes that the percentage of educational workers with burn-out symptoms has increased most significantly among primary school teachers compared to teachers in other educational environments such as high schools and higher education (Hagen, 2019; TNO, 2019). Considering these important factors, we aim to assist primary school teachers in teaching students how to plan their homework tasks.

To achieve this task, Taaldiertje is developed. By using explainable artificial intelligence (XAI) integrated in a webapp, Taaldiertje aims to help students aged 10-12 to effectively plan and learn word lists. The AI will be able to predict how long a specific task will take and the potential improvement a student will make by practising, and give planning recommendations based on this. When this system will be implemented in a classroom of primary school students it aims at reducing the workload of primary school teachers while simultaneously providing students with the skills they need in their future professional careers and to fully develop into lifelong learners which this information age requires them to be.

1.2 Related works

1.2.1 Self-Regulated learning

In general, Self-Regulated learning can be described as the ability to initiate, regulate and reflect on activities independently and is based on the interaction between 3 key components: motivation, cognition and metacognition (Zimmerman, 1990). Motivation refers to task selection, initiation of the task and the effort and persistence during the task. Cognition covers conceptual and strategic knowledge and the ability to apply the right strategies for the task at hand. Metacognition is the component that refers to both the knowledge and regulation of the person's own cognitions. This also includes the observation and assessment of one's own acting and thinking (Dörr & Perels, 2019).

But up to 2007 it was only researched among upper elementary grades through college. This was because of the wide belief that children under 10 years old had difficulty coordinating the cognitive and metacognitive processes required to complete complex, multifaceted tasks (Adagideli, Saraç & Ader, 2017). According to research conducted by Whitebread, Bingham, Grau, Pino Pasternak and Sangster (Whitebread, Bingham, Grau, Pino Pasternak, & Sangster, 2007), studies done within a laboratory setting and other studies based on children's self-reported data have been shown to underestimate young children's abilities when it comes to self-regulated learning. While studying children in a natural setting and while performing familiar tasks, it showed that young children are indeed capable of and engage in self-regulated learning (Whitebread & Coltman, 2010).

1.2.2 Explainable AI

Research about explainable AI often goes hand in hand with research looking into making reliable systems that can cooperate with humans (Tulli, 2020). Making sure that the expected agent behaviour closely resembles the actual agent behaviour secures a better cooperation between the human and the system (Rader, Cotter, & Cho, 2018). Research done by Friedrich and Zanker (2011, p. 90) classifies AI explanation in two categories: "white box" and "black box". How explanations describe a "white box" system and explains elements such as inputs and outputs and what the system does to get to particular outcomes. These explanations bridge the gap between the working system and the user interacting with the system. Why explanations treat the system as a "black box", justifying the system's actions and choices, but these explanations do not explain how the system comes to certain conclusions. Why explanations do help the user determine whether their goals match the system's goals, and when users feel like they understand a system's recommendations, they'll be more comfortable following these recommendations for themselves (Sinha & Swearingen, 2002).

The research done by Rader, Cotter, and Cho (2018) also describes different functions of transparency within systems: creating awareness that there's a working algorithm in a system, evaluate the correctness of the outputs users might experience when working with a system, making the system's behaviour interpretable and helping users feel comfortable acting on the outputs and governing a system through accountability.

1.2.3 Existing solutions

JARET is an AI powered web application designed to help students create a homework and study schedule (Schwabe, 2020). The interface is a weekly calendar view where students can double click to add events and goals such as classes, tests and due dates. The student can also add an estimate for the required time this task will take. JARET also has settings where students can indicate certain preferences such as when they want to study (morning/evening) or when to review knowledge for tests (in advance/hours before). JARET's AI system will then recommend a schedule based on the student's inputs and preferences, which students can edit and then load into their personal calendar app. JARET is currently still in development and they are working on self-reflection and finer adjustments.

JARET is capable of generating a planning based on a set amount of inputs in combination with user preferences. What JARET lacks is a system to automatically keep track of results of the performed tasks. Since Taaldiertje is aimed at younger students, result tracking might be beneficial to them in order for Taaldiertje to effectively help these students. Furthermore, JARET is a tool which can propose a planning students can then import into their own personal calendar, where primary school students might benefit from a Taaldiertje system where everything is in the same place.

FRACTOS focuses on the mathematical problem of fractions, and aims to help children with this mathematical challenge by providing a virtual peer and a system based on simplifying fractions using LEGO style blocks in order to help the child (Krishna, Pelachaud, & Kappas, 2020). This system is also heavily based on self-regulated learning and aims to emphasize the child on the different phases such as planning, performance and reflection. FRACTOS is a Unity3d built game that can be played on a tablet together with a virtual tutor and a robot peer which are both partially controlled by a human wizard to make sure the interaction stays on topic. The system was showcased to the public for a pilot interaction with 25 children at a science festival. It gathered positive feedback on agent perception and task engagement. Their next step is to deploy FRACTOS in primary schools.

FRACTOS provides a proper interaction between the child and the system which remains on topic. Furthermore it focuses on the mathematical problem of fractions, which means that the setting is very on topic and controlled when using the system since it doesn't divert to various different other topics within mathematical knowledge.

2. METHOD & MATERIALS

The related Dutch word-learning platform WRTS, was considered as a source of inspiration for this project (WRTS, n.d.). Within this methodology, it will be explained how our regressor predicts the time and the score of the next learning session through two separately trained models. These outputs are necessary to determine the optimal planning for the student. The aim of the AI is that the student should master each list sufficiently, with a preference for the list(s) that the student will achieve most progress.

2.1 Methods & Approach

2.1.1 Regression

A Support Vector Regression (SVR) is used for multiple independent variables (inputs) that have a plausible correlation with one or more dependent variables (outputs) (NCSS Statistical Software, n.d.). Hence, it was decided to use two SVRs for Taaldiertje. Regression has several methods of prediction including linear and nonlinear regression. Linear regression does not have to be linear in its plot, but it is always linear in its parameters. Consequently, the linear regression line could fit other shapes of data (U-form) since the independent variables could be squared in the formula, while the parameters will have a linear relation (NCSS Statistical Software, n.d.). Still, linear regression is mostly applied to get an estimation of a relation or when there is certainty that the relation will be linear. For more complex problems which do not fit a linear model, a nonlinear regression should be used (NCSS Statistical Software, n.d.). As there was no indication that the data would behave linearly the decision was made to use a nonlinear regression model.

The two regression models will be trained on general user data. After deployment they will be partially fitted with the data from the specific user, to get more accurate results for each individual student.

2.1.2 Inputs & Outputs

Seven inputs have been defined for the AI, with two corresponding outputs, one for each regressor (Appendix IV). These inputs were selected as they are easily accessible within the online environment, and because we suspected they would be indicative of the selected outputs:

- Proficiency Level [1 or 2]
- Dutch [true or false]
- English [true or false]
- Amount of Words [n]
- Previous Time [in minutes]
- Times Practiced [n]
- Percentage Correct [in %]

The input 'Proficiency Level' specifies at which speed the student learns either Dutch or English words, depending on the list at hand, and is determined by the teacher (ScienceGuide, 2017). Level 1 means that the student will need relatively much time to learn words of the corresponding language, while level 2 means the student will learn this language relatively efficiently. Due to how the education program of Dutch preliminary schools is set up, the language choices are limited to two inputs, namely 'Dutch' and 'English' (Persson, 2017). Dutch word lists are used for vocabulary training using synonyms, and English lists are used for learning English translations. Within the current AI there is no distinction between the different learning directions (English to Dutch, or Dutch to English). As regressors do not accept categories as inputs, the languages each have an individual column with true (1) or false (0) as inputs.

Further, the inputs 'Amount of Words', 'Previous Time', 'Times Practiced' and 'Percentage Correct' are used since these are suspected to correlate with the predicted time and score. For example, it is plausible that it takes more time to master a list with more words, because the list is actually longer and the student has to memorize more words.

2.1.3 Datafile

In order to train the AI, a datafile was generated. WRTS was contacted for potentially providing statistics of learning performances as a precedent for the generation of this datafile, but unfortunately could not share it with us. Additionally, there was no literature found that could provide the specific information needed. Hence, the datafile was generated based on personal experience. Furthermore the datafile is relatively small to test the performance of the AI with limited training data. Although it is not ideal to use a made-up dataset, it could give an indication whether the selected model would be suitable for the task at hand.

The nonlinear regression was implemented using Jupyter Notebook 5.0.0 with Python 3 (Pérez & Granger, 2015). With the pandas library (The pandas development team, 2020), a data frame was created. The variables were further converted to NumPy arrays to obtain a readable format. The use of other libraries and the working of the learning algorithm will be explained in the next section (2.2 Learning Algorithm).

2.1.4 Web application

For the embodiment of the learning algorithm, a webapp with an avatar was created (Figure 1). The advantages of using a webapp is that it provides a very controlled environment, in which the AI can directly obtain its necessary inputs, and external interaction with the agent is inherently limited. The webapp is made using NodeJS (OpenJS Foundation, 2020), ReactJS (ReactJS, 2020), MongoDB, Express (Express, 2020), Redux (Redux, 2020), and Bootstrap (Bootstrap, 2020).

The avatar was created to represent the learning algorithm itself: it was opted to go for cartoon animal, as it can convey information through different cues (e.g. gaze, facial expression, pointing), and as we suspect it to be easier for the child to question its advice: we wanted the child to explore what he/she thinks is best, even if this does not align with the AI's advice. This way the child would have room to learn from his/her mistakes (Dinkmeyer & Dreikurs, 2000).

