Introduction and Executive Summary

Machine learning (ML) plays a pivotal role in the development of pedagogy not only in the classroom, but also outside with digital learning tools. Notably, Khan Academy, the client, is working to develop its platform to better adjust its learning material suggestions for students aged 8-17. To achieve this, they turned to the industry-leading Artificial Intelligence (AI) Lab Inc. for support in processing, training, and predicting outcomes from their assessment data, with a focus on addressing two key questions.

1. How can Khan Academy leverage student assessment insights to predict students' performance on checkpoint diagnostic questions?
2. How can such predictions help Khan Academy better suggest learning material to maximize students' efficiency and growth potential while learning new subjects?

The answer to Question 2 will be explored in the **Wrap-Up and Final Thoughts**. Meanwhile, using Khan Academy's data from 542 students who each solved a subset of the platform's impressive 1,774 mathematics and science diagnostic questions, the dedicated AI team worked diligently to develop three ML models in its mission to solve Question 1.

1. K-Nearest Neighbors (KNN)

   The KNN model is a simple approach to predict missing student diagnostic question. The method intends to take a student's diagnostic test information from the questions they have completed, and predicts their score on questions they have not attempted yet. This is done based on finding the k-closest neighboring students in the dataset, averaging their pass/fail scores on the missing questions, and rounding the result up to indicate the prediction is a correct answer, or down to indicate the prediction is an incorrect answer. This model reported a test accuracy of up to **68.42%**.

2. Item Response Theory (IRT)

   The IRT model is a common choice for predicting response correctness. The problem defines an ability vector $\boldsymbol{\theta}$ that allows the model to factor in student ability, where each component is indicative of a weight of a student's problem solving ability. The problem also defines a difficulty vector $\boldsymbol{\beta}$ that similarly assigns a difficulty to each question. The solution involves an alternating gradient descent to determine the "optimized" values of these parameters, which are then used in the evaluation step to make predictions on the Khan Academy train dataset with a parameter-dependent logistic model. This model reported a test accuracy of up to **70.28%**.

3. Neural Network (NN)

   The NN model involves taking students' performance and predicting their performance on a problem they have not attempted yet by first running their existing pass/fail scores through a "forward pass" of an autoencoder network with two linear function layers that encode and reconstruct the original input respectively, each using a sigmoid activation function. Then, the model involves a "backpropogation" of the loss function and the two layers that takes the partial derivatives of each with respect to the loss to holistically optimize all the weight $\mathbf{W}$ and bias $\mathbf{b}$ vectors containing components of all the weights and biases applied to each student's input vector. Like the IRT model, the optimized parameters are then used to form predictions for the students' performance on diagnostic questions they have yet to attempt. This model reported a test accuracy of up to **67.99%**.

To extend the initial models' base capabilities, the team developed a plan to build on the KNN model, experimented with it, and decided to pivot to a different idea due to some observed limitations that hindered the ability for the model to improve. In the following sections, each attempt will be covered step by step, along with the rationale behind the pivot.

Attempt 1 (KNN Student Metadata Extension)

Procedure

In order to augment the KNN model, the team requested further student demographic data from Khan Academy's archive to provide the model with more insights to base its predictions on. Khan Academy responded with student metadata with the following column schema.

1. user id: ID of the student who answered the question (starts from 0).
2. gender: Gender of the student, when available. 1 indicates a female, 2 indicates a male, and 0 indicates unspecified.
3. data of birth: Birth date of the student, when available.
4. premium pupil: Student's eligibility for free school meals or pupil premium due to being financially disadvantaged, when available.

Unfortunately due to inconsistencies in the date of birth data such as a big age gap and incorrect dates of birth after the present-day outside the range of 8-17 years old [1], the team was restricted to gender and premium_pupil data.

To leverage this data, the team elected to augment the initial 542 x 1774 *train_data* (rows representing each student and columns representing each question) matrix by adding the columns aligning with the respective row for each student. The team applied a pre-processing procedure as follows.

