UNIVERSITY OF LIMERICK
OLLSCOIL LUIMNIGH

CS6502 Applied Big Data and Visualization

Predicting User Recall probability in Duolingo using Apache spark

Supervised by: Prof. Sarmad Ali

Written by:

24138789: Anish Kamble

24131504: Kaif Nadaf

24011703: Cian Murphy

24055298: Prajwal Mahanawar

24101583: Nambari Yuva Narasimha Rao

1. Introduction

In this project, we explored how machine learning can be applied to model and predict human memory using large-scale educational data. Specifically, we worked with the Duolingo Spaced Repetition dataset, which contains over 13 million interaction records of language learners attempting vocabulary recall tasks. Each record reflects how likely a user was to remember a given word, along with contextual features such as time since last review, language interface, and correctness history.

We chose this dataset because of its scale, richness, and relevance to real-world learning systems. Predicting word recall can significantly improve personalized learning, helping systems like Duolingo decide what content to show next to optimize retention. The project aligned well with our interests in EdTech and data science, and it offered a chance to work with a real-world problem that combines cognitive modelling with machine learning.

We used Apache Spark on Databricks Community Edition to manage, process, and analyse the large dataset. The project involved end-to-end tasks, including data ingestion and cleaning, exploratory data analysis (EDA), feature engineering, model building, and evaluation using SparkML. We also performed hyperparameter tuning to improve model performance.

Our goal was to predict the p_recall score (a floating-point value between 0 and 1), which represents the system's estimate of a user's likelihood to recall a word correctly at a given time. We approached this as a regression task, and after testing multiple models, achieved a strong R^2 of 0.82 using Random Forest Regression, built on a carefully engineered and cleaned sample of ~1.4 million records.

2. Dataset Overview

We used the Duolingo SLAM dataset (13M+ rows), which tracks how users practice words while learning a new language. Each row represents one user-word interaction. Our goal was to predict p_recall, the likelihood that a user remembers a word at a given moment.

Original Features:

Column Name	**Description**
p_recall	Target variable (probability of recall)
timestamp	Unix time of the interaction
delta	Time (in seconds) since the last exposure to this word
user_id	Anonymized user identifier
Column Name	**Description**
learning_language	Language being learned
ui_language	Language used in the app interface
lexeme_id	Internal ID for the word

lexeme_string	Actual word being studied
history_seen	Number of times the word has been seen historically
history_correct	Number of times the word has been answered correctly historically
session_seen	Number of times the word has been seen in the current session
session_correct	Number of times the word has been answered correctly in the current session

We selected the Duolingo dataset for its size (13M+ rows) and rich user interaction data. It captures both performance and time-based learning patterns, making it ideal for prediction tasks. To speed up development on Databricks CE, we used a stratified sample of ~1.4M rows, ensuring user diversity was preserved.

3. Data Cleaning & Preprocessing

Before any analysis, we undertook rigorous cleaning to ensure the dataset was reliable and ready for machine learning. The original dataset contained over 13 million spaced repetition records from Duolingo users, and while it was mostly well-structured, we encountered a few issues that needed to be addressed.

I. Handling Missing & Invalid Data

We began by checking for nulls, NaNs, and empty string values across all columns. Fortunately, the dataset had no missing values, which allowed us to retain most of the data. However, we did find a small number of duplicate rows (81), which were promptly removed.

Next, we focused on filtering logically inconsistent or invalid entries. Specifically:

- Rows where delta (time since last exposure) was less than or equal to 0 were removed.
- Records with history_seen or session_seen equal to 0 were dropped, as accuracy could not be calculated.
- We also filtered rows where history_correct > history_seen or session_correct > session_seen, which are impossible scenarios.

After these steps, the cleaned dataset contained over 12.8 million rows.

To speed up processing and fit within computational limits, we later sampled approximately 1.4 million rows using stratified sampling by user ID. We've mentioned this clearly in our notebook as well.

II. Feature Engineering

We engineered two new features to better capture learner behavior:

- **history_accuracy** = history_correct / history_seen
- **session_accuracy** = session_correct / session_seen

These features gave us a normalized view of a user's long-term and short-term performance, which helped during EDA and correlation analysis.

Later, we dropped the original count-based columns (like history_seen, history_correct, etc.) once we confirmed that their information was already captured in the accuracy metrics.

4. Exploratory Data Analysis (EDA)

With the cleaned data in place, we performed a series of analyses to better understand the patterns, trends, and possible predictors of word recall (p_recall). Our EDA made use of both SparkSQL and Databricks visualizations, which helped uncover valuable insights.