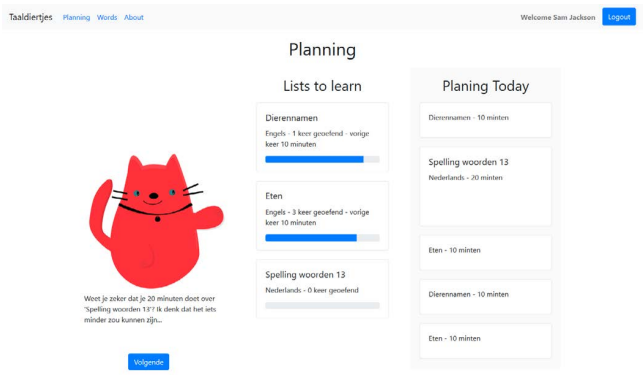


Figure 1: Web application showing the implementation of the AI.

2.2 Learning algorithm

2.2.1 Machine Learning

As mentioned previously, the optimal solution for predicting the time estimation and the score prediction after practicing a list is through a regression. Several options within regression were explored, but due to the nature of the problem a decision was made to use Support Vector Machine (SVM) (Noble, 2006). The opportunities of this method include the opportunity to use different dimensions of data and it does not suffer massively from limited samples (Durgesh & Lekha, 2010). Furthermore the method has proven to provide excellent accuracy on well-defined problems with clear inputs and outputs (Son, Kim & Kim, 2012). For these reasons the choice was made to implement SVM learning into the planning AI used in Taaldiertje.

2.2.2 Scikit-learn

Scikit-Learn (Sklearn) is an open source, commercially usable library developed to implement machine learning in the Python programming environment (Pedregosa et al, 2011). Sklearn is developed to be used both by experienced and inexperienced programmers and provides the user with examples for their different regression, classification and clustering algorithms. The SVM example code provided a strong and verified basis for the

creation of the Taaldiertje planning AI (Pedregosa et al, n.d.-a). Both fitting the data and predicting outputs based on training data was already functional. Simply inserting the training data, based on the planning problem which this paper focuses on, immediately provided the opportunity to predict outputs based on inputs. This provided room to focus on the optimization of the AI, by investigating the effect of the Regularization parameter and the Epsilon parameter (See chapter 2.2.3 & 2.2.4).

2.2.3 Optimization of Regularization parameter (C)

The model can be optimized on two main parameters. The first parameter is the Regularization parameter (C). This influences the model complexity, increasing C will force the model to attempt to incorporate individual data entries more in the prediction of the model (Zhang, Li & Tsai, 2010). This means the model will be altered to prevent underfitting. Complexity is increased by being more precise, straying away from an averaging prediction curve. This allows it to better predict inputs similar to existing training data. This however comes at the cost of risking overfitting (Zhang, Li & Tsai, 2010). Putting a higher importance on incorporating individual data points means that the model could simply output the learning data output and not adapt to new inputs. This obviously leads to increased error rates as new inputs do include a certain randomness based on human inconsistency.

Due to the fairly simple model combined with the small datafile the optimal Regularization parameter (C) can be found by monitoring the R^2 value while systematically changing the C value. The model is tested using the same method used to test the initial model (see chapter 3.3.1 Validation). In Figure 2 the results for the optimization of the predicted score SVM is visualized. The optimal C value for the predicted score model is C=1024|R2=.94. In Figure 3 the results for the optimization of the predicted time required for practicing the list is visualized. The optimal C value for the practice time estimation model is C=64|R2=.91. The improvement from the base model is significant and provides the model with trustworthy validated results.

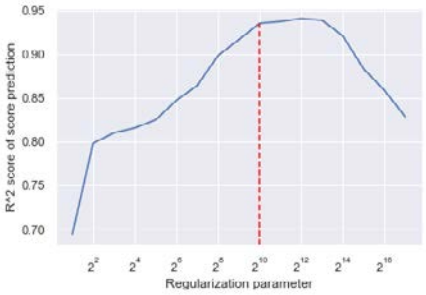


Figure 2: Test results of the Influence of the Regularization parameter in the score prediction SVM model. The R^2 error calculated using LeaveOneOut is visualized against the different tested regularization parameters.

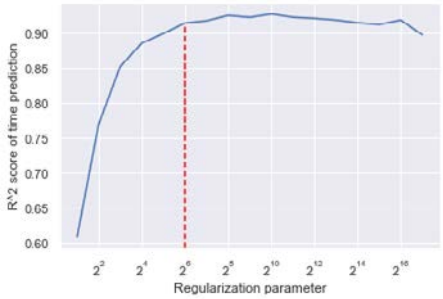


Figure 3: Test results of the Influence of the Regularization parameter in the time prediction SVM model. The R2 error calculated using LeaveOneOut is visualized against the different tested regularization parameters.

2.2.4 Optimization of the Epsilon parameter (e)

The second free parameter in SVM models is the Epsilon parameter. This variable influences the free play that is allowed within the model before a penalty is applied. This allows the model more freedom to find patterns and prevent overfitting. The optimal value for this variable can be found through an algorithm specialized in discovering the clusters in the training data (Ester, Kriegel, Sander, & Xu, 1996). Alternatively similar to the Regularization parameter the method of trial and error can be used to monitor the error rates while inserting different e values.

In order to find the optimal Epsilon value for the prediction model the method proposed by Ester, et al (1996) was used. Within Python this is implemented by looking at the clusters with arbitrary shapes (Maklin, 2019). This resulted in the output shown in Figure 4. The optimal Epsilon value is defined at the place in the graph where the incline changes, for this analysis epsilon=7.

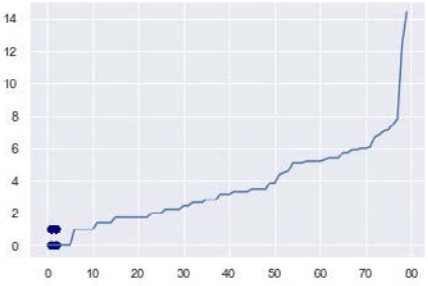


Figure 4: Calculating the optimal epsilon value using the method proposed by Ester, et al (1996)

However, when rerunning the model with the new defined epsilon value showed an increase in error from Rscore2(epsilon=default(.1))=.94 to Rscore2(epsilon=.1)=.88. And Rtime2(epsilon=default(.1))=.91 to Rtime2(epsilon=.1)=.41. For this reason the decision was made to optimize epsilon through model testing and again a for loop was created testing different epsilon values. The optimal epsilon value found for the score prediction model is epsilon=.1| Rscore2=0.94 (Figure 5.). The optimal epsilon value for the time prediction is epsilon=.65| Rtime2=.92 (Figure 6.).

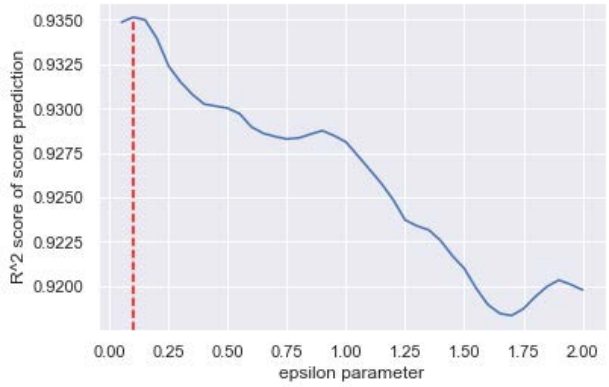


Figure 5: Testing the influence of the epsilon parameter on the score prediction SVM model. The R2 error calculated using LeaveOneOut is visualized against the different tested Epsilon parameters.

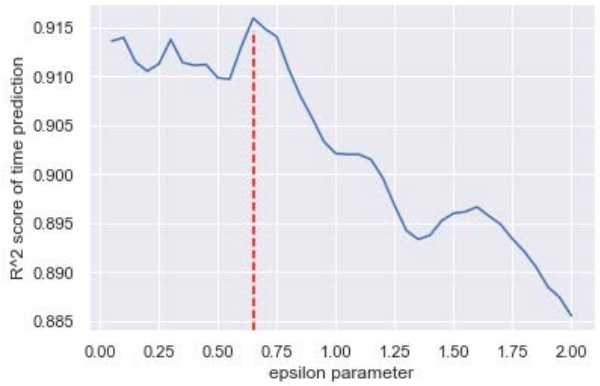


Figure 6: Testing the influence of the epsilon parameter on the time prediction SVM model. The R2 error calculated using LeaveOneOut is visualized against the different tested Epsilon parameters.

2.2.5 Comparative formula

A comparative algorithm was created to determine the 'decision-making' process of the AI, when defining a suggestion. It is important that the agent, Taaldiertje, will first make sure that every list is mastered sufficiently (60% in this design, however changeable according to teacher input). Although, regression and the comparative algorithm could work with multiple outputs; the decision was made to work with two separate outputs. This is because of the small dataset, which will then be run twice for each output; to get a more accurate result.

The output of the comparative algorithm is combined and then weighed against important classifiers that arrange the different lists that the student has to master on improvement potential. This is to make sure that students who excel in a certain learning task don't get advised to spend too much time learning this as the trade-off would be racking up insufficient grades on more difficult tasks.

2.2.5.1 Formula parameters

The comparative algorithm includes the following elements: Current score percentage, Predicted score percentage, estimated time and the amount of words in the list. The formula is optimized through examples. Inputs are generated by the programmers which are then inserted into the formula. The results are examined and the formula altered accordingly. Below the formula will be explained.

