Group 120: Machine Project Overview

Introduction

Determining the difficulty level of a course is often challenging for students, as multiple factors contribute to how demanding a class may be. In this project, we aim to simplify this process by analyzing three primary data points: grade distributions, professor ratings, and class sizes at Georgia Tech. Our objective is to classify classes into difficulty tiers, such as “Easy,” “Medium,” and “Hard,” using machine learning techniques. By making these insights available, we hope to provide students with an accessible resource for better course planning.

Problem Definition

Our approach addresses a common dilemma for students: assessing the effort required for various courses. Instead of relying solely on subjective online reviews or anecdotal feedback, our model will use actual performance data. This structured classification can serve as an informed guide to help students anticipate course rigor and workload.

Data Collection & Preprocessing

Our initial ideas for data were ambitious. We have not been able to get professor ratings from external sources nor class sizes in an efficient way without manually combing through each class and professor for each data point. We still would like to do this but are still figuring out an automatic way to do this. However, we were able to obtain the Georgia Tech grade distribution data from the owners, the Office of Institutional Research & Planning. This data included many features of each class, but we decided to use only average grade, standard deviations of the grades, instructor name, subject, and course number.

Most of our preprocessing included dropping columns and encoding strings into numbers. For dropping columns and rows, we removed any labs and recitations from our data to remove significant outliers as well as any classes missing an average grade as this indicated a 100% withdrawal rate. Our main preprocessing method was data encoding, where we would convert strings into integers. We did this both for subject and for instructor. For our true labels, we decided to initially assign certain classes “very easy”, “easy”, “medium”, “hard”, “very hard” based on our own understanding of the course number and average GPA with a simple algorithm.

Machine Learning Model

K-means actual implementation

After first preprocessing our data, we then used this preprocessed data as the training data for the K-means clustering, which was our first and currently only implemented model. We selected kmeans because it's a relatively simple and easy to use starter model for our data, and to act as a baseline for later comparisons. We used scikit-learn for the k-means clustering implementation, then used Streamlit for the visualization of the results. We decided to plot the grade distribution data variables such as course number and average grade as the independent variables and the true labels as the dependent variable for better visualized clusters.

Results

As seen in the visualization, our original labels do not match the output clusters. One reason for this could be that our K-means algorithm is actually finding a different pattern to determine the difficulty of the class that we aren’t noticing. Another reason is that the variables input into the algorithm are not good indicators of the difficulty of the class. We originally thought the K-means would find a connection between higher course number and a more difficult class, but our implementation actually saw a wide variety of difficulties among the higher courses. This means course number is most likely not a good indicator of the difficulty to input for a K-means implementation. We may also be trying to find correlations between variables from classes that are too dissimilar either in course number or class type. If we chose a dataset too variably different then we would need many more input variables to accurately determine good clustering for difficulty among classes.

We also think that our encoding may have some problems as well. We have no semantics, such as encoding harder subjects with higher numbers, within our encoding. This could be acceptable in some ML methods, but K-Means is an unsupervised method that measures difference.

Next Steps

First, we want to improve our encoding methods to have more semantics, especially for kmeans. To do this, we would want professor ratings to help with our new encoding method to ensure lower rated professors are encoding with higher values. Next, we would want to introduce new preprocessing methods such as PCA to reduce the number of features. Finally, we will implement our other models, such as SVMs and random forests, for our analysis and comparison of the algorithms.

Contributions Table

- **Hayk Arsenyan**: Report Review, Methods Review

- **Alexandre Abchee**: Report Writing, Methods Review, Report Review
- **John Andrade**: Preprocessing, Methods Review, Data Collection, Report Review
- **Mattias Anderson**: Methods Implementation, Preprocessing, Results, Report Review
- **Juan Macias-Romero**: Methods Implementation, Streamlit, Data Collection, Report Review

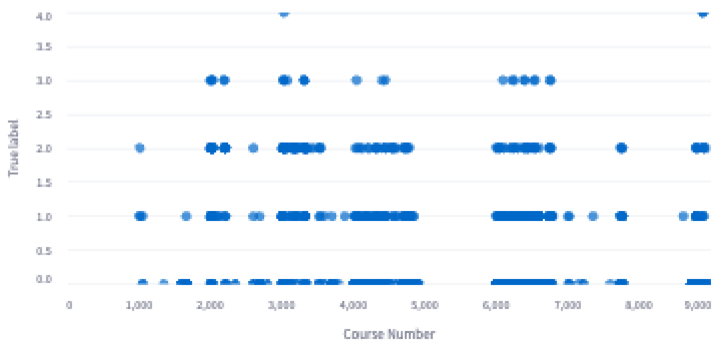
Citations

1. Small Classes, Big Possibilities [Accessed Oct. 4, 2024]
2. Grade Distribution - Georgia Tech [Accessed Oct. 4, 2024]
3. Rate My Professors [Accessed Oct. 4, 2024]

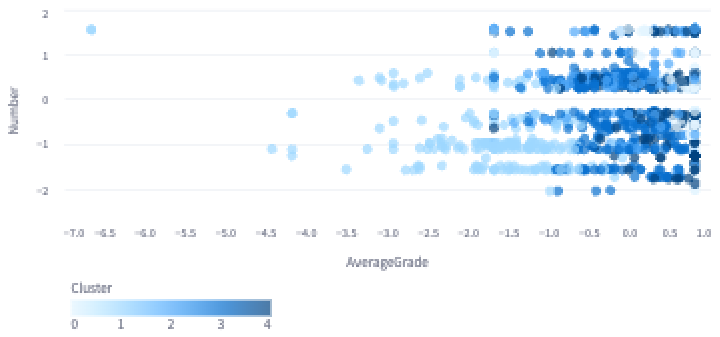
Gantt Chart

You can view our Gantt chart here.