Summary Statistics

We first calculated key descriptive statistics for the main numeric features:

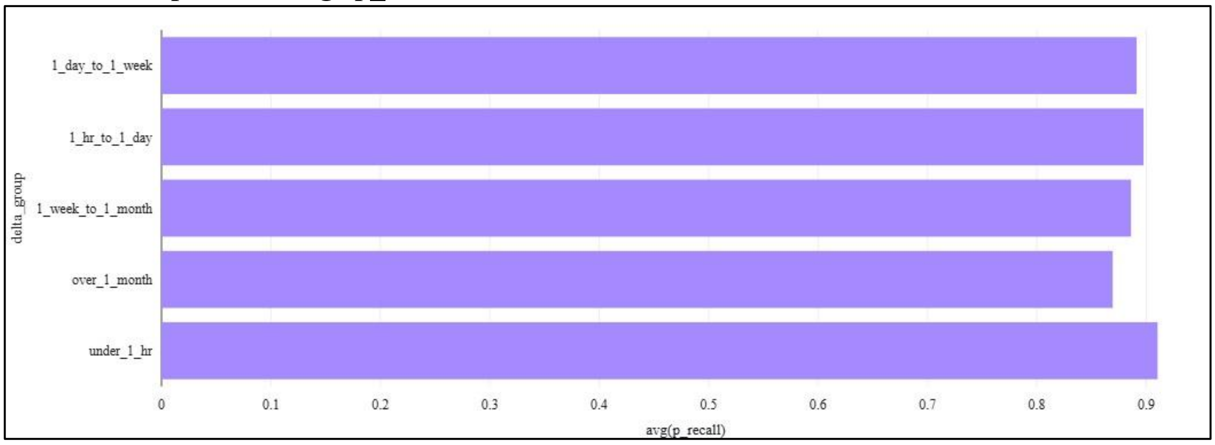
- p_recall had a mean of **~0.89** with low variance, suggesting that most predictions are close to full recall.
- delta (time since last review) ranged from a few seconds to over a year, confirming the need to treat it carefully in modeling.
- session_correct had an average around **1.64**, showing most interactions involved just one or two questions.

These metrics helped us get an overall sense of the dataset and were also useful later when standardizing numerical inputs.

Visual Patterns & Relationships

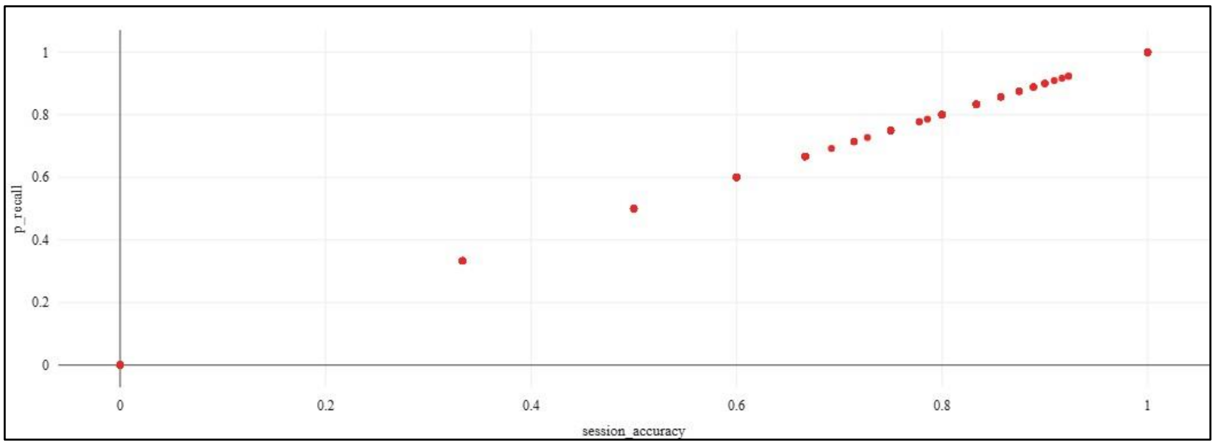
We used various grouping and visualization techniques to extract insights:

1. Delta Groups vs. Average p_recall



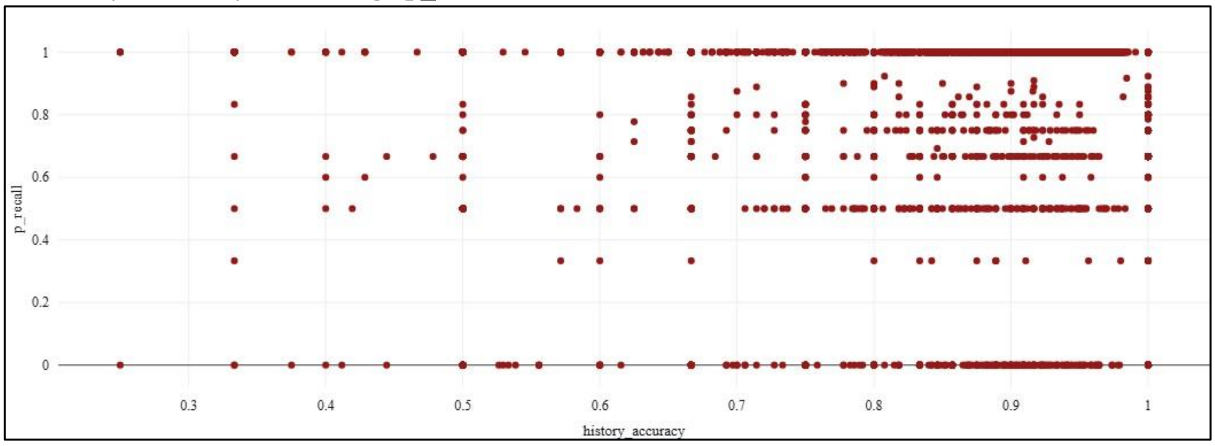
We binned delta into time groups (under 1 hour, 1 day, 1 week, etc.). We found a clear downward trend, the longer the delay between exposures, the lower the recall. This aligned with the psychological idea of the forgetting curve.

2. Session Correct vs. Average p_recall



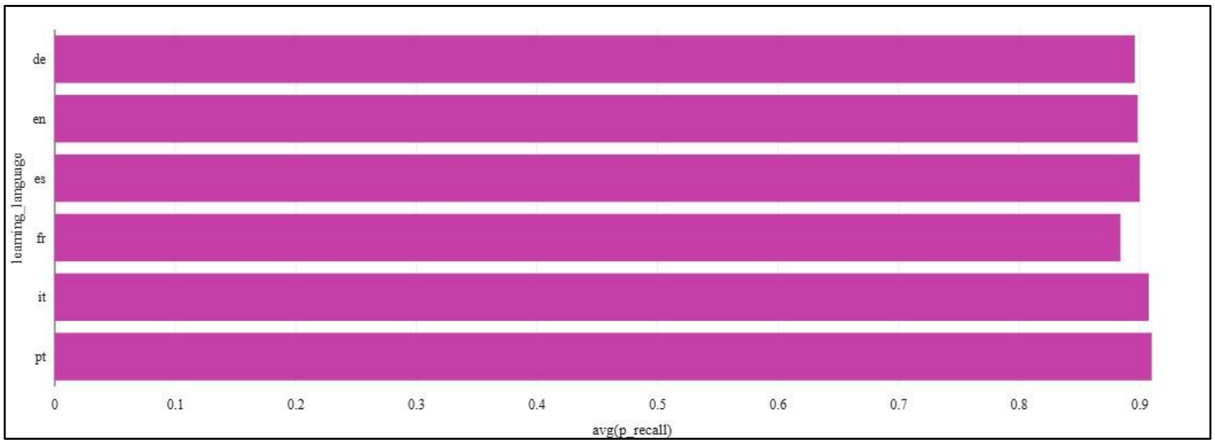
We observed that learners who got more answers correct in a session also had higher recall values. This suggested that engagement and performance in the current session were strong indicators of memory.