First the improvement is calculated by subtracting the current score from the predicted score after practicing the list:

$$improvement = new\_score - old\_score$$

This improvement is sent to a conditional formula, which favours lists which currently score insufficient (below 60% correct) over lists that already have a passing grade (over 60% correct). This is to make sure the student does not prioritize the lists in which he makes progress quickest. Furthermore the effect is also solidified in the multiplication factor, which decreases if the old score is higher. This effect also applies above 60% correct.

$$r_1 = \begin{cases} \left(2 \times \left(1 - \left(\frac{old\_score}{100}\right)\right)\right) \times improvement, & old\_score < 60 \\ 2 \times \left(2 \times \left(1 - \left(\frac{old\_score}{100}\right)\right)\right) \times improvement, & old\_score \geq 60 \end{cases}$$

Next the amount of words is included in the classification. Students who improve a list by 5% which contains 20 words are inferior to students improving 5% on a list containing 50 words.

$$r_2 = 2 \times r_1 \times \left(\frac{amount\_of\_words}{10}\right)$$

The final score that will be used to indicate which list improves the most in the set amount of time is calculated here. All previous calculations are compensated for the time it takes the student to practice the list. This way the AI of Taaldiertje is able to optimize the time the student has available to receive the best result.

$$r_{final} = \left(\frac{r_2}{time}\right)$$

3. DESIGN

3.1 Design of the interaction

The design of the interaction with the webapp takes place in the classroom: every week (or other set period of time), the child has to learn a set of wordlists in either Dutch or English. At the end of the week, the child will have a test on all these lists. During the week, the child will have fixed time available to learn the lists, but is free to choose what lists to prioritize. By doing so, the child gradually learns how to determine which lists to prioritize, and estimate how long it would take to finish the corresponding task. The tasks consist of an automatically generated exercise based on the selected word list, which he/she can then practise with.

On the first day of the week the child receives all the lists, and is able to view them. The child can then go to the planning interface, where it will see Taaldiertje, together with the lists that have to be learned. Students are encouraged to practise all the lists at least once, to give the AI a baseline but also for their own understanding of the task at hand. Taaldiertje embodies the AI,

and the information on the lists which is displayed on the page is also the information that is sent to the AI. Based on this information, Taaldiertje calculates which list would be most beneficial to learn first. It then uses its gaze to subtly provide hints to the user. The child is free to plan whatever lists it thinks are best by dragging it to the planning, and entering how much time he/she thinks the exercise would take. The planning fills up according to how much time the child thinks it will take to do the exercises. When the child deems the planning ready, he/she can click on next, where Taaldiertje will give suggestions on whether it thinks the time estimations are feasible/accurate. The child can choose either to ignore them or change the times accordingly.

The child can then proceed to doing their exercises: after the child selects a wordlist, the web app will automatically create an exercise in which every word is presented and the child has to give the translation. If the answer is right, the next word is displayed; if it is wrong, the webapp shows the correct answer, and will present that specific word again after the other words have been practised. This repeats until the child has answered all the words correctly. At the end of the exercise, the total time is displayed together with a score based on the total amount the student had correct the first try. If the list had been practiced before, previous results are shown as well.

3.2 Intelligent Behaviour and Embodiment

In this concept, the avatar is the embodiment of the AI, and mainly responsible for all explainability surrounding the AI: it makes the algorithm more transparent, as everything the avatar sees, the AI can also "see" and is used as input; it creates awareness of where the AI is functioning, through the avatars presence; it makes the systems behaviour interpretable, though the explanations given by the avatar; it shows the accountability lies with the child, as it ultimately has to make the decision whether to accept or reject the suggestions.

As described before the datafile is created without a reliable connection to learning performances. In the actual design context, it is proposed to get access towards general statistics about learning performances of the target group. This way, a formula could be defined to create a datafile based on realistic learning performances. This datafile will function as a baseline for suggestions when a student uses the AI for the first time. The avatar could intuitively look more or less confident about the suggestion, which is further supported with an optional text explanation; to make the student aware of the suggestion's correctness. The suggestions' quality will increase when the AI is more acquainted with a specific user's learning performances and preference of suggestions. Meaning, the general baseline gets adapted due to the learning behaviour of the AI. The validation of the learning algorithm will be discussed in the next section (3.3 Test and Analysis).

3.2.1 State Diagram

This design is validated based on a state machine. In order to visualize this, a state diagram was created (Figure 7, 8 and 9). It presents the function and change in states of the AI with regard to specific user actions. The decision was made to use a state diagram validation instead of a flowchart, because the design is based on explicit events and not on nodes (flowchart). Hence, the state machine performs actions in response to explicit events that can be detected as sensory readings of the cure state, instead of making 'automatic' node to node transitions. This diagram supports the possibility of going back and forth between states, with an explicit definition of these states.

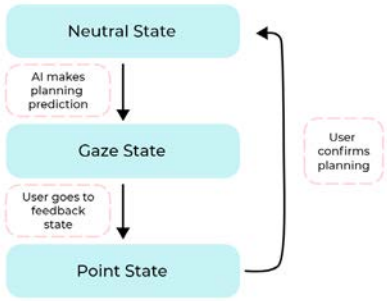


Figure 7: State diagram going from inactive to initial planning phase to reflecting on planning

Figure 8:State diagram during student planning phase. This circular state diagram also applies to the pointing behaviour during the reflecting on the created planning phase.

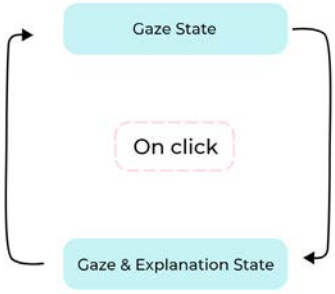


Figure 9: Student can ask for additional explanation by clicking on the virtual agent. The student can ask for additional information in both in planning phase and reflecting on planning phase.

With regard to the learning algorithm used in this project, the state diagram shows the state changes of the AI. The explicit state of the agent pointing at the favoured list, is based on previous information that was fed back via the personalized list scores. Both states of accepting or rejecting the AI's suggestion will be fed back to the personalized AI prediction. Additionally, this happens with the previous score and other inputs; which makes sure that the AI learns from a specific student's performances.

The state diagrams shows the behaviour of the agent during the students planning and reflecting stage. The agent based on previously mentioned inputs gazes at the optimal list to practise. This remains the case until the student makes his decision. At this point the AI recalculates the optimal list which means the gaze could change, but also remain the same if that list is best to practise twice. This same behaviour occurs when the student reviews his planning during the reflecting phase, however here the virtual agent directly points at a list.

3.3 Testing and analysis

3.3.1 Validation

Validation of the created AI is done through the cross validation method LeaveOneOut (Pedregosa et al, n.d.-b). This method extracts each unique input/output combination from the training data and tests this against the predicted variable based on the remaining variables trained model. Due to the looping nature of this method it works best for relatively small datasets, which is applicable to the training data used in the Taaldiertje concept validation. When the data set increases this method could be converted to approaches that apply more random testing, since validity then follows through the randomness in larger datafiles. Examples within the Sklearn library that can be used to validate larger datasets are KFold, ShuffleSplit or StratifiedKFold (Pedregosa et al, n.d.-b).

4. CONCLUSIONS

In this paper, we presented Taaldiertje, a web app aimed to help primary school students effectively plan and learn word lists. We have demonstrated the functionality of this web app together with an extensive validation of the machine learning implemented. Currently, the prototype is split up in two parts: the Web App running in NodeJS and the AI running in Python, and are nearly ready to be combined.

The SVM regressors were optimized and reached an accuracy score of R2= .94 and R2 = .92 for the score- and time-predictions respectively. These scores were calculated using the Leave-One-Out validation method.

A virtual agent, Taaldiertje, is used to make the algorithm comprehensible for the students. The information that is fed to the AI is "visible" both to the agent as well as the student, as they are displayed on the same page. In future iterations, it is envisioned to have Taaldiertje also give a written explanation of what variables it based its decision on.

The Web App is in a state near completion. Currently, the researchers have to input the word lists manually, and the connection to the Python environment is not yet fully functioning due to library issues. Once these issues are addressed, the Web App would be ready for deployment, and validation with the target user group could start.

5. DISCUSSION

Currently, the prototype is split up in two parts: the Web App running in NodeJS and the AI running in Python, and are nearly ready to be put together.

The SVM regressors were optimized and reached an accuracy score of R2= .94 and R2 = .92 for the score-prediction and time-prediction regressors respectively. These are respectable scores, and indicate that the AI should generally give accurate predictions.

The Web App is in a state near completion. Currently, the researchers have to input the word lists manually, and the connection to the Python environment is not yet fully functioning due to library issues. Apart from this, it has all necessary features: the child can log in, see which wordlists they have to learn, plan the wordlists, get feedback from Taaldiertje generated by the Python scripts, train on wordlists, and see results. We therefore see this as a partial success, as it is very near completion, but not finished or validated yet.

5. 1 Limitations

Although the initial results of the SVM regressor are promising, it is important to keep in mind it was trained with a simulated dataset. To see whether the model truly works as intended, sufficient data from actual users should be collected, and the model should be retrained, optimized, and validated accordingly. Furthermore, the current model does not make the distinction between the different directions of learning the language, i.e. translating from Dutch to English is seen as the same as translating from English to Dutch.

Furthermore, when optimizing the SVM, optimal results were found with relatively high C values, which could indicate the simulated data did not have sufficient variance. Also, the epsilon calculation method which was backed up by research did not give the expected results: the error did not improve but worsened when using this technique for validation, so instead another for loop was used to determine the epsilon value.

As mentioned before, the web app still needs some alterations before completion, and needs to be validated. The way we envision this validation can be found in the future work section of this paper.

5.2 Future work

We think Taaldiertje's virtual embodiment still has opportunities for AI explainability: Taaldiertje could intuitively show its current confidence level through e.g. facial expressions and more confident body language. This is a proposed improvement when this design is used in context, which has not yet happened due to time constrictions.

For the validation of the Web App, we envision an experiment in which one or two classrooms of 10-12 year old primary school students are split up. One group of children will be the control group, which will receive the Taaldiertje Web App without the Taaldiertje agent giving suggestions about their planning, but with the AI monitoring their planning and result from behind the scenes. The other group will receive Taaldiertje in its completed form with both the AI and the Taaldiertje agent present, which will give suggestions about the planning like currently envisioned. This way, the functionality and possible benefit of the Taaldiertje agent can be researched.

If Taaldiertje is successfully deployed within a primary school setting to students, it will be able to relieve some of the workload for the teachers. Since the act of planning can be simplified into improvement and time, the AI can predict these scores with high accuracy. Students will be more efficient when it comes to planning their school work and thereby be better prepared for a future in the information age.