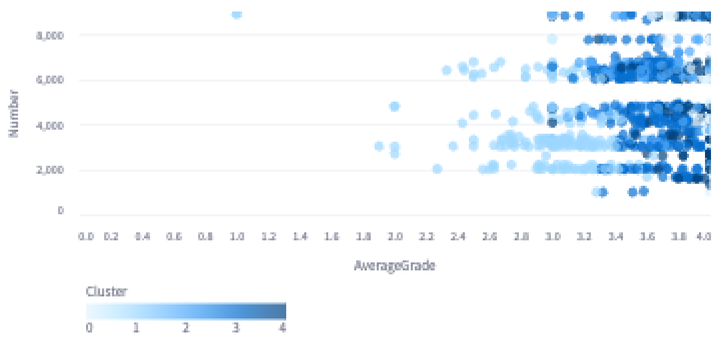
Raw data



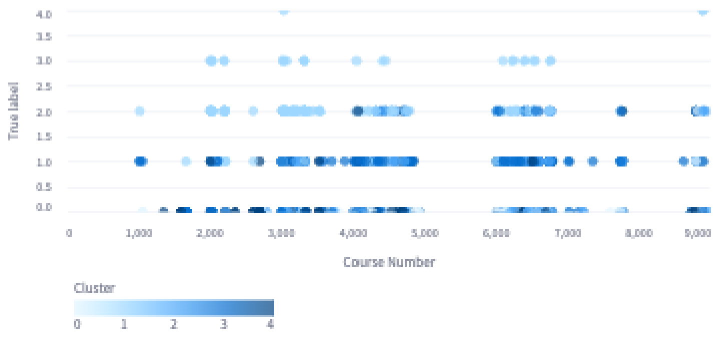
Scaled data



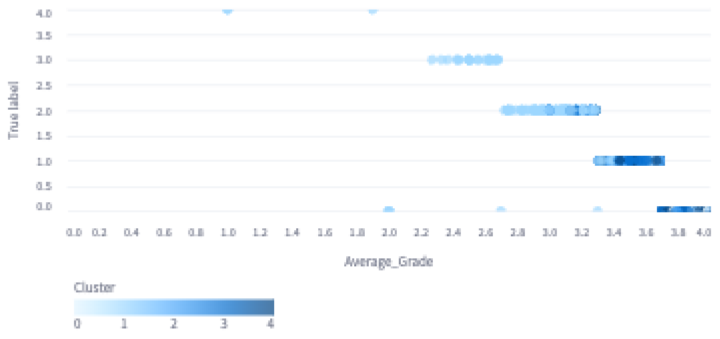
Original data



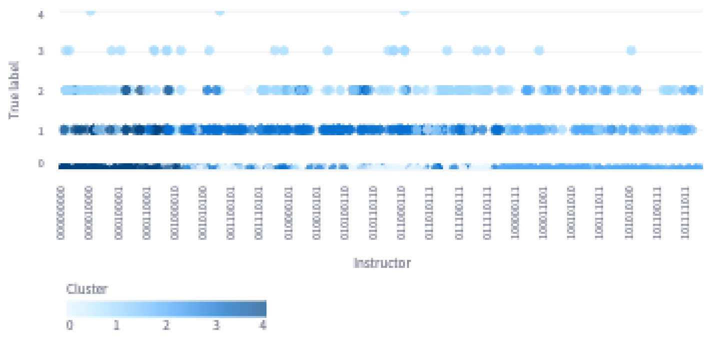
Course Number



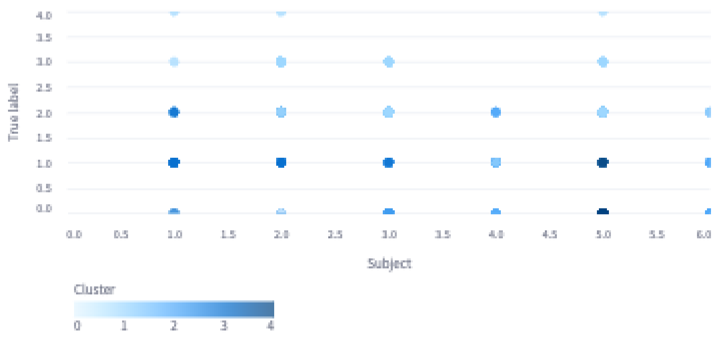
Average Grade



Instructor



Subject



Clustering Evaluation Metrics

Fowlkes-Mallows score: 0.4545787704454856

Silhouette score: 0.11361851697489748

Please read our report for more information