1. gender

   The gender data indicates a 1 if the student with a particular user ID is a female, a 2 if the student is a male, and a 0 if unspecified. Hence, the team needed to match the data format with the rest of the entries in the table, which only uses 0s (incorrect solution), 1s (correct solution), and NaN values (needs predicting, whether it the entry is held back for *test_data* or the question is unanswered by the student).

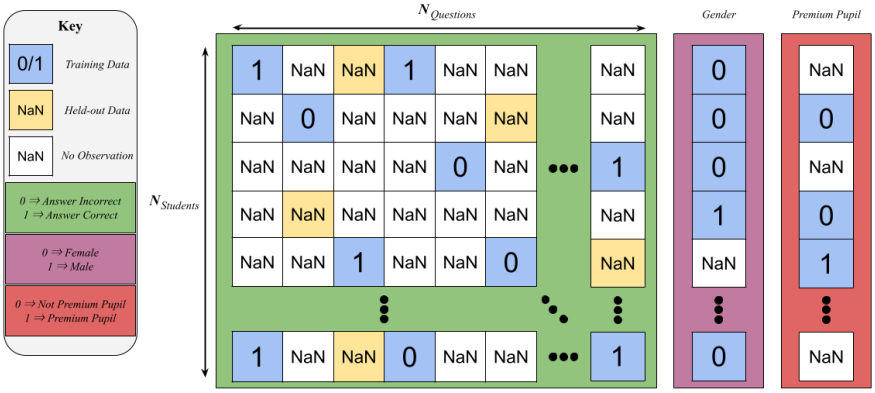
   Therefore, to pre-process the data, the team mapped the gender column values such that all 0s (previously unspecified) are mapped to NaN values.

2. premium_pupil

   The premium_pupil data indicates a 0 if the student is not eligible for free school meals or pupil premium due to being financially diadvantaged, 1 if they are, or NaN if unspecified. This matches the structure of the data in the existing question information, so the team left it untouched.

The resulting matrix (when converted to 542 x [1774 + 2] = 542 x 1776) sparse matrix) resembles the following diagram. The KNN model then forms predictions on rows 0 to 1773 (the questions) with the added information from the two new columns in the same manner as the base model previously explained.

Sparse Training Matrix with Added Student Metadata



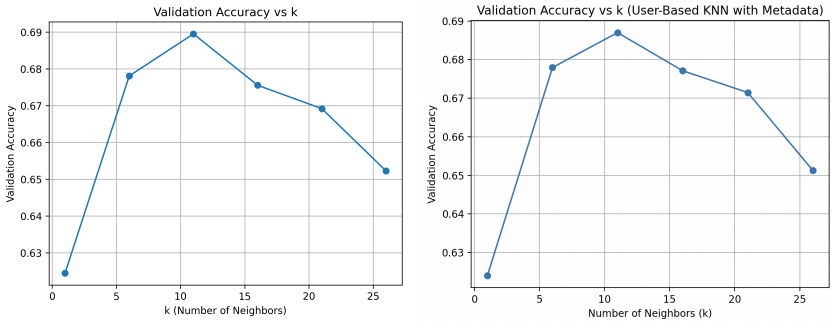
[1] The team attempted to clean this data by only including students whose birth year is between 2006 and 2017 (aged 7-18) to account for a possible Khan Academy database error, however this resulted in a reduction of eligible students (for both train and test) from 542 to 50. The lack of data resulted in greater sensitivity to noise, and thus, overfitting. Even with regularization, the team was not able to extract any accuracy improvements from the base model's **68.42%** accuracy

Intended Improvement Metric vs. True Effect

The intended improvement with the augmented student metadata pre-processing change is a decrease in underfitting. This hypothesis is because with added unique information, the model is able to better capture patterns and new relationships in the data. Hence, the model can calculate neighboring distances with an improved accuracy by factoring in more features that could be indicative of deeper relations in the data.