If all of this goes to plan, Taaldiertje will be able to give insights into how (explainable) AI can help students with their ability to plan and regulate their learning. This knowledge can help researchers to design for young children to help them learn the valuable skill of self-regulated learning with the help of AI. Providing feedback while allowing students to learn from their mistakes will have a positive effect on the development of students during the rest of their professional career.

6. REFERENCES

Adagideli, F., Saraç, S., & Ader, E. (2017). Assessing preschool teachers' practices to promote self-regulated learning. International Electronic Journal Of Elementary Education, 7(3), 423-440. Retrieved from https://www.iejee.com/index.php/IEJEE/article/view/89

Bootstrap (Version v4.5.3) [Open source front end framework software]. (2020). Retrieved from https://getbootstrap.com

Christie, L. G., & Kumar, G. (2018). THE NEED FOR LIFELONG LEARNING. International Journal of Learning and Intellectual Capital, 1(1), 1. https://doi.org/10.1504/ijlic.2018.10010867

Dienst Uitvoering Onderwijs & Algemeen Dagblad. (2015, August). Rapportage Onderzoek passend onderwijs. DUO Onderwijsonderzoek. Retrieved from https://www.duo-onderwijsonderzoek.nl/wp-content/uploads/2015/08/Rapportage-Passend-Onderwijs-augustus-2015.pdf

Dinkmeyer, D. C., & Dreikurs, R. (2000). Encouraging children to learn. Psychology Press. [Page xi]

Dörr, L., & Perels, F. (2019). Improving Metacognitive Abilities As An Important Prerequisite for Self-Regulated Learning in Preschool Children. International Electronic Journal of Elementary Education, 11(5), 449–459. https://doi.org/10.26822/iejee.2019553341

Durgesh, K. S., & Lekha, B. (2010). Data classification using support vector machine. Journal of theoretical and applied information technology, 12(1), 1-7.

Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996, August). A density-based algorithm for discovering clusters in large spatial databases with noise. In Kdd (Vol. 96, No. 34, pp. 226-231).

Express (Version v5.0) [Fast, unopinionated, minimalist web framework for Node.js]. (2020). Retrieved from https://expressjs.com

Friedrich, G., & Zanker, M. (2011). A Taxonomy for Generating Explanations in Recommender Systems. AI Magazine, 32(3), 90. https://doi.org/10.1609/aimag.v32i3.2365

Hagen, K. (2019, June 19). Burn out-probleem groeit snelst in primair onderwijs. Retrieved 25 October 2020, from https://www.aob.nl/nieuws/burn-out-probleem-groeit-snelst-in-primair-onderwijs/#:%7E:text=In%20het%20primair%20onderwijs%20(po,naar%2024%2C1%20in%202017.

Krishna, S., Pelachaud, C., & Kappas, A. (2020). FRACTOS. Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction, 1. https://doi.org/10.1145/3371382.3378318

Maklin, C. (2019, July 14). DBSCAN Python Example: The Optimal Value For Epsilon (EPS). Retrieved October 20, 2020, from https://towardsdatascience.com/machine-learning-clustering-dbscan-determine-the-optimal-value-for-epsilon-eps-python-example-3100091cfbc

MongooseJS (5.10). (2020). [MongoDB object modeling designed to work in an asynchronous environment.]. MongoDB. https://mongoosejs.com

NCSS Statistical Software. (n.d.). Chapter 315: Nonlinear Regression. Retrieved October 25, 2020, from https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Nonlinear_Regression.pdf

Noble, W. S. (2006). What is a support vector machine?. Nature biotechnology, 24(12), 1565-1567.

OpenJS Foundation (15.0.1). (2020). [Open Source Software]. Joyent Inc. https://nodejs.org/en/

Pedregosa et al. (n.d.-a). Support Vector Machines. Retrieved October 08, 2020, from https://scikit-learn.org/stable/modules/svm.html

Pedregosa et al. (n.d.-b). LeaveOneOut . Retrieved October 08, 2020, from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.LeaveOneOut.html

Pedregosa et al. (n.d.-c). LeaveOneOut . Retrieved October 08, 2020, from https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html

Pérez, F., & Granger, B. (2015). Jupyter Notebook (5.0.0) [Python 3]. Project Jupyter. https://jupyter.org/

Persson, L. (2017, March 17). Alvast een nieuwe taal leren op de basisschool. NEMO Kennislink. https://www.nemokennislink.nl/publicaties/alvast-een-nieuwe-taal-leren-op-de-basisschool/

Rader, E., Cotter, K., & Cho, J. (2018). Explanations as Mechanisms for Supporting Algorithmic Transparency. Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18, 103:1-103:13. https://doi.org/10.1145/3173574.3173677

ReactJS (17.0.1). (2020). [Open Source Software]. Facebook Inc. https://reactjs.org/

Redux. (2020). Predictable state container for JavaScript apps. Retrieved from https://redux.js.org

Schwabe, A. (2020). Demo of JARET: A.I. Powered Web App for Goal Review and Time Management. Proceedings of the Seventh ACM Conference on Learning @ Scale, 1. https://doi.org/10.1145/3386527.3405954

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

Sinha, R., & Swearingen, K. (2002). The role of transparency in recommender systems. CHI '02 Extended Abstracts on Human Factors in Computing Systems - CHI '02, 1. https://doi.org/10.1145/506443.506619

Son, H., Kim, C., & Kim, C. (2012). Hybrid principal component analysis and support vector machine model for predicting the cost

performance of commercial building projects using pre-project planning variables. Automation in Construction, 27, 60-66.

The pandas development team. (2020). pandas-dev/pandas: Pandas (latest) [Python library]. Zenodo. https://pandas.pydata.org/

TNO. (2019, June). WERKDRUK IN HET ONDERWIJS. Author. Retrieved from https://www.rijksoverheid.nl/ministeries/ministerie-van-onderwijs-cultuur-en-wetenschap/documenten/rapporten/2019/06/06/werkdruk-in-het-onderwijs

Tulli, S. (2020). Explainability in Autonomous Pedagogical Agents. Proceedings of the AAAI Conference on Artificial Intelligence, 34(10), 13738–13739. https://doi.org/10.1609/aaai.v34i10.7141

Whitebread, D., Bingham, S., Grau, V., Pino Pasternak, D., & Sangster, C. (2007). Development of Metacognition and Self-Regulated Learning in Young Children: Role of Collaborative and Peer-Assisted Learning. Journal of Cognitive Education and Psychology, 6(3), 433–455. https://doi.org/10.1891/194589507787382043

Whitebread, D., & Coltman, P. (2010). Aspects of pedagogy supporting metacognition and self-regulation in mathematical learning of young children: evidence from an observational study. ZDM, 42(2), 163–178. https://doi.org/10.1007/s11858-009-0233-1

WRTS (n.d.). Alle voordelen van WRTS op een rijtje.https://wrts.nl/voordelen

Zhang, Y., Li, R., & Tsai, C. L. (2010). Regularization parameter selections via generalized information criterion. Journal of the American Statistical Association, 105(489), 312-323.

Zimmerman, B. J. (1990). Self-Regulated Learning and Academic Achievement: An Overview. Educational Psychologist, 25(1), 3–17. https://doi.org/10.1207/s15326985ep2501_2

7. APPENDIX

Appendix I: Python code
Appendix: II: Datafile
Appendix III: Images webpage
Appendix IV: Overview Inputs & Outputs

