Exploring Classroom Emotional Dynamic Through Data Visualization and Clean Code Practices on Classroom Observation Interactive Learning System (COILS)

Major Qualifying Project

Written By:

Manh Nhu Pham

A Major Qualifying Project WORCESTER POLYTECHNIC INSTITUTE

Submitted to the Faculty of the Worcester Polytechnic Institute in partial fulfillment of the requirements for the Degree of Bachelor of Science in Computer Science.This report represents the work of one or more WPI undergraduate students submitted to the faculty as evidence of completion of a degree requirement. WPI routinely publishes these reports on the web without editorial or peer review.

January 8, 2022 - October 10, 2022

EXECUTIVE SUMMARY

In the modern world, data is an extremely powerful resource that can be used to provide invaluable insights to tackle any problem. To produce meaningful decisions for any problems, rigorous data analysis and data visualization are required. Deep and accurate technical data analysis can help identify underlying patterns to empower the decision-making process of any problems from small businesses to large corporations. However, in addition to producing these insights, making sense of these insights visually or visualizing them intuitively is an equally important process to put the knowledge into real practices. Dynamic data visualization allows us to get visual information in an easy and compelling manner from data sets. It helps to uncover new and useful insights which are otherwise difficult to discover from piles of abstract data [6]. With The Classroom Observation Interactive Learning System (COILS), WPI Computer Science department aims to employ the full cycle of data analysis process to assist pre-school teachers to better understand their class emotional and behavioral dynamics. COILS combines the analysis of data in raw forms of video including natural language processing and common technical analysis and data visualization techniques to visualize emotion data of students in a meaningful way that a normal preschool teacher can interpret easily.

COILS is a collaborative effort between multiple WPI students and computer science professionals, and as such the project has gained substantial progress over the course of development. As at the beginning of this project, the backend infrastructure had been ready for development, while the User Interface (UI) design and implementation was modest. The backend infrastructure includes a parent clip and multiple smaller clips showing different emotions. The sub-clips come with relevant data regarding emotion type, intensity, subject, and durations. Through preliminary examination, UI design seems to prioritize ease of access for clips and sub-clips with only primitive data exploration techniques. Much of the information has been discovered in the data analysis process, but not much has been dedicated to telling the story of the data effectively. In this project, we have attempted to explore different approaches to leverage the data visualization aspects of the UI. These approaches include researching on what data to visualize and what information we're trying to convey, refactoring existing JavaScript (JS) code for easier access of data, exploring different visualization techniques and the required technologies, choosing a visualization technique and implementing it.

Our main goal for this project is to use the data that we already have in the existing progress and

show the teacher the emotional trends of the students. These emotional trends include the intensity distribution of each student for each emotion and the extreme outliers for each emotion. The data used for the visualization are the emotional intensity of each clip. The intensity denoted by a single integer ranging from 0 to approximately 200. The higher the number, the more intense the emotion is. We use D3 to implement the distribution, specifically in the form of frequency histogram graph. D3 is an open-sourced JavaScript library developed specifically for data visualization. We also implement a front-end feature which provides the top 3 most intense clips for each emotion for users to quickly pinpoint individuals that might require the most attention.

Over the course of the project, we have also explored different types of visualization for our emotional trends such as the pie chart. Another possibility of the project is the graph could be further developed for ease of access and better understandability. For example, the graph could be accompanied with a common list of distribution shapes and their corresponding statistical interpretation. These interpretations are then explained in a more common-sense way, which will help teachers translate them into according actions that will positively reinforce the emotional trends.

TABLE OF CONTENTS

CHAPTER 2. INTRODUCTION

2.1 MOTIVATION

According to the National Education Association [11], Kindergarten education helps strengthen children's problem solving skills and essential social and emotional skills. Kindergarten teachers are trained to ensure that these skills are developed properly. Traditional training for these teachers often includes methods of teaching, classroom management strategies, and planning for materials and class activities. Because each student's social navigation and cognitive ability are different, some students might find it more difficult to engage in common activity in the classroom than others. Thus, it is essential for the caretaker to be flexible in their approach to take care of each student academically and socially.

In the modern world, data is quickly becoming an invaluable resource where it can be used in every profession to help elevate the results of work in addition to traditional measures. For that, using data science in education, particularly kindergarten education is no exception. According to Reuters, big data applications in the education sector accounted for 8% of the world's total in 2015 and are growing at an alarming rate of 10% a year. It is expected that by 2020, the education industry will become the eighth largest market for big data applications in the world [7]. As mentioned above, the social compass and cognitive ability could deeply vary between students. Traditionally teachers could understand these dynamics gradually by interacting with students or observing their behaviors, and thus allowing them to modify their caretaking approach to fit each student's need. Now, techniques of data science could offer teachers a deeper look into these dynamics by analyzing students' interactions and behavior and presenting the information in meaningful ways so that teachers could absorb insights easily and speed up their process of understanding their class dynamic.

This is the goal of Classroom Observation Interactive Learning System (COILS). COILS is a combination of machine learning, natural language processing, psychology, and educational theory to provide feedback and emotions seen in the classroom from teacher-student interactions (O'Conell, 2018). The tool offers the emotional analysis of students through applying natural language processing, image segmentation in captured daytime footage of students, and a carefully designed UI to help teachers interpret this statistical evidence most effectively. COILS can be splitted into 2 major components, the back-end component and the front-end component. The back-end component mainly deals with analyzing the emotions of the students. Another name for this component is called Automatic Classroom

Observation Recognition Neural Network (ACORN). On the ACORN side, video footage of classrooms are collected. The footage is pre-processed and analyzed to extract each subject's emotions. Data collected are stored based on individuals, where each subject will have their own information such as name, age, and what emotions they have and their intensity. On the other hand, the front-end side will focus on delivering the result to the users, and this is also the main focus for this project. Our goal for front-end development is to provide the teachers with straightforward interfaces that are easy to use and informative. Given that kindergarten teachers aren't always equipped with technical knowledge of data analysis and the project is under development and going to be expanded further, we attempt to explore UI designs, data visualization techniques, and code maintenance for future development.

2.2 CURRENT STATE OF ACORN AND COILS

The ACORN training and COILS platform has been through multiple development progress with multiple teams focusing on a specific component of the overall project over time. As of the time of beginning of this project, a team of graduate research has worked on data collection and analysis or ACORN while another team of WPI undergraduate has worked on the user interface side of the project or COILS. The ACORN team collected data of facial emotions through video of mock class rooms, and then used neural network algorithms and natural language processing to produce the result data that are ready to be presented for the user. The COILS team has conducted experimental research using surveys and think-out-loud sessions to examine users' preference to the page. Based on collected data, The COILS team has designed a prototype model of the user interface. The prototype includes several component mostly includes ease-of-access features like the master video, quick access to each individual's clips, and the entire video collections