The true effect, after carrying out the experiment, is a 0.15% drop in test accuracy from **68.42%** pre-augmentation to **68.27%** post-augmentation, where the slight difference can likely be attributed to variability. The optimal k, or k^* remained at $k^* = 11$. This implies that the team's conclusion is that the KNN student metadata extension introduces a negligible change, if any. The details of the team's findings are as follows.

Original Findings (Pictured Left) vs. Improvement Attempt 1 Findings (Pictured Right)




```
Best k for User–Based KNN: 11
Validation Accuracy with k=11: 0.6827547276319503
Test Accuracy for User–Based KNN with k=11: 0.6827547276319503
```

Limitations

1. The Curse of Dimensionality (CoD) is an important consideration the team made when adding new features. The CoD, as proven by Professor Rahul G. Krishnan of the University of Toronto[2], says that "as dimension increases, so too do the number of irrelevant dimensions that nearest neighbor will compute distances based on." Hence, since the team increased the number of features by two, resulting in 1776 features, the number of dimensions further approaches infinity. This means that the points become increasingly equidistant with less value coming from each added feature.

2. It is possible that, along with the CoD, the specific information given by gender and premium_pupil metadata lacks discriminative power. In other words, the added columns do not provide relevant information that correlates to any underlying patterns in the data. This, in turn, makes them not worth including because the data behaves simply as added noise, thereby not improving the overall accuracy of the model.

3. With a high number of features, the computational efficiency grows linearly in accordance. In tandem with the Curse of Dimensionality and the lack of discriminative power of the added metadata, it makes the linear increase in computational cost and exponential increase in memory usage lack any substantial return in underfit reduction, which was the goal.

Pivot Intuition

After observing the underwhelming performance of Attempt 1, the team re-evaluated its approach. In particular, Khan Academy's brief indicates a request to improve performance. The team capitalized on the CoD with a new idea and goal; apply Principle Component Analysis (PCA) to reduce the number of features (diagnostic questions) in the original dataset down from 1773 to a lower dimension (discovered optimally through a form of trial and error). This will improve computational efficiency linearly, and even better, improve memory usage exponentially by reducing features so that there is less irrelevant information being used.

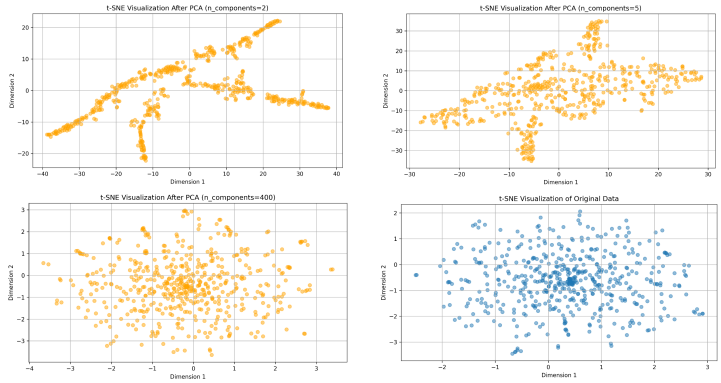
Attempt 2 (KNN *train_matrix* PCA Application)

Procedure

To improve computational efficiency, the team decided to counteract the Curse of Dimensionality through use of Principal Component Analysis (PCA). The training matrix for the KNN model started off as 542 (rows) by 1774 (columns), and had its columns reduced using the built-in PCA function from sklearn. This dimensionality reduction was internally applied through this high-level process:

1. Input Matrix: The original dataset (train_matrix) was provided as input to the PCA function.
2. Covariance Matrix: PCA evaluates the covariance matrix, wherein the library intends to find the most valued c components (where c is the reduced number of features) to capture relationships between the features.
3. Eigenvalues and Eigenvectors: Using the covariance matrix, PCA then goes on to mathematically find the eigenvalues and eigenvectors of the matrix. Each eigenvalue represents the variance explained by its associated eigenvector (or principal component).
4. Selecting Top Components: The library selects the largest c eigenvalues and projects the data onto the new desired subspace with the associated eigenvectors as a linear combination, as they capture the majority of the variance in the dataset.
5. Using the Projection: The input matrix, having now been projected onto a new subspace has a reduced dimensionality of 542 $rows \times ccolumns$.