Appendix I: Python code

```
[1]  import numpy as np
[2]  import pandas as pd
[3]  import matplotlib.pyplot as plt
[4]  import seaborn as sns
[5]  import matplotlib.lines as mlines
[6]  import matplotlib.transforms as mtransforms
[7]
[8]
[9]  from sklearn.svm import SVR
[10] from sklearn.model_selection import LeaveOneOut
[11] from sklearn.metrics import accuracy_score
[12] from sklearn.metrics import r2_score
[13] from sklearn.neighbors import NearestNeighbors
[14] from sklearn.cluster import DBSCAN
[15]
[16] sns.set()
[17]
[18] #The training data gets loaded
[19] df = pd.read_excel (r'C:\Users\Tim\Documents\Master\Q1\DBM140 Embodying Intelligent behavior in
     social context\input_data.xlsx')
[20]
[21] #Training data gets sorted into input variables and predictor variables
[22] x=[df.Level, df.Dutch, df.English, df.Amount_of_Words,
     df.Previous_Time,df.Times_Practiced,df.Percentage_Correct_Previous_Session]
[23] Y1=df.O1_Correct
[24] Y2=df.O2_Time
[25]
[26] #Changing the format of the variables to correctly load them into the regressor
[27] y1 = np.asarray(Y1).transpose()
[28] y2 = np.asarray(Y2).transpose()
[29] X = np.asarray(x).transpose()
[30]
[31] #Showing how the training data looks
[32] #print(X)
[33] #print(y1)
[34] #print(y2)
[35]
[36] print("Fitting data to model")
[37]
[38] #Here we twice load in the regression model and fit the training data to the two seperate regressors
[39] regr1 = SVR(kernel='rbf')
[40] regr1.fit(X, y1)
[41] regr2 = SVR(kernel='rbf')
[42] regr2.fit(X, y2)
```

```
[43]
[44] print("Fitting data to model completed")
[45]
[46]
[47] Level_i=2 ##Input from webpage
[48] Dutch_i=1 ##Input from webpage
[49] English_i=0 ##Input from webpage
[50] Amount_of_Words_i=20 ##Input from webpage
[51] Previous_Time_i=12 ##Input from webpage
[52] Times_Practiced_i=2 ##Input from webpage
[53] Percentage_Correct_Previous_Session_i=35 ##Input from webpage
[54] X_i=[Level_i, Dutch_i, English_i, Amount_of_Words_i, Previous_Time_i, Times_Practiced_i,
     Percentage_Correct_Previous_Session_i]
[55]
[56] # Here the model predicts the outcomes
[57] Output_Score=regr1.predict([X_i])
[58] Output_Time=regr2.predict([X_i])
[59]
[60] # Both outputs get show in textual form
[61] print("Predicted percentage correct after practising")
[62] print(regr1.predict([X_i]))
[63] print("Predicted time required for practising the list")
[64] print(regr2.predict([X_i]))
[65]
[66] # Since the algoritm has a preference of getting positives grades first over optimizing one list that
     might be easier
[67] # for the student, it here uses a comparative formula, where factors like improvement, time, amount
     of words learned all
[68] # influence the outcome of the reward score. The AI will choose the highest reward score and advice
     that list to learn.
[69]
[70] if Percentage_Correct_Previous_Session_i<60:
[71]     R1=2*((2*(1-(Percentage_Correct_Previous_Session_i/100)))*(Output_Score-
     Percentage_Correct_Previous_Session_i))
[72]
[73] else:
[74]     R1=((2*(1-(Percentage_Correct_Previous_Session_i/100)))*(Output_Score-
     Percentage_Correct_Previous_Session_i))
[75]
[76] R2=2*R1*(Amount_of_Words_i/10)
[77] Reward_Final=R2/Output_Time
[78]
[79] #Reward_final is the output of the whole AI. The highest number of a list will mean the student can
     best learn this list
[80] print("Number used to choose which list to practise (higher is better):")
print(Reward_Final)
```

```
# LeaveOneOut will be used to extract
LOO = LeaveOneOut()

# This is where we create a list containing the true y1 values and the values the model predicts
# These will be used to run the r^2 validation on
y1_true, y1_pred = list(), list()
for train_ix, test_ix in LOO.split(X):
    X_train, X_test = X[train_ix, :], X[test_ix, :]
    y1_train, y1_test = y1[train_ix], y1[test_ix]
    # We use the model that was fitted above
    regr1.fit(X_train, y1_train)
    # evaluate model
    yhat = regr1.predict(X_test)
    # store
    y1_true.append(y1_test[0])
    y1_pred.append(yhat[0])
    # calculate accuracy


    # This is where we create a list containing the true y2 values and the values the model predicts
    # These will be used to run the r^2 validation on
    y2_true, y2_pred = list(), list()
    for train_ix, test_ix in LOO.split(X):
        X_train, X_test = X[train_ix, :], X[test_ix, :]
        y2_train, y2_test = y2[train_ix], y2[test_ix]
        # We use the model that was fitted above
        regr2.fit(X_train, y2_train)
        # evaluate model
        yhat = regr2.predict(X_test)
        # store
        y2_true.append(y2_test[0])
        y2_pred.append(yhat[0])
        # calculate accuracy

    # Calculation the R2 score for both models
    y1_r2 = r2_score(y1_true, y1_pred)
    y2_r2 = r2_score(y2_true, y2_pred)

    # Showing the errors numerically
    print("Error in percentage correct prediction (R^2)")
    print(y1_r2)
    print("Error in task time estimation (R^2)")
    print(y2_r2)

    # Showing the errors in plot form
```

```
[126]   fig, ax = plt.subplots()
[127]   plt.gray()
[128]   ax.scatter(y1_true, y1_pred)
[129]   line = mlines.Line2D([0, 1], [0, 1], color='red')
[130]   transform = ax.transAxes
[131]   line.set_transform(transform)
[132]   ax.add_line(line)
[133]   plt.xlabel("True correct after practising score")
[134]   plt.ylabel("Predicted correct after practising score")
[135]   plt.show()
[136]
[137]   fig, ax = plt.subplots()
[138]   plt.gray()
[139]   ax.scatter(y2_true, y2_pred)
[140]   line = mlines.Line2D([0, 1], [0, 1], color='red')
[141]   transform = ax.transAxes
[142]   line.set_transform(transform)
[143]   ax.add_line(line)
[144]   plt.xlabel("True time spent on practising")
[145]   plt.ylabel("Predicted time spent on practising")
[146]   plt.show()
[147]   # Now the model is trained and validated. Now the model can be used to predict on new inputs
[148]   # Here we input the new input variables to use in the predictor algoritm


[149]   #C_val = [2,4,8,16,32,64,128,256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072]
[150]   C_val = [2,4,8,16,32,64,128,256]
[151]
[152]   r1_R2_C_val = []
[153]
[154]
[155]   for i in range(len(C_val)):
[156]       regr1 = SVR(kernel='rbf', C=C_val[i])
[157]
[158]       # LeaveOneOut will be used to extract
[159]       LOO = LeaveOneOut()
[160]
[161]       # This is where we create a list containing the true y1 values and the values the model
    predicts
[162]       # These will be used to run the r^2 validation on
[163]       y1_true, y1_pred = list(), list()
[164]       for train_ix, test_ix in LOO.split(X):
[165]           X_train, X_test = X[train_ix, :], X[test_ix, :]
[166]           y1_train, y1_test = y1[train_ix], y1[test_ix]
[167]           # We use the model that was fitted above
```

```
[168]            regr1.fit(X_train, y1_train)
[169]            # evaluate model
[170]            yhat = regr1.predict(X_test)
[171]            # store
[172]            y1_true.append(y1_test[0])
[173]            y1_pred.append(yhat[0])
[174]            # calculate accuracy
[175]
[176]        y1_r2 = r2_score(y1_true, y1_pred)
[177]
[178]        print("Error in percentage correct prediction (R^2) at " + str(C_val[i]))
[179]        print(y1_r2)
[180]        r1_R2_C_val.append(y1_r2)
[181]
[182]    r2_R2_C_val = []
[183]
[184]    for i in range(len(C_val)):
[185]        regr2 = SVR(kernel='rbf', C=C_val[i])
[186]
[187]        # LeaveOneOut will be used to extract
[188]        LOO = LeaveOneOut()
[189]
[190]        # This is where we create a list containing the true y1 values and the values the model
    predicts
[191]        # These will be used to run the r^2 validation on
[192]        y2_true, y2_pred = list(), list()
[193]        for train_ix, test_ix in LOO.split(X):
[194]            X_train, X_test = X[train_ix, :], X[test_ix, :]
[195]            y2_train, y2_test = y2[train_ix], y2[test_ix]
[196]            # We use the model that was fitted above
[197]            regr2.fit(X_train, y2_train)
[198]            # evaluate model
[199]            yhat = regr2.predict(X_test)
[200]            # store
[201]            y2_true.append(y2_test[0])
[202]            y2_pred.append(yhat[0])
[203]            # calculate accuracy
[204]
[205]        y1_r2 = r2_score(y2_true, y2_pred)
[206]
[207]        print("Error in percentage correct prediction (R^2) at " + str(C_val[i]))
[208]        print(y1_r2)
[209]        r2_R2_C_val.append(y1_r2)
[210]
```

```
[211]  # Showing the errors in plot form
[212]  fig, ax = plt.subplots()
[213]  plt.gray()
[214]  ax.scatter(y1_true, y1_pred)
[215]  line = mlines.Line2D([0, 1], [0, 1], color='red')
[216]  transform = ax.transAxes
[217]  line.set_transform(transform)
[218]  ax.add_line(line)
[219]  plt.xlabel("True correct after practising score")
[220]  plt.ylabel("Predicted correct after practising score")
[221]  ax.set_ylim(30, 100)
[222]  ax.set_xlim(30, 100)
[223]  plt.show()
[224]
[225]  fig, ax = plt.subplots()
[226]  plt.gray()
[227]  ax.scatter(y2_true, y2_pred)
[228]  line = mlines.Line2D([0, 1], [0, 1], color='red')
[229]  transform = ax.transAxes
[230]  line.set_transform(transform)
[231]  ax.add_line(line)
[232]  plt.xlabel("True time spent on practising")
[233]  plt.ylabel("Predicted time spent on practising")
[234]  ax.set_ylim(0, 28)
[235]  ax.set_xlim(0, 28)
[236]  plt.show()
[237]  # Now the model is trained and validated. Now the model can be used to predict on new inputs

[237]  fig, ax = plt.subplots()
[238]  ax.plot(C_val, r1_R2_C_val)  # Plot some data on the axes.
[239]  plt.xscale("log",basex = 2)
[240]  plt.xlabel("Regularization parameter")
[241]  plt.ylabel("R^2 score of score prediction")
[242]  plt.axvline(x=1024, ymin=0, ymax=0.9351415365438123, linestyle = '--',color='red')
[243]  plt.show()
[244]
[245]  fig, ax = plt.subplots()
[246]  ax.plot(C_val, r2_R2_C_val)  # Plot some data on the axes.
[247]  plt.xscale("log",basex = 2)
[248]  plt.xlabel("Regularization parameter")
[249]  plt.ylabel("R^2 score of time prediction")
[250]  plt.axvline(x=64, ymin=0, ymax=0.9139674817757812, linestyle = '--',color='red')
       plt.show()
[251]  fig, ax = plt.subplots()
```

```
[252]   ax.plot(C_val, r1_R2_C_val)  # Plot some data on the axes.
[253]   plt.xscale("log",basex = 2)
[254]   plt.xlabel("Regularization parameter")
[255]   plt.ylabel("R^2 score of score prediction")
[256]   plt.axvline(x=1024, ymin=0, ymax=0.9351415365438123, linestyle = '--',color='red')
[257]   plt.show()
[258]   fig, ax = plt.subplots()
[259]   ax.plot(C_val, r2_R2_C_val)  # Plot some data on the axes.
[260]   plt.xscale("log",basex = 2)
[261]   plt.xlabel("Regularization parameter")
[262]   plt.ylabel("R^2 score of time prediction")
[263]   plt.axvline(x=64, ymin=0, ymax=0.9139674817757812, linestyle = '--',color='red')
[264]   plt.show()


[265]   X, y = X, y1
[266]
[267]   neigh = NearestNeighbors(n_neighbors=2)
[268]   nbrs = neigh.fit(X)
[269]   distances, indices = nbrs.kneighbors(X)
[270]
[271]   distances = np.sort(distances, axis=0)
[272]   distances = distances[:,1]
[273]   plt.plot(distances)
[274]
[275]   m = DBSCAN(eps=0.3, min_samples=5)
[276]   m.fit(X)
[277]
[278]   clusters = m.labels_
[279]
[280]   colors = ['royalblue', 'maroon', 'forestgreen', 'mediumorchid', 'tan', 'deeppink', 'olive',
    'goldenrod', 'lightcyan', 'navy']
[281]   vectorizer = np.vectorize(lambda x: colors[x % len(colors)])
[282]   plt.scatter(X[:,0], X[:,1], c=vectorizer(clusters))

[283]   sns.set()
[284]
[285]   X, y = X, y2
[286]
[287]   neigh = NearestNeighbors(n_neighbors=2)
[288]   nbrs = neigh.fit(X)
[289]   distances, indices = nbrs.kneighbors(X)
[290]
[291]   distances = np.sort(distances, axis=0)
[292]   distances = distances[:,1]
```

```
[293]   plt.plot(distances)
[294]
[295]   m = DBSCAN(eps=0.3, min_samples=5)
[296]   m.fit(X)
[297]
[298]   clusters = m.labels_
[299]
[300]   colors = ['royalblue', 'maroon', 'forestgreen', 'mediumorchid', 'tan', 'deeppink', 'olive',
    'goldenrod', 'lightcyan', 'navy']
[301]   vectorizer = np.vectorize(lambda x: colors[x % len(colors)])
[302]   plt.scatter(X[:,0], X[:,1], c=vectorizer(clusters))


[303]   eps = [0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.50, 0.55, 0.60, 0.65, 0.70, 0.75,
    0.80, 0.85, 0.90, 0.95, 1.00, 1.05, 1.10, 1.15, 1.20, 1.25, 1.30, 1.35, 1.40, 1.45, 1.50, 1.55, 1.60,
    1.65, 1.70, 1.75, 1.80, 1.85, 1.90, 1.95, 2.00]
[304]   eps_C_val = []
[305]   for i in range(len(eps)):
[306]       regr1 = SVR(kernel='rbf', C=1024, epsilon=eps[i])
[307]
[308]       # LeaveOneOut will be used to extract
[309]       LOO = LeaveOneOut()
[310]
[311]       # This is where we create a list containing the true y1 values and the values the model
    predicts
[312]       # These will be used to run the r^2 validation on
[313]       y1_true, y1_pred = list(), list()
[314]       for train_ix, test_ix in LOO.split(X):
[315]           X_train, X_test = X[train_ix, :], X[test_ix, :]
[316]           y1_train, y1_test = y1[train_ix], y1[test_ix]
[317]           # We use the model that was fitted above
[318]           regr1.fit(X_train, y1_train)
[319]           # evaluate model
[320]           yhat = regr1.predict(X_test)
[321]           # store
[322]           y1_true.append(y1_test[0])
[323]           y1_pred.append(yhat[0])
[324]           # calculate accuracy
[325]
[326]       y1_r2 = r2_score(y1_true, y1_pred)
[327]
[328]       print("Error in percentage correct prediction (R^2) at " + str(eps[i]))
[329]       print(y1_r2)
[330]       eps_C_val.append(y1_r2)
[331]
[332]
[333]   eps2_C_val = []
```