3. History Accuracy vs. Average p_recall



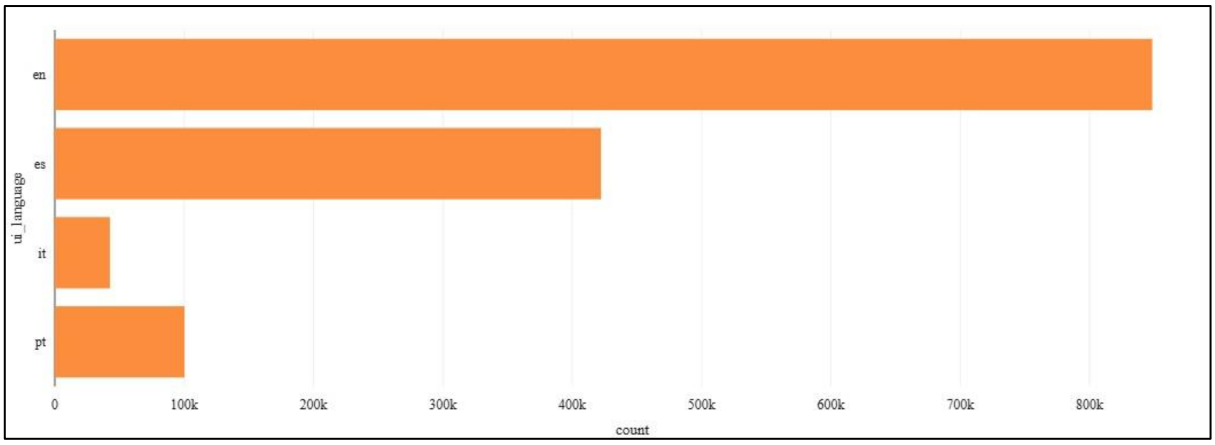
Plotting this revealed a positive relationship, but with diminishing returns. Users with high long-term accuracy performed better, but the gain levelled off: possibly due to saturation or repeating familiar words.

4. Language Influence

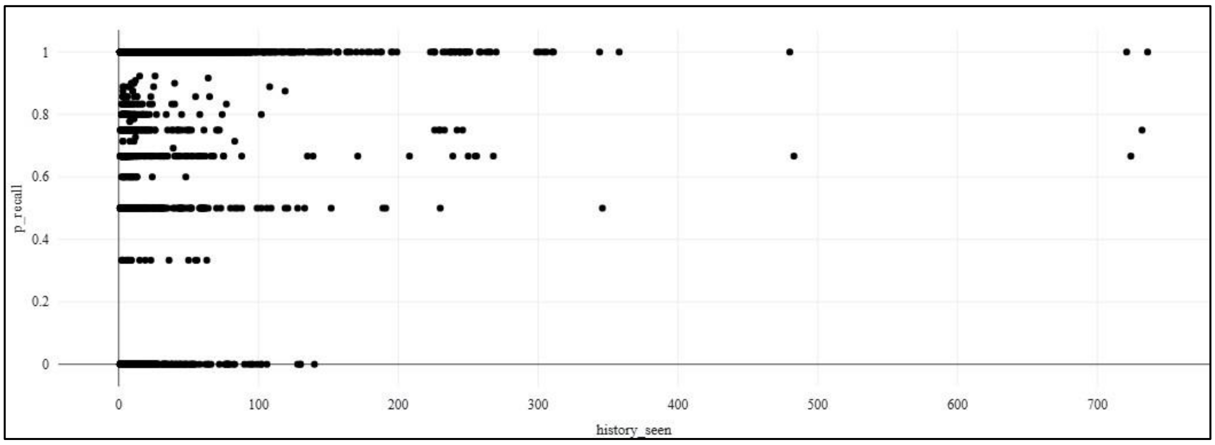


Different learning languages had slightly different average recall scores. For example, users learning English from Spanish showed slightly higher recall rates compared to other pairs.

Some other visualizations:



Unsurprisingly, English dominates by a significant margin, suggesting that most users in this sample preferred English as their primary interface language. This could reflect Duolingo's global user base, where English remains a widely adopted default language.



This scatter plot shows the relationship between history_seen (how many times a user has seen a word before) and p_recall .While the points are quite scattered, we can observe a loose upward trend, as users are exposed to a word more often, their recall probability tends to increase.

5. Feature Engineering and Analysis

Based on our EDA insights, we moved into preparing and refining features for machine learning. This phase involved selecting relevant variables, addressing multicollinearity, transforming categorical values, and scaling numerical features.

Feature Correlation Analysis

We first computed the Pearson correlation matrix for all numeric variables. This helped identify redundancy and potential data leakage.

Key findings:

1. session_accuracy had an almost perfect correlation with the target p_recall (≈ 1.0), meaning it was essentially a duplicate of the label. We dropped it to avoid leakage.
2. history_correct, history_seen, and session_seen were all strongly correlated with other variables, particularly history_accuracy and session_correct, so we retained only the most meaningful representatives.
3. delta, history_accuracy, and session_correct showed moderate correlation with p_recall, validating them as potentially useful predictors.

