Predicting Grade-Level Difficulty of Math Problems Using Machine Learning

Ruben Martinez

Professor: Dr. Olac Fuentes

CS 5361/6361: Machine Learning

University of Texas at El Paso

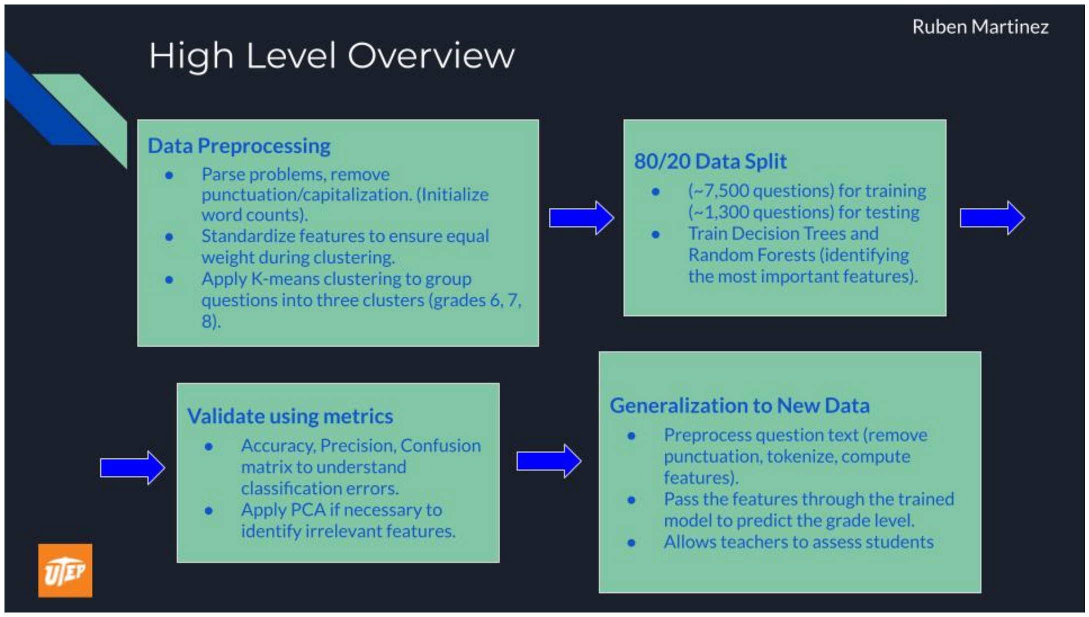
December 12, 2024

Contents

1 Introduction

As a math tutor at Mathnasium, I have consistently observed the challenges that arise when students work on math problems that do not match their skill level. Problems that are too simple fail to engage students, making the learning process feel repetitive and uninspiring, while overly advanced problems often lead to frustration and a loss of confidence. This mismatch disrupts their progress, leaving students either disheartened or unmotivated. Inspired by these challenges, I developed a project that leverages machine learning to automate the process of categorizing math problems by grade level. Using the GSM8K dataset, a high-quality collection of middle-school-level math problems, this project aims to predict the approximate grade level—6th, 7th, or 8th grade—of math problems.

To achieve this, I extracted meaningful features from the dataset, such as word count, sentence complexity, and mathematical operations, and applied clustering algorithms to group the problems into categories based on their complexity. These clusters were then mapped to grade levels, and supervised classification models, including Logistic Regression, Decision Trees, Random Forests, and K-Nearest Neighbors, were trained to refine the predictions. This system not only achieves high accuracy but also offers insights into what makes a math problem suitable for a particular grade. By addressing the recurring issue of mismatched problem difficulty, this project provides a scalable and objective tool to help educators better align problems with students' abilities, enhancing engagement and confidence in learning mathematics. Figure 1 provides a high-level overview of the project workflow, from data preprocessing to final predictions.


High Level Overview

Ruben Martinez

Data Preprocessing
- Parse problems, remove punctuation/capitalization. (Initialize word counts).
- Standardize features to ensure equal weight during clustering.
- Apply K-means clustering to group questions into three clusters (grades 6, 7, 8).

80/20 Data Split
- (~7,500 questions) for training (~1,300 questions) for testing
- Train Decision Trees and Random Forests (identifying the most important features).

Validate using metrics
- Accuracy, Precision, Confusion matrix to understand classification errors.
- Apply PCA if necessary to identify irrelevant features.

Generalization to New Data
- Preprocess question text (remove punctuation, tokenize, compute features).
- Pass the features through the trained model to predict the grade level.
- Allows teachers to assess students

Figure 1: High-level overview of the project workflow.