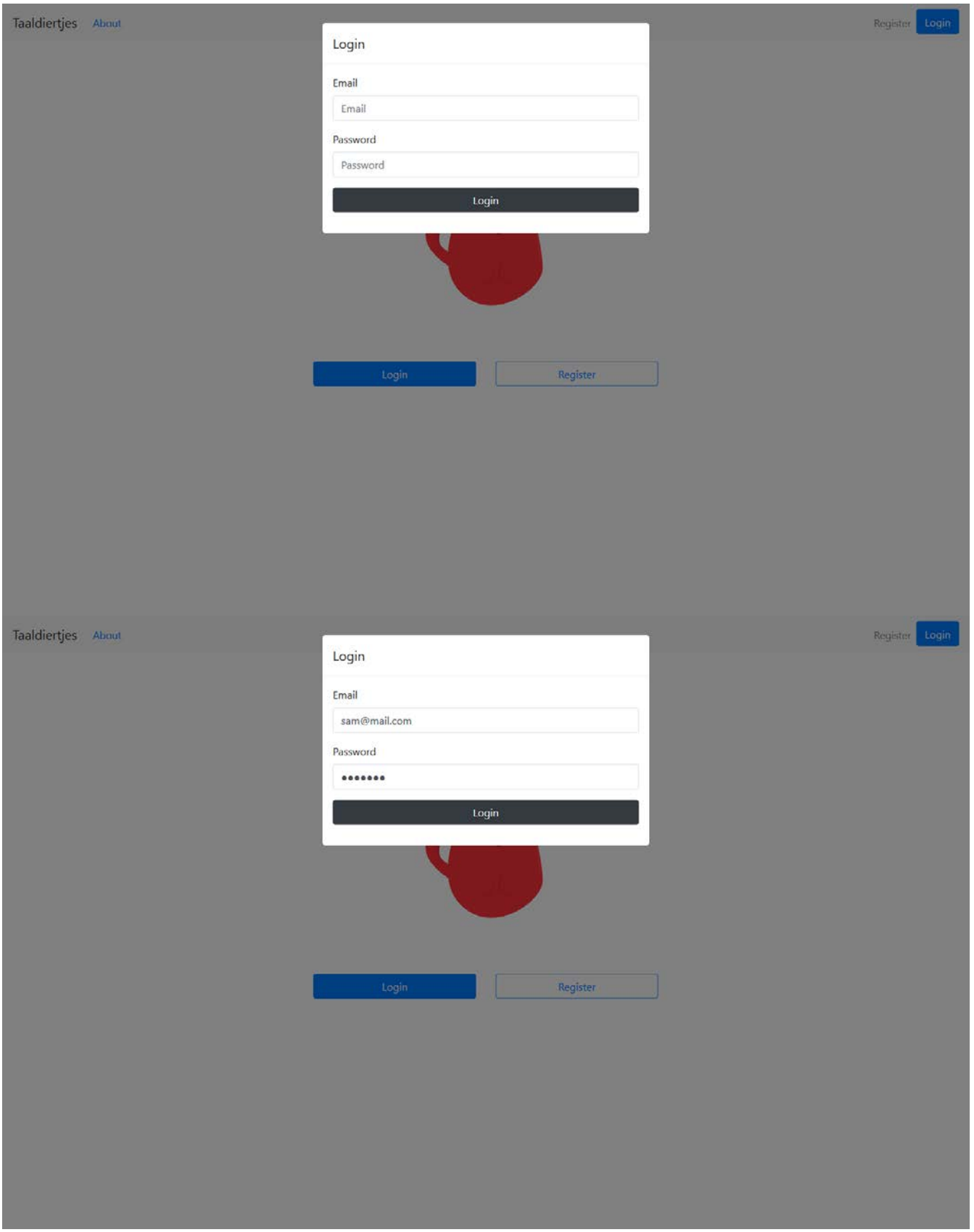
```
[334]
[335]   for i in range(len(eps)):
[336]       regr2 = SVR(kernel='rbf', C=64 ,epsilon=eps[i])
[337]
[338]       # LeaveOneOut will be used to extract
[339]       LOO = LeaveOneOut()
[340]
[341]       # This is where we create a list containing the true y1 values and the values the model
    predicts
[342]       # These will be used to run the r^2 validation on
[343]       y2_true, y2_pred = list(), list()
[344]       for train_ix, test_ix in LOO.split(X):
[345]           X_train, X_test = X[train_ix, :], X[test_ix, :]
[346]           y2_train, y2_test = y2[train_ix], y2[test_ix]
[347]           # We use the model that was fitted above
[348]           regr2.fit(X_train, y2_train)
[349]           # evaluate model
[350]           yhat = regr2.predict(X_test)
[351]           # store
[352]           y2_true.append(y2_test[0])
[353]           y2_pred.append(yhat[0])
[354]           # calculate accuracy
[355]
[356]       y1_r2 = r2_score(y2_true, y2_pred)
[357]
[358]       print("Error in percentage correct prediction (R^2) at " + str(eps[i]))
[359]       print(y1_r2)
[360]   eps2_C_val.append(y1_r2)
```