Feature Selection

Features that were selected for our final modelling:

- **Numerical:**
    - delta: Time since last seen.
    - history_accuracy: Ratio of correct to total answers historically.
    - session_correct: Number of correct answers in the current session.
- **Categorical:**
    - learning_language o ui_language

These were transformed using the following methods:

- StandardScaler was applied to the numerical features to normalize their range.
- StringIndexer + OneHotEncoder were used to convert categorical language fields into binary vectors suitable for ML models.
- All features were combined into a final features vector using VectorAssembler.

Sampling Strategy

Since the original dataset had over 13 million rows, running full-scale modelling was computationally expensive on Databricks Community Edition. To manage this:

- We used **stratified sampling** based on user_id to retain a diverse set of learners.
- This brought us to **~1.4 million rows**, allowing us to stay within project requirements (1M+ rows) while ensuring the model trained efficiently.

We cached this final sampled DataFrame (df_sample) for faster downstream processing.

Machine Learning Implementation & Evaluation

Objective

Our core objective was to **predict the probability of a user recalling a word (p_recall)** based on session and historical features using regression algorithms in SparkML.

Model Selection

We chose two regression models for this task:

1. **Linear Regression** – as a baseline model due to its simplicity and interpretability.
2. **Random Forest Regressor** – for its ability to capture complex non-linear relationships in the data and handle high-dimensional inputs effectively.

Model Training

After feature engineering, we split our dataset into:

- **Training set** – 80%
- **Testing set** – 20%

We trained both models using the transformed feature set.

Values/Metric	**Linear Regression**	**Random Forest**
RMSE	0.2544	0.1132
MAE	0.1766	0.0641
R^2 Score	0.1024	0.8222

Our results clearly showed that the linear regression model struggled to capture the underlying patterns in the data, which wasn't too surprising. Human memory and behavior aren't always straightforward or linear; they often follow more complex, non-obvious paths. On the other hand, the random forest model handled these complexities much better. It was able to pick up on subtle, non-linear relationships in the features and gave us a much stronger, more reliable fit. This reinforced our belief that for behavioral prediction tasks like this, tree-based models are often more suitable than simple linear ones.

Feature Importance:

1. **session_correct** – most predictive of p_recall.
2. **history_accuracy** – contributed moderately.
3. **delta** – time between exposures had a smaller, but noticeable impact.

Hyperparameter Tuning (Bonus)

To further optimize performance, we performed **Grid Search Cross-Validation (3-fold)** on the Random Forest:

- numTrees: [20, 50]
- maxDepth: [5, 10]
- minInstancesPerNode: [1, 5]

The cross-validation job was initiated but couldn't complete due to time and resource constraints on Databricks CE. Despite that, our manually tuned model already showed strong performance.

6. Challenges, Reflections & Future Work

Big Data, Small Engine

Working with 13M+ rows on Databricks CE meant long runtimes. We used a stratified 1.4M-row sample to strike a balance between speed and data diversity.

Cross-Validation Limits

Our initial grid search with cross-validation failed due to resource constraints. We adjusted by limiting parameter combinations to keep things stable.

Key Insights

More Correct Answers = Higher Recall

Users who performed better in a session had noticeably higher p_recall. Session-level success turned out to be one of the best predictors.

History Accuracy Helped, But Less

Users with good long-term performance also recalled better, but the impact was less sharp than recent session results.

Time Since Review Matters

As delta (time since last review) increased, p_recall dropped slightly confirming the effect of memory decay, just as expected in spaced repetition systems.

Future Work

Time-Aware Models: We'd like to explore models that handle sequences and time more deeply, like LSTMs or time series techniques.

Word-Level Difficulty: Incorporating word-specific difficulty could improve personalization.

Better Tuning: With more resources, we would expand hyperparameter tuning and try advanced ensemble models.

Final Remarks

This project gave us valuable hands-on experience working with large-scale data using Apache Spark. From cleaning and transforming millions of rows to building and tuning machine learning models, we got a strong understanding of the challenges and potential of big data analytics. While not everything went perfectly, especially with compute constraints and tuning, we were able to extract meaningful insights and build a model that performed well. Overall, it was a rewarding learning experience that pushed us to apply both technical and analytical skills in a real-world context.

NOTE:

Use of GenAi:

- We have used Ai tools only at few places as we got some difficulties and also we tried to explore dataset, used ChatGpt to understand how to do sample ~1M Rows(Sampling ~1M Rows for Efficiency)
- Used ChatGpt for allocating divisions
- Model Tuning with Cross-Validation and to understand ParamGridBuilder and CrossValidator.