2 Motivation and Background

One of the most rewarding aspects of tutoring is witnessing students' growth as they master increasingly complex concepts. However, this growth requires problems that are challenging yet achievable. From my experience, assigning the correct difficulty level is crucial but not always straightforward. This inspired me to develop a tool that automates the process of categorizing problems by grade level, saving time for educators and improving student outcomes.

This project aligns with broader trends in adaptive learning technologies, which aim to personalize educational experiences. By predicting grade levels, this tool has the potential to enhance adaptive learning platforms, enabling educators to focus more on teaching and less on manual grading or problem selection.

3 Dataset and Challenge of Missing Grade-Level Labels

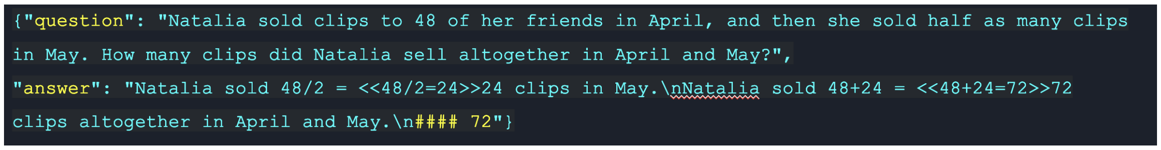
The GSM8K dataset consists of around 8,500 math problems targeting middle-school-level mathematics. Each problem includes:

- A word problem framed around real-world scenarios,
- A human-written solution or reasoning process,
- A final numerical answer.

The dataset is high-quality and provides a great variety of math problems designed for middle-school students, but the lack of explicit grade-level labels made this project more challenging. Without these labels, I couldn't use supervised learning to directly train a model to predict grade levels. Instead, I had to rely on unsupervised methods like K-Means clustering to group the problems based on features I extracted. The tricky part is that clustering doesn't automatically tell me which grade a group of problems belongs to. I had to make my own decisions about which cluster should represent 6th, 7th, or 8th grade based on how complex the problems seemed. This adds some subjectivity to the process.

One of the challenges is that complexity isn't always clear-cut. A problem that looks advanced in one context might seem appropriate for a lower grade in another, depending on how students are taught or what their curriculum covers. Also, the clustering results depend a lot on the features I used, so if I missed anything important in the feature extraction process, the clusters might not reflect the true difficulty of the problems. For example, if I didn't fully capture the math operations or sentence complexity in a problem, it might end up in the wrong group.

Another limitation is that, without actual grade-level labels, it's hard to check if the clusters and their assigned grades are accurate. I can't compare the results to a ground truth, so I have to trust that the clustering and my mapping decisions are reasonable. While this approach works for now, having access to a dataset with explicit grade-level labels would make the results more reliable and give me a better way to validate the system.



```
{"question": "Natalia sold clips to 48 of her friends in April, and then she sold half as many clips
in May. How many clips did Natalia sell altogether in April and May?",
"answer": "Natalia sold 48/2 = <<48/2=24>>24 clips in May.\nNatalia sold 48+24 = <<48+24=72>>72
clips altogether in April and May.\n#### 72"}
```

Figure 2: A single test instance from the GSM8K dataset.

4 Data Preprocessing and Feature Extraction

Getting the data ready for analysis was an important step in this project because raw text can't be directly used by machine learning algorithms. The math problems in the dataset had to be cleaned up and transformed into a format that the models could understand. To do this, I applied a few key preprocessing steps:

- **Tokenization**: Each math problem was split into individual words, and punctuation was removed. This helped break the text into smaller pieces that were easier to analyze.

