Anomaly Detection with Isolation Forest and Segmentation with K-means Clustering

A Brief Introduction to Unsupervised Learning in Student Data Analysis

Saugat The Great

December 1, 2025

Presentation Outline

Abstract

The presentation explores the application of unsupervised machine learning techniques to analyze student interaction data.

- The workflow focused on robust data preprocessing (handling, cleaning, and feature extraction) to prepare the dataset.
- Anomaly Detection was performed using the highly effective Isolation Forest algorithm to identify irregular student behaviors.
- Segmentation was achieved with K-means Clustering to group students into $K = 4$ distinct behavioral clusters for targeted intervention and analysis.

Project Focus and Methodology

Project Focus: This project details the comprehensive process of data ingestion, cleaning, transformation, and analysis. The core objective is the implementation of Anomaly Detection and data segmentation.

Methodology: The workflow emphasizes robust data preprocessing (handling, cleaning, and manipulation) to prepare the dataset before applying the machine learning models.

Key Technologies and Libraries Used (Part 1/2)

Key Technologies and Libraries:

1. NumPy: Essential for high-performance numerical computation and serving as the foundational internal data structure for array manipulation.
2. Pandas: Used for efficient data loading, handling, and analysis via DataFrames.
3. Matplotlib: Primary library for generating static and interactive visualizations.

Key Technologies and Libraries Used (Part 2/2)

Key Technologies and Libraries:

4. Seaborn: Statistical visualization library; provides a high-level interface for attractive statistical graphics.
5. Scikit-learn (sklearn): Comprehensive library for implementing the machine learning algorithms, including Isolation Forest and K-means.

Data Handling and Cleaning

Data Overview and Initial Preparation

Source of the Data

- The data is collected from UCI Machine Learning Repository.
- Initial total number of individual interactions: 9546.
- The dataset contains 8 columns and they are: `student_id`, `country`, `question_id`, `is_correct`, `q_level`, `topic`, `subtopic`, and `keywords`.

Data Information:

- Handling Missing Values: A full check revealed no missing values, thus no imputation techniques were necessary.
- Column Renaming: Columns were renamed for enhanced readability and ease of programming, e.g., 'Student ID' $\rightarrow$ 'student_id'.

Raw Dataset Schema

This table summarizes the eight columns used in the initial data ingestion.

Feature Name	**Data Type**	**Description**
`student_id`	Integer	Unique identifier for each student.
`country`	String	Geographic location of the student.
`question_id`	String	Unique identifier for the math question.
`is_correct`	Integer (0/1)	Target Outcome: 1 for Correct, 0 for Incorrect attempt.
`q_level`	String	Difficulty level of the question.
`topic`	String	Broad math topic (e.g., Algebra, Calculus).
`subtopic`	String	Specific sub-topic within the main topic.
`keywords`	String	Tags/keywords associated with the question content.

Duplicate Data Filtering

Handling Duplicate Values

- 2083 rows were found to be identical instances of data.
- Rationale: Given the nature of a single student interaction being logged, identical rows likely represent data collection errors rather than simultaneous events. These duplicates were removed.
- Final Interaction Count $= 9546 - 2083 = 7463$ interactions remaining.

Low-Activity Student Filtering

Filtering Low-Activity Students

- Students with fewer than a minimum number of interactions ($MIN\_ATTEMPTS = 5$) were removed from the analysis.
- Rationale: Students with very few attempts do not provide enough data to form reliable behavioral features, and their inclusion could skew the clustering and anomaly detection processes.

This ensures the final dataset for feature engineering represents only students with meaningful activity.

Feature Engineering

Feature Engineering: Overview

The raw interaction data (long format) was aggregated to create a single feature vector (wide format) for each unique student ($\sim$ 372 students after filtering).

Goal: Transform individual question attempts into comprehensive student behavioral profiles.