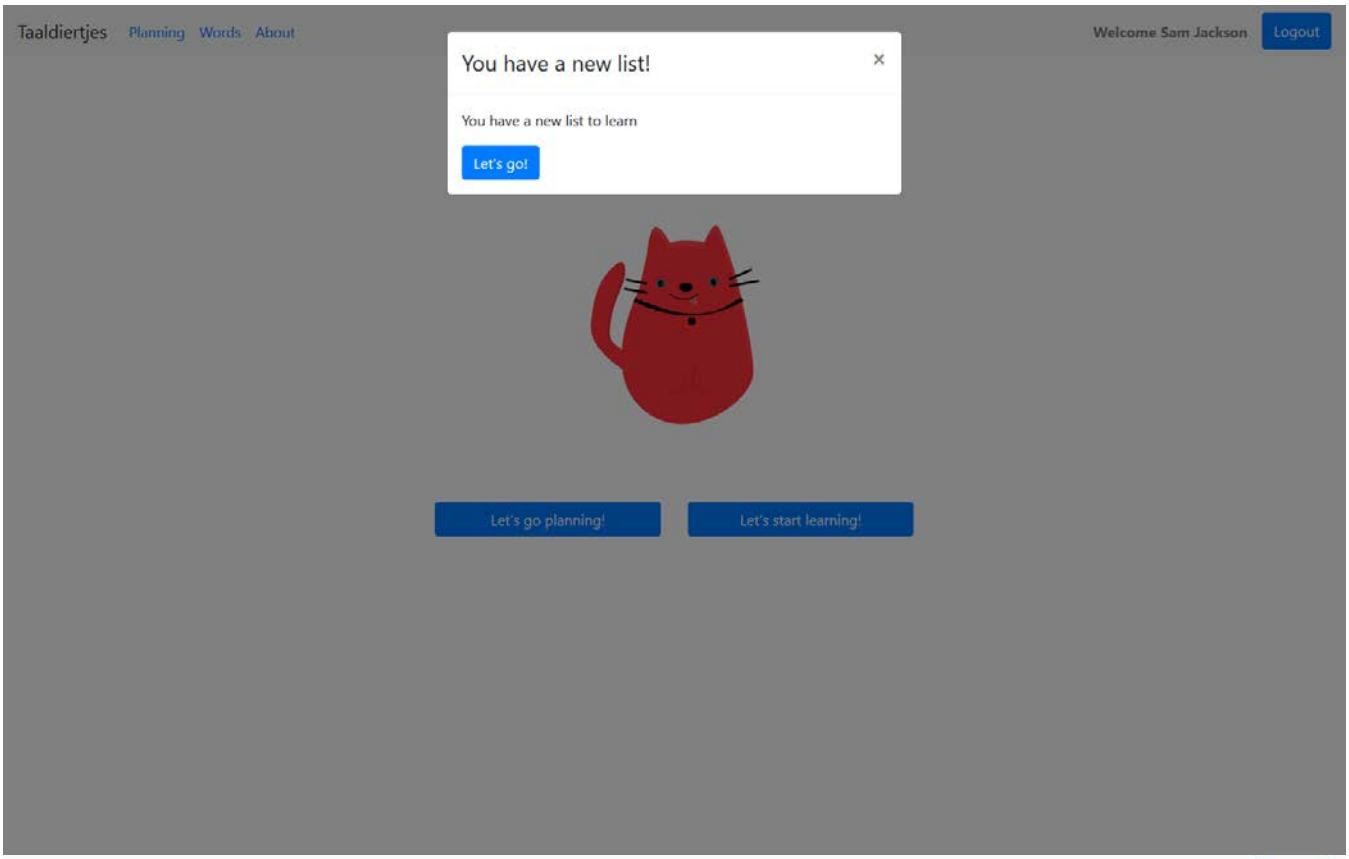
Appendix II: Data file

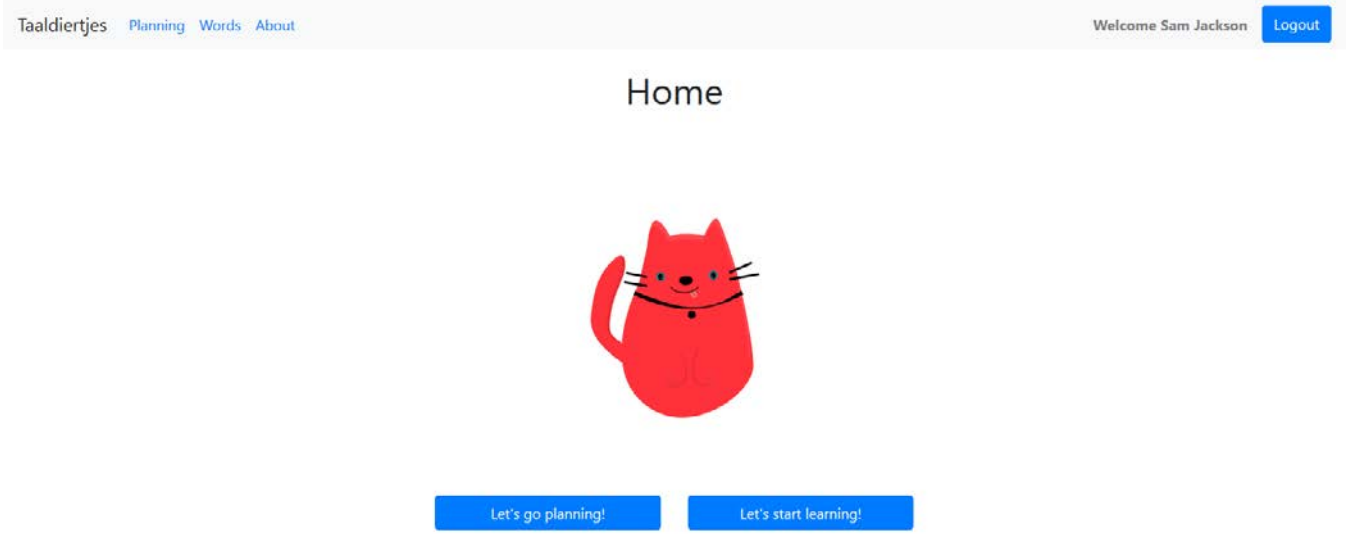
Level	Dutch	English	Amount_of_Word	Previous_Time	Times_Practiced	Percentage_Correct_	O1_Correct	O2_Time
1	1	0	20	16	1	35	50	15
1	1	0	20	14	2	50	63	14
1	1	0	20	14	3	65	68	14
1	1	0	20	14	3	69	77	14
1	1	0	20	14	4	75	78	13
1	1	0	20	11	5	77	80	11
1	1	0	20	10	6	78	81	10
1	1	0	25	18	1	33	55	17
1	1	0	25	14	4	67	75	12
1	1	0	30	20	1	29	53	19
1	1	0	30	16	4	65	72	13
1	1	0	40	25	1	31	56	23
1	1	0	40	22	3	65	71	18
1	0	1	20	18	1	36	54	17
1	0	1	20	13	4	71	77	12
1	0	1	20	10	6	82	84	8
1	0	1	25	23	3	68	76	18
1	0	1	30	23	4	64	72	19
1	0	1	40	26	1	28	51	24
1	0	1	40	22	3	64	73	20
2	1	0	20	9	1	40	67	8
2	1	0	20	7	2	68	80	8
2	1	0	20	5	3	79	84	4
2	1	0	20	4	3	82	87	3
2	1	0	20	4	4	84	89	2
2	1	0	20	3	5	88	80	2
2	1	0	20	2	6	91	93	2
2	1	0	25	11	1	42	67	10
2	1	0	25	5	4	83	87	3
2	1	0	30	12	1	39	64	10
2	1	0	30	6	4	80	84	4
2	1	0	40	14	1	38	65	13
2	1	0	40	8	3	78	82	6
1	0	1	20	17	1	37	55	16
1	0	1	20	13	4	71	77	10
1	0	1	20	10	6	82	84	6
1	0	1	25	22	3	68	77	17
1	0	1	30	22	4	70	75	19
1	0	1	40	26	1	33	53	24
1	0	1	40	23	3	69	75	21
1	1	0	20	10	1	25	40	8
1	1	0	20	14	5	75	80	12
1	1	0	20	16	6	85	90	14
1	1	0	20	16	1	20	35	15
1	1	0	20	18	1	25	40	14
1	1	0	40	24	3	45	60	21
1	1	0	40	20	1	15	25	19
1	1	0	30	18	6	75	75	15
1	1	0	30	24	6	70	75	21
1	1	0	25	12	2	45	55	11
1	1	0	25	15	3	50	55	13
1	1	0	20	14	3	55	65	12
1	1	0	20	12	2	50	60	10
2	0	1	20	3	5	92	95	2
2	0	1	20	12	2	65	90	8
2	0	1	30	18	2	60	80	13
2	0	1	25	7	6	95	96	4
2	0	1	40	10	3	75	85	9
2	0	1	20	10	1	40	70	8
2	0	1	40	14	1	36	68	13
2	0	1	20	15	1	75	95	10
2	0	1	20	16	1	75	95	10
2	0	1	20	17	1	80	85	12
2	0	1	25	18	1	68	84	15
2	0	1	25	21	2	84	96	12
2	0	1	20	19	2	90	95	15
2	0	1	30	22	1	80	90	15
2	0	1	25	19	1	84	92	12
2	0	1	20	18	2	95	95	12
2	1	0	20	18	2	95	95	10
2	1	0	25	20	1	84	88	14
2	1	0	30	25	3	70	84	15
2	1	0	30	27	3	80	93	13
2	1	0	35	28	3	66	86	20
2	1	0	20	17	2	70	85	10
2	1	0	40	28	4	60	80	20
2	1	0	20	17	1	85	90	10
2	1	0	25	22	3	88	96	12
2	1	0	30	26	2	83	90	16
2	0	1	20	15	1	75	90	10

Appendix III: Images webpage

The webpage is fully functioning, both back-end and front-end and is data is encrypted.


Taaldiertjes
About
Register
Login
Login
Email
Email
Password
Password
Login
Login
Register
Taaldiertjes
About
Register
Login
Login
Email
sam@mail.com
Password
•••••••
Login
Login
Register


Taaldiertjes
Planning
Words
About
Welcome Sam Jackson
Logout
You have a new list!
You have a new list to learn
Let's go!
Let's go planning!
Let's start learning!


Taaldiertjes
Planning
Words
About
Welcome Sam Jackson
Logout
Home
Let's go planning!
Let's start learning!

Taaldiertjes Planning Words About Welcome Sam Jackson Logout

Dierennamen
Engels
Let's practice!

Eten
Engels
Let's practice!

Spelling woorden 13 New
Nederlands
Let's practice!

Taaldiertjes Planning Words About Welcome Sam Jackson Logout

Planning

Volgende

Lists to learn

Dierennamen
Engels - 1 keer geoefend - vorige keer 10 minuten

Eten
Engels - 3 keer geoefend - vorige keer 10 minuten

Spelling woorden 13
Nederlands - 0 keer geoefend

Planing Today

Taaldiertjes Planning Words About Welcome Sam Jackson Logout

Planning

Volgende

Lists to learn

Dierennamen

Engels - 1 keer geoefend - vorige keer 10 minuten

Eten

Engels - 3 keer geoefend - vorige keer 10 minuten

Spelling woorden 13

Nederlands - 0 keer geoefend

Planing Today

Dierennamen - 10 minten

Spelling woorden 13

Nederlands - 20 minten

Taaldiertjes Planning Words About Welcome Sam Jackson Logout

Planning

Weet je zeker dat je 20 minuten doet over 'Spelling woorden 13'? Ik denk dat het iets minder zou kunnen zijn...