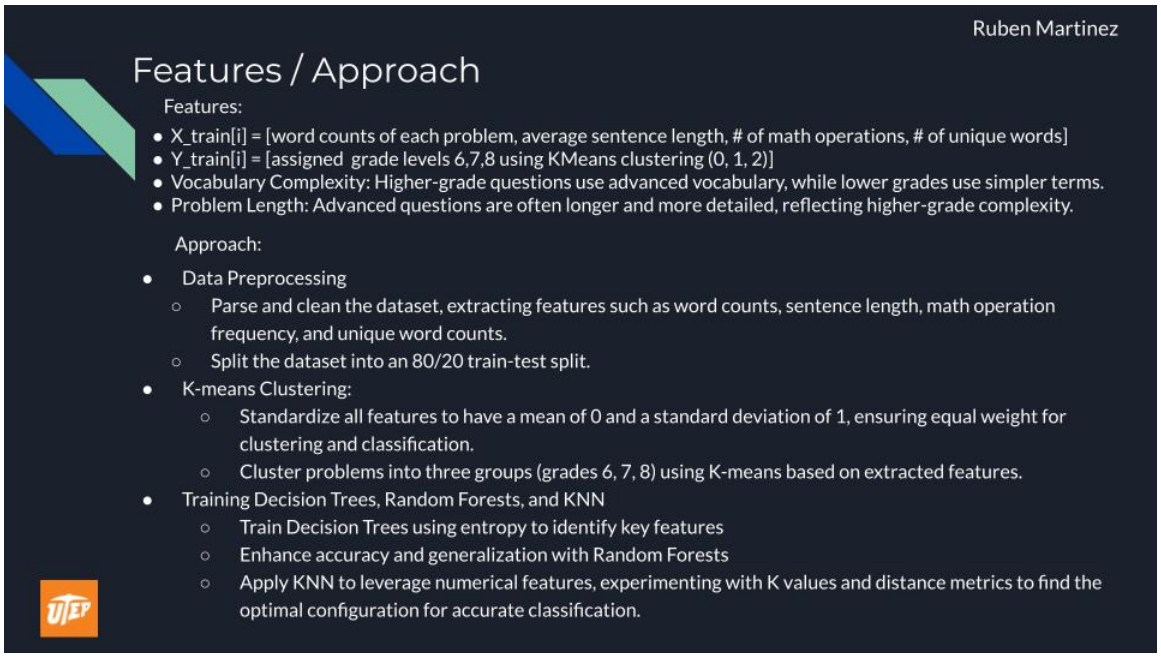
- **Lowercasing**: All the text was converted to lowercase so that words like "Math" and "math" would be treated as the same word. This step was about keeping everything consistent.

- **Vocabulary Reduction**: I narrowed the focus to the 100 most common words in the dataset. Rare words were left out because they don't appear often enough to provide useful information for predicting grade levels. Reducing the vocabulary also helped simplify the analysis and kept the model from being overwhelmed with unnecessary details.

After cleaning up the data, I extracted features from each math problem to capture its complexity and characteristics. These features are what the machine learning models used to make predictions. The main features included:

- **Total Word Count**: This counted how many words were in the problem. Problems with more words tend to be longer and sometimes more complex.
- **Average Sentence Length**: I divided the total number of words by the number of sentences. If a problem had longer sentences, it could mean it required more reasoning or was harder to follow.
- **Math Operation Count**: This counted how many math operations, like addition or multiplication, were in the problem. Problems with more operations usually involve more steps and are harder to solve.
- **Unique Word Count**: This measured how many different words were used in the problem. A problem with a lot of unique words might be more descriptive or advanced.

These features helped turn each problem into a set of numbers that could be used by the clustering and classification models. Figure 3 shows how these features were broken down and why they're important for understanding each problem.



Ruben Martinez

Features / Approach

Features:

- X_train[i] = [word counts of each problem, average sentence length, # of math operations, # of unique words]
- Y_train[i] = [assigned grade levels 6,7,8 using KMeans clustering (0, 1, 2)]
- Vocabulary Complexity: Higher-grade questions use advanced vocabulary, while lower grades use simpler terms.
- Problem Length: Advanced questions are often longer and more detailed, reflecting higher-grade complexity.

Approach:

- Data Preprocessing
  - Parse and clean the dataset, extracting features such as word counts, sentence length, math operation frequency, and unique word counts.
  - Split the dataset into an 80/20 train-test split.
- K-means Clustering:
  - Standardize all features to have a mean of 0 and a standard deviation of 1, ensuring equal weight for clustering and classification.
  - Cluster problems into three groups (grades 6, 7, 8) using K-means based on extracted features.
- Training Decision Trees, Random Forests, and KNN
  - Train Decision Trees using entropy to identify key features
  - Enhance accuracy and generalization with Random Forests
  - Apply KNN to leverage numerical features, experimenting with K values and distance metrics to find the optimal configuration for accurate classification.

Figure 3: Explanation of extracted features and their importance.

Since the features had different scales like word count being much larger than the math operation count, I applied standardization to level the playing field. Standardization adjusted all the features so that they had a mean of 0 and a standard deviation of 1. This way, no single feature could dominate the analysis just because it had a larger range of values. For example, without standardization, a high word count might overshadow the number of math operations, even if both are equally important.