To illustrate the effects of varying the number of components, some scatterplot (t-distributed stochastic neighbor embedding) visualizations[1] were generated for $c = 2$, $c = 5$, and $c = 400$, as well as for the original data (with no PCA applied). These projected visualizations demonstrate how increasing the number of retained components affects the structure of the data in the low-dimensional embedding:

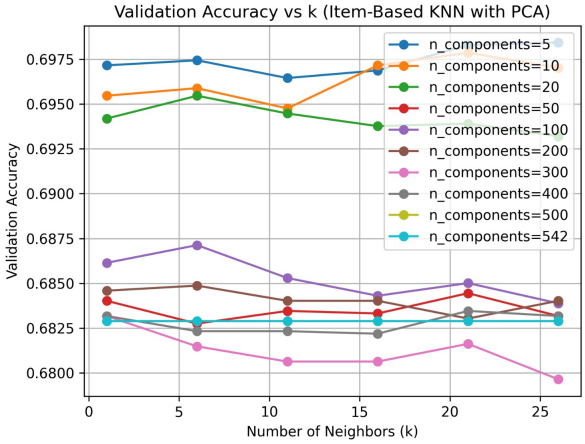


- $c = 2$: Demonstrates a massive reduction in dimensionality with most meaningful patterns lost.
- $c = 5$: Demonstrates a case where a substantial reduction in dimensionality has still discarded too much variance and lost meaningful patterns in the data.
- $c = 400$: This dimensionality balances variance retention and computational efficiency, resulting in clear clusters and patterns while still simplifying the original problem.
- $c = OriginalData$: While this approach ensures 100% variance retention, it is computationally expensive and has likely overfitted to noise.

Intended Improvement Metric vs. True Effect

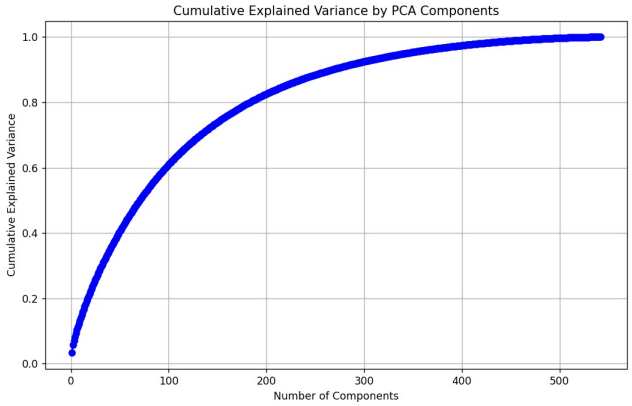
The team experimented with running the KNN model on the post-PCA-projected set of reduced components $C = [5, 10, 20, 50, 100, 200, 300, 400, 500, 542]$ for one $c \in C$ at a time, and evaluated the resulting accuracies on the same set of k-values previously used, $K = [1, 6, 11, 16, 21, 26]$) for one $k \in K$ at a time.

Evaluating KNN Accuracies with Different PCA-Projected Dimensionalities



Evidently, when $k = 26$ and $c = 5$, the resulting accuracy is the greatest of all the combinations at **69.88%**, a 1.42% increase from the original pre-augmented peak ($k = 11$) user-based KNN accuracy of **68.42%**. However, selecting this for further experimentation is naive since the variance captured and explained by the post-PCA dataset is negligible (5%) as can be seen in the graph that follows. Hence, there is a lack of balance between dimensionality reduction and the preservation of explained variance post-PCA.