Volgende

Lists to learn

Dierennamen

Engels - 1 keer geoefend - vorige keer 10 minuten

Eten

Engels - 3 keer geoefend - vorige keer 10 minuten

Spelling woorden 13

Nederlands - 0 keer geoefend

Planing Today

Dierennamen - 10 minten

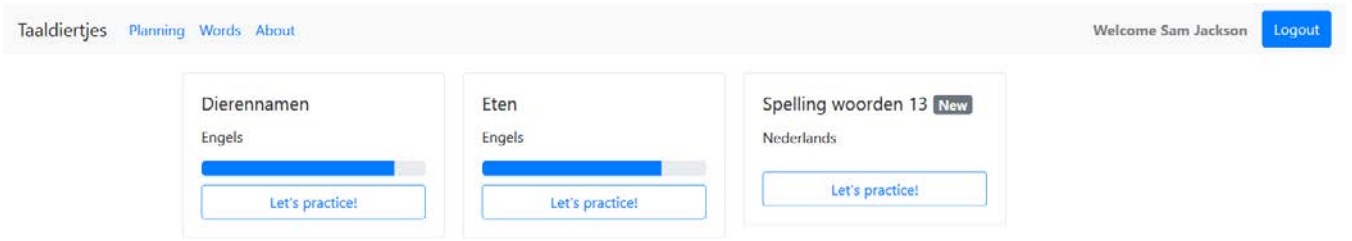
Spelling woorden 13

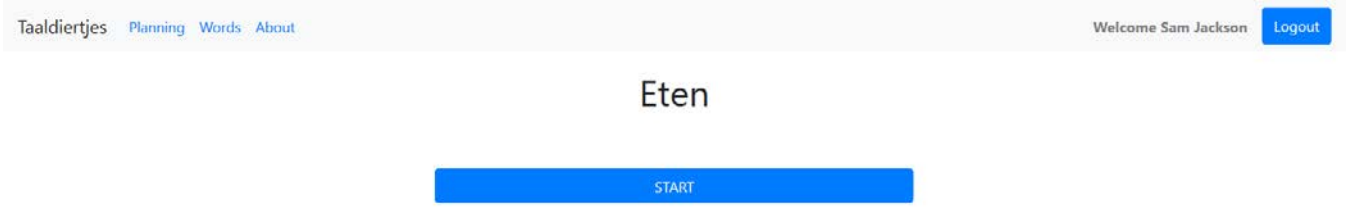
Nederlands - 20 minten

Eten - 10 minten

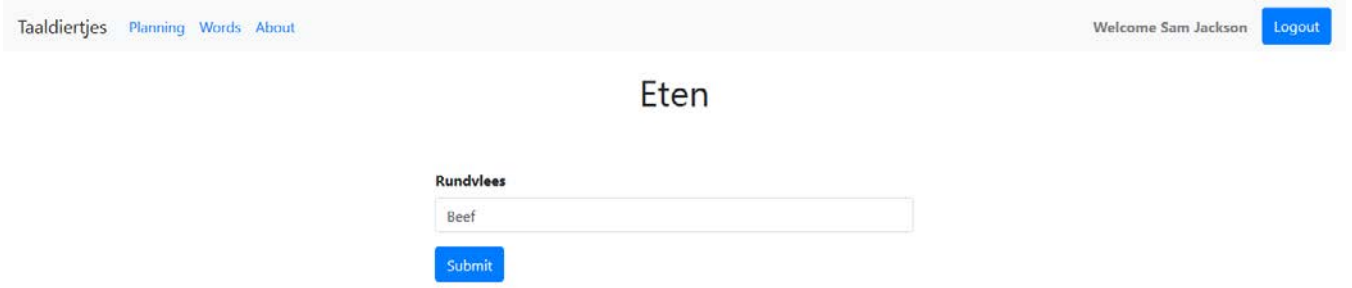
Dierennamen - 10 minten

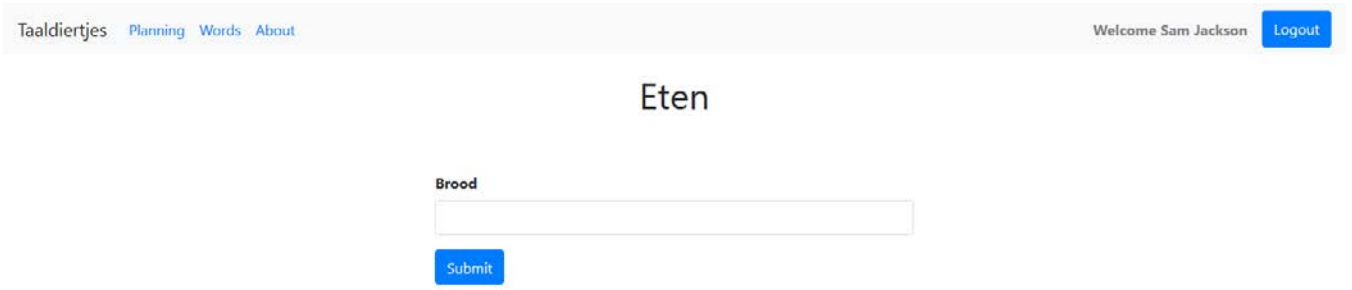
Eten - 10 minten

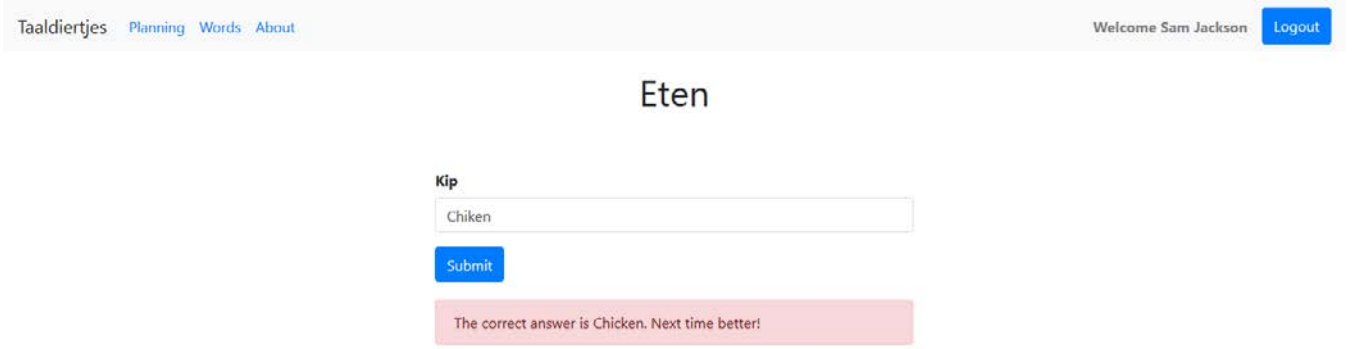

Taaldiertjes
Planning
Words
About
Welcome Sam Jackson
Logout
Dierennamen
Engels
Let's practice!
Eten
Engels
Let's practice!
Spelling woorden 13
New
Nederlands
Let's practice!

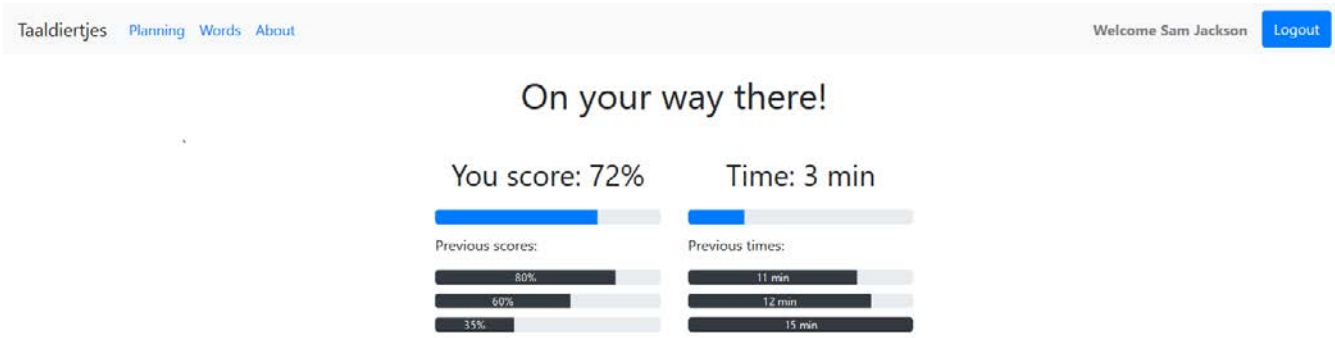

Taaldiertjes
Planning
Words
About
Welcome Sam Jackson
Logout
Eten
START


Taaldiertjes
Planning
Words
About
Welcome Sam Jackson
Logout
Eten
Rundvlees
Submit


Taaldiertjes
Planning
Words
About
Welcome Sam Jackson
Logout
Eten
Rundvlees
Beef
Submit


Taaldiertjes
Planning
Words
About
Welcome Sam Jackson
Logout
Eten
Brood
Submit


Taaldiertjes
Planning
Words
About
Welcome Sam Jackson
Logout
Eten
Kip
Chiken
Submit
The correct answer is Chicken. Next time better!


Taaldiertjes
Planning
Words
About
Welcome Sam Jackson
Logout
On your way there!
You score: 72%
Time: 3 min
Previous scores:
80%
60%
35%
Previous times:
11 min
12 min
15 min


Taaldiertjes
Planning
Words
About
Welcome Sam Jackson
Logout
Eten
Kaas
Cheese
Submit
Correct!

Taaldiertjes About Register Login

Home

Login Register

Appendix IV: Overview Inputs & Outputs