Figure 4 illustrates how standardization works. It shows the formula, $(x - \mu)/\sigma$, which adjusts each value based on how far it is from the average (μ) and divides by the standard deviation (σ). The result is a balanced dataset where all features contribute fairly to the model's decisions.

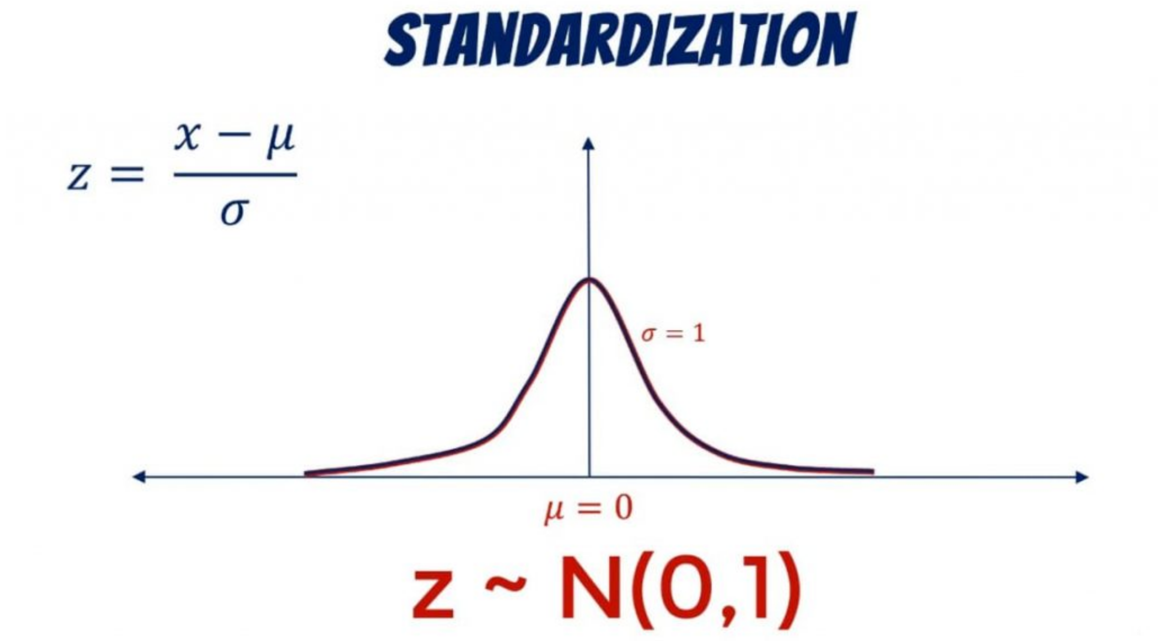


Figure 4: Standardization formula and the resulting bell curve.

By cleaning the data, picking the right features, and standardizing everything, I made sure the dataset was ready for the clustering and classification steps. These preprocessing steps set the stage for the machine learning models to understand and analyze the math problems effectively.

5 Classification Models: Decision Trees, Logistic Regression, and More

Once the clustering step was done, I trained several classifiers to predict the grade levels of the math problems. Each model brought something different to the table and helped me better understand how well the features I chose could separate the problems by grade. Here's a breakdown of the models I used:

- **Decision Trees**: These models are great for understanding how decisions are made because they clearly show the paths taken to arrive at a prediction. They also highlight which features are the most important. While their accuracy wasn't the highest, they offered valuable insights into the relationships between features and grade levels.

- **Logistic Regression**: This model ended up being the most accurate, achieving a 99% accuracy rate. It worked so well because the features were fairly simple and the relationships between them were mostly linear. Logistic Regression is also efficient and easy to interpret, making it a strong choice for this task.

- **Random Forests**: This model combined multiple Decision Trees to make its predictions. The idea was that by averaging the results of several trees, I could reduce the chance of overfitting and improve accuracy. Random Forests performed well, but they were slightly less accurate than Logistic Regression.

- **K-Nearest Neighbors (KNN)**: This model predicted the grade level of a problem by looking at the "distance" between it and other problems in the dataset. While it wasn't as accurate as the other models, KNN was helpful in understanding how important it was to standardize the features. Without standardization, the model's predictions wouldn't have been as reliable.

Each model contributed to the project in its own way. Logistic Regression stood out for its accuracy and simplicity, making it the best performer overall. Decision Trees were especially helpful for understanding which features mattered most, while Random Forests offered a good balance between accuracy and robustness. Even though KNN didn't perform as well, it highlighted the importance of feature scaling and helped me appreciate the role of distance metrics in classification tasks.