Retained Variance with Different PCA-Projected Dimensionalities



To solve this dilemma, the team selected $c = 400$ for the remaining tests as a fair balance. This is backed by a Stack Overflow heuristic where the suggestion is a lower limit of 95% retained variance, where $c = 400$ saves roughly 97% variance.

As previously mentioned, the intended improvement metrics lie in comparing the model's computational efficiency and memory usage by combating the effects of the CoD, which are the metrics of interest.

Upon carrying out the experiment to test the hypothesis, the team controlled the number of k-values to be confined to set $K = [1, 6, 11, 16, 21, 26]$[2].

The results are staggering. Firstly, upon exporting the resulting PCA-projected *train_matrix* with $c = 400$ components into a new file and comparing the sizes side-by-side with the original, the original file holds **7,512 KB** of data, whereas the post-PCA matrix holds **5,544 KB**, a 26.2% decrease in memory allocation.

Exported *train_matrix* Memory Usage Comparison

matrix_after_pca.txt	11/30/2024 2:30 AM	Text Document	5,544 KB
matrix_before_pca.txt	11/30/2024 2:30 AM	Text Document	7,512 KB

[2]Note that the team used the built-in Python start/stop timer to establish benchmarks for the total time to execute the KNN model pre- and post-PCA projection on the same machine.

Impressively, the computation time also decreased drastically. Pre-PCA time showed **11.053664445877075 seconds**, compared to a significantly quicker **2.2573747634887695 seconds** post-PCA. This boasts a 79.58% decrease in computation time.

Limitations

1. The true data is restricted to being linearly independent, or must be assumed to be so as a means of simplification. This is because when the projection occurs through PCA, it is of a linear fashion. Hence, if, for instance akin to what is observed in IRT, there exists a logistic or exponential relation between student ability and the probability of a correct answer, this must be simplified in order to satisfy the preconditions for a valid projection onto a new subspace.

2. PCA is susceptible to be significantly worse in its performance if the data is riddled with outliers. Such outliers can shift the eigenvalues observed in the variance matrix due to the fact that the outliers skew certain members of the variance matrix to higher values. Hence, if data is otherwise centered with a couple of distant outliers, the top eigenvalues selected as part of the PCA algorithm will be "incorrect" in truly representing data.

3. Data that is noisy with respect to large number of features is susceptible to a lack of capturing the finer details when PCA is applied. An example of this is the lack of balance between dimensionality reduction and information (variance) retention when $c = 5$ in the team's experiment. Hence, there is an dimensionality-information trade-off that must be considered carefully when using PCA.

Wrap-Up and Final Thoughts

Although the team was disappointed to see limitations of both attempts being in the form of a lack of growth in the base KNN model's accuracy, the impressive base model and gains in memory usage and computational efficiency will deliver great value to Khan Academy's mission.

To revisit Question 2, Khan Academy is now better equipped to suggest learning material based on the predictive modeling and insight into the company's data provided in this report; Khan Academy is now able to effectively predict student performance on select tagged data (using the company's subject metadata) based on their completed diagnostic questions. Using the models created in this report, they can do this at scale with increased memory efficiency and lowered computational cost to identify which subjects each student is struggling with, and tailor their learning experience accordingly. The team is proud to have had the opportunity to work with the influential Khan Academy to leave a lasting impact on the future of the EdTech industry.

References

[1] Plotly. (n.d.). PCA visualization. Retrieved November 30, 2024, from https://plotly.com/python/pca-visualization/

[2] Rahul G. Krishnan. (2024). *CSC311 - Introduction to Machine Learning: Lecture 1 Notes*. University of Toronto. Retrieved from https://www.cs.toronto.edu/ rahulgk/courses/csc311_f24/lectures/lec01.pdf

[3] Stack Overflow. (2019, January 15). How to choose the number of components PCA (scikit-learn)? Stack Overflow. Retrieved from
https://stackoverflow.com/questions/53802098/how-to-choose-the-number-of-components-pca-scikit-learn