Feature Categories:

1. Basic Performance Metrics
2. Difficulty-Level Accuracy
3. Topic Diversity

This process transforms the focus from individual attempts to student-level behavior, enabling machine learning algorithms to identify patterns.

Category 1: Basic Performance Features

Feature 1: Average Accuracy

- Measures overall student performance across all attempts
- For student i with n_i total attempts:

$$Avg\_Accuracy_i = \frac{\text{Number of correct answers}}{n_i}$$

- Range: $[0, 1]$ where $1 = 100\%$ correct

Feature 2: Total Questions

- Counts the total number of questions attempted
- For student i:

$$Total\_Questions_i = n_i$$

- Indicates engagement volume and persistence

Category 2: Accuracy by Question Difficulty

Motivation: Students may excel at easy questions but struggle with hard ones, or vice versa. We need to capture performance at each difficulty level.

Feature Creation: For each unique difficulty level ℓ (e.g., $\ell \in \{1, 2, 3, 4\}$), create separate accuracy features:

Formula for Acc_ℓ (Accuracy at level ℓ):

$$Acc\_\ell_i = \frac{\text{Correct at level } \ell}{\text{Total attempts at level } \ell}$$

Where only questions of difficulty level ℓ are considered for student i.

Category 2: Example Features

Example Features Created:

- Acc_1: Accuracy on Level 1 (easiest) questions
- Acc_2: Accuracy on Level 2 questions
- Acc_3: Accuracy on Level 3 questions
- Acc_4: Accuracy on Level 4 (hardest) questions

Handling Missing Data:

- If a student didn't attempt any questions at level ℓ, then $Acc\_\ell = 0$
- This captures both performance AND engagement patterns

Insight: These features help distinguish high performers from struggling students at different difficulty levels.

Category 3: Topic Diversity Metric

Feature: Topic Diversity

- Measures the breadth of topics a student has explored
- For student i:

$$\mathit{Topic\_Diversity}_i = \text{Count of unique topics attempted}$$

- Higher values indicate broader exploration

Example: If a student attempted questions from Algebra, Calculus, and Geometry:

$$\mathit{Topic\_Diversity} = 3$$

Insight: This feature distinguishes students who:

- Focus narrowly on few topics (specialists)
- Explore widely across curriculum (generalists)

Summary of Engineered Features

Final Feature Vector for each student contains:

Feature Name	**Description**
$Avg\_Accuracy$	Overall accuracy rate
$Total\_Questions$	Total questions attempted
$Acc\_1, Acc\_2, \ldots$	Accuracy at each difficulty level
$Topic\_Diversity$	Number of unique topics explored

Dimensionality: Each student represented by ∼7-10 features (depending on number of difficulty levels)

Result: Transformation from 7463 interactions to 372 student profiles ready for ML algorithms.

Standardization for Model Readiness

Machine learning algorithms like K-means and Isolation Forest are highly sensitive to the scale of features. Features with a larger magnitude (e.g., Total Attempts) would unfairly dominate the distance calculations.

Technique: Standard Scaling (Z-score Normalization)

- The features were standardized to have a mean of 0 and a standard deviation of 1.
- For a feature value x, the standardized value z is:

  $$z = \frac{x - \mu}{\sigma}$$

  Where μ is the mean and σ is the standard deviation of the feature across all students.

This ensures all features contribute equally to the distance metrics used in the subsequent algorithms.

Isolation Forest for Anomaly Detection

Isolation Forest: Introduction to Anomaly Detection

What is Isolation Forest?

Isolation Forest (iForest) is an efficient algorithm for unsupervised anomaly detection based on a simple principle:

"Anomalies are few and different, thus easier to isolate"

Goal: Identify students with significantly irregular behavioral patterns compared to the general population.