Together, these models provided a well-rounded view of how the data could be analyzed and how features like word count and sentence complexity impact predictions. By comparing their strengths and weaknesses, I was able to choose the right tools for different parts of the project.

6 Experimental Setup

To evaluate how well the machine learning models could predict the grade levels of math problems, I split the dataset into training and testing sets. I used 80% of the data for training and reserved the remaining 20% for testing. This split ensured that the models were trained on a large portion of the dataset while still leaving enough unseen data to fairly assess their performance. After the split, the training set contained 7,473 math problems, while the test set contained 1,319 problems. Each problem was represented as a feature vector with 100 dimensions, corresponding to the structural and linguistic features extracted during preprocessing. This resulted in training and test matrices with shapes of (7473, 100) and (1319, 100), respectively.

The first step in the experimental process was clustering the training data using K-Means. The clustering algorithm grouped the problems into three clusters based on their features. These clusters were then heuristically mapped to grade levels. To determine the mapping, I calculated the mean feature values for each cluster and ordered the clusters by their complexity. The cluster with the lowest average complexity was assigned to grade 6, the middle cluster to grade 7, and the most complex cluster to grade 8. This mapping was implemented using the following logic:

```
cluster_centers = kmeans.cluster_centers_
sorted_clusters = np.argsort([np.mean(center) for center in cluster_centers])
grade_mapping = {sorted_clusters[i]: i for i in range(3)}  # Maps to 0: 6th
   grade, 1: 7th grade, 2: 8th grade
```

With this mapping, class 0 corresponded to grade 6, class 1 to grade 7, and class 2 to grade 8. This process ensured that the clusters were interpreted in a meaningful way, aligning with the expected progression of problem complexity across grade levels.

Once the clusters were mapped to grades, I labeled the training data based on the assigned cluster grades. The classifiers were then trained on this labeled data, learning to predict the grade levels of new problems. The models were evaluated on the test set using several key metrics:

- **Accuracy**: This metric measured the overall correctness of the predictions, providing a high-level view of model performance.
- **Precision**: Precision evaluated how well the model avoided false positives by measuring the proportion of correctly predicted grades among all predictions for a given grade level.
- **Confusion Matrix**: The confusion matrix gave a detailed breakdown of how well the model performed for each grade, showing the number of correct and incorrect predictions for each class. This helped identify specific areas where the models struggled.

This setup allowed me to assess how well the clustering and classification steps worked together. By splitting the data and mapping the clusters to grades, I created a framework for evaluating the models and understanding how the extracted features impacted their ability to predict grade levels accurately.

7 Results and Conclusion

Out of all the models I tested, Logistic Regression performed the best, achieving a 99% accuracy. This result showed that the features I selected worked well for separating the math problems into the correct grade levels. Random Forests came in second, with solid accuracy, and Decision Trees followed closely behind. Although Decision Trees didn't perform as well as Logistic Regression, they were still valuable because they helped me understand which features were most important in making predictions.

When I looked deeper into the results, most of the mistakes the models made were between grade levels that are next to each other, like between grade 6 and grade 7 or between grade 7 and grade 8. This makes sense because the differences in complexity between these grades aren't always clear-cut. A problem that's easy for one 7th grader might feel more like a 6th-grade problem to another, depending on their background or how it's worded.

Overall, the models handled the task well, especially considering the challenges of not having explicit grade-level labels in the dataset. Logistic Regression stood out for its simplicity and effectiveness, while Random Forests showed strong performance by combining multiple decision trees. Decision Trees, despite their slightly lower accuracy, offered clear insights into the relationships between the features and the grade levels. These results gave me confidence that the system could accurately predict grade levels and highlighted areas where it could be improved.

This project demonstrates how machine learning can automate grade-level prediction, enhancing educational tools and adaptive learning platforms. Future work includes refining feature extraction, exploring semi-supervised learning, and incorporating expert feedback to improve the labeling process.

References

1. *Source Code Colab Notebook*. Google Colab, `https://colab.research.google.com/drive/1J1B-lNMObjyV`
   `usp=sharing`.

2. *GSM8K Dataset*. GitHub repository, `https://github.com/openai/grade-school-math/tree/master`.

3. *Machine Learning Project Presentation Slides*. Google Slides, `https://docs.google.com/presentation/d/1_lXEdDKHGeFpvfVsC9wQiamiiZMBhpbHuLdQCI1NBO4/edit?usp=sharing`.