Key Principle:

- **Normal points**: Dense, similar to neighbors, require many splits to isolate
- **Anomalies**: Sparse, different from others, isolated quickly with few splits

How Isolation Forest Works

Step-by-Step Process:

1. **Build Isolation Trees (iTrees)**:
   - Randomly select a feature f from $FEATURE\_COLS$
   - Randomly select a split value v between $\min(f)$ and $\max(f)$
   - Split data: points where $f < v$ go left, $f \geq v$ go right
   - Repeat recursively until each point is isolated
2. **Create an Ensemble**: Build $n\_estimators = 100$ trees
3. **Measure Path Length** $h(x)$:
   - Count edges from root to leaf for each point
   - Shorter paths $\Rightarrow$ easier isolation $\Rightarrow$ likely anomaly
   - Longer paths $\Rightarrow$ harder isolation $\Rightarrow$ likely normal

iForest: Path Length and Anomaly Score Formula

Step 1: Calculate Path Length $h(x)$

- $h(x) =$ number of edges from root to leaf for point x in one tree
- Average across all trees: $E[h(x)] = \frac{1}{100} \sum_{t=1}^{100} h_t(x)$
- Shorter $E[h(x)] \Rightarrow$ anomaly; Longer $E[h(x)] \Rightarrow$ normal

Step 2: Normalize with $c(n)$

- $c(n) = 2H(n-1) - \frac{2(n-1)}{n}$ where $H(k) = \ln(k) + 0.5772$ (harmonic number)
- $c(n) =$ average path length in a BST of n points
- Used to normalize scores across different dataset sizes

Step 3: Compute Anomaly Score $s(x) \in [0, 1]$:

$$s(x) = 2^{-\frac{E[h(x)]}{c(n)}}$$

Interpreting the Anomaly Score

Anomaly Score $s(x) \in [0, 1]$ interpretation:

Score Range	**Interpretation**
$s(x) > 0.6$	Strong anomaly (very short path)
$s(x) \approx 0.5$	Boundary case (unclear)
$s(x) < 0.5$	Normal instance (long path)

Why this works?

- When $E[h(x)]$ is very small (easy isolation): $s(x) \to 1$
- When $E[h(x)] \approx c(n)$ (normal): $s(x) = 2^{-1} = 0.5$
- When $E[h(x)]$ is large (hard isolation): $s(x) \to 0$

Applying Isolation Forest to Student Data

Model Configuration:

- `n_estimators = 100`: Build 100 isolation trees
- `contamination = 0.03`: Expect 3% of data to be anomalous
- `random_state = 42`: For reproducibility

Detection Strategy:

1. Calculate anomaly score $s(x)$ for each of 372 students
2. Rank students by score (highest = most anomalous)
3. Label top 3% (contamination rate) as anomalies
4. Output: $+1$ (normal) or -1 (anomaly) for each student

Result: **11** students ($\approx 3\%$ of 372) flagged as anomalies

These students exhibit highly irregular patterns in accuracy, engagement, or topic diversity compared to their peers.

K-means Clustering

K-means Clustering: Segmenting Student Behavior

What is K-means Clustering?

K-means is a partitioning algorithm that groups data points into K clusters by minimizing within-cluster variance.

Goal: Partition 361 normal students (after removing 11 anomalies) into K behavioral groups.

Intuition: Students in the same cluster should be similar to each other, and different from students in other clusters.

Why K-means?

- Simple, fast, and scalable
- Works well with numerical features
- Produces interpretable cluster centroids (average student profiles)

How K-means Works (Part 1/2)

Step 1 - Initialization:

- Randomly select K students as initial centroids $\mu_1, \mu_2, \ldots, \mu_K$
- Using `n_init=10`: try 10 different random initializations, pick the best result
- This helps avoid poor local optima

Step 2 - Assignment (E-Step):

- For each student x_i, assign to the nearest centroid:

$$c_i = \arg \min_{j \in \{1,\ldots,K\}} \|x_i - \mu_j\|^2$$

- $\|x_i - \mu_j\|^2$ = squared Euclidean distance between student i and centroid j
- Intuitively: assign each student to their closest cluster center

How K-means Works (Part 2/2)

Step 3 - Update (M-Step):

- Recalculate each centroid as the mean of all students assigned to it:

$$\mu_j = \frac{1}{|S_j|} \sum_{x_i \in S_j} x_i$$

- Where $S_j =$ set of students assigned to cluster j
- Intuitively: move the cluster center to the average position of its members

Step 4 - Convergence:

- Repeat steps 2-3 until centroids stop moving (or change very little)
- Typically converges in 10-50 iterations
- Result: Final cluster assignments for all students

Mathematical Objective: Within-Cluster Sum of Squares

Optimization Goal: Minimize the Within-Cluster Sum of Squares (WCSS or Inertia)

Objective Function J:

$$J = \sum_{j=1}^{K} \sum_{x_i \in S_j} \|x_i - \mu_j\|^2$$

Where:

- J = Total inertia (lower is better)
- K = Number of clusters (e.g., $K = 4$)
- S_j = Set of students in cluster j
- x_i = Feature vector for student i
- μ_j = Centroid (mean) of cluster j
- $\|x_i - \mu_j\|^2$ = squared Euclidean distance

Method 1: Elbow Method (WCSS vs K)

Approach: Plot inertia (WCSS) for different values of K

Procedure:

1. Run K-means for $K \in \{2, 3, 4, \ldots, 9\}$
2. Calculate J (inertia) for each K
3. Plot J vs K
4. Look for an "elbow" where J drops sharply then levels off

Interpretation:

- Before elbow: Adding clusters significantly reduces inertia
- After elbow: Diminishing returns (minor improvement)
- The elbow point = optimal balance between complexity and fit

Result: Elbow observed around $K = 4$

Method 2: Silhouette Score Analysis

Silhouette Score: Measures how well-separated and compact clusters are

Formula for student i in cluster C_j:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

Where:

- $a(i) =$ mean distance from i to other students in same cluster
- $b(i) =$ mean distance from i to students in nearest other cluster

Score Range: $s(i) \in [-1, +1]$

- $s(i) \approx +1$: Well-clustered (far from other clusters)
- $s(i) \approx 0$: On cluster boundary
- $s(i) \approx -1$: Possibly mis-clustered

Choosing K=4 and Running Final Model

Decision: $K = 4$ clusters selected based on:

- Elbow point in WCSS plot
- Highest Silhouette Score
- Interpretability (meaningful student segments)

Final Model Configuration:

- `n_clusters = 4`
- `n_init = 10`: Run 10 times with different initializations
- `random_state = 42`: For reproducibility
- Input: 361 normal students (after anomaly removal)

Output: Each student assigned to one of 4 clusters

Next Step: Analyze cluster centroids to interpret behavioral patterns

Interpreting the $K = 4$ Student Clusters

The final step is to interpret the meaning of the clusters by analyzing their Centroids (mean feature values) for the $K = 4$ segments.

Example Cluster Characteristics (based on standardized centroid analysis):

- Cluster 0: The Engaged High Achievers (High CR, very high Diversity)
- Cluster 1: The Struggling but Persistent (Low CR, high Total Attempts)
- Cluster 2: The Focused Low Engagers (Average CR, low Diversity and Total Attempts)
- Cluster 3: The Efficient High Achievers (Very high CR, moderate Attempts)

Conclusion

Summary of Findings

- The data was successfully preprocessed, resulting in a clean dataset of 7463 interactions from robust students.
- Isolation Forest effectively flagged a small fraction (3%) of students as anomalous, suggesting they exhibit highly irregular behavior compared to the norm.
- K-means Clustering successfully segmented the students into 4 distinct behavioral groups, which can be used to tailor educational strategies (e.g., targeted assistance for struggling clusters).

Future Work:

- Explore DBSCAN or OPTICS clustering for natural density-based groups.
- Incorporate temporal features (e.g., time between attempts) into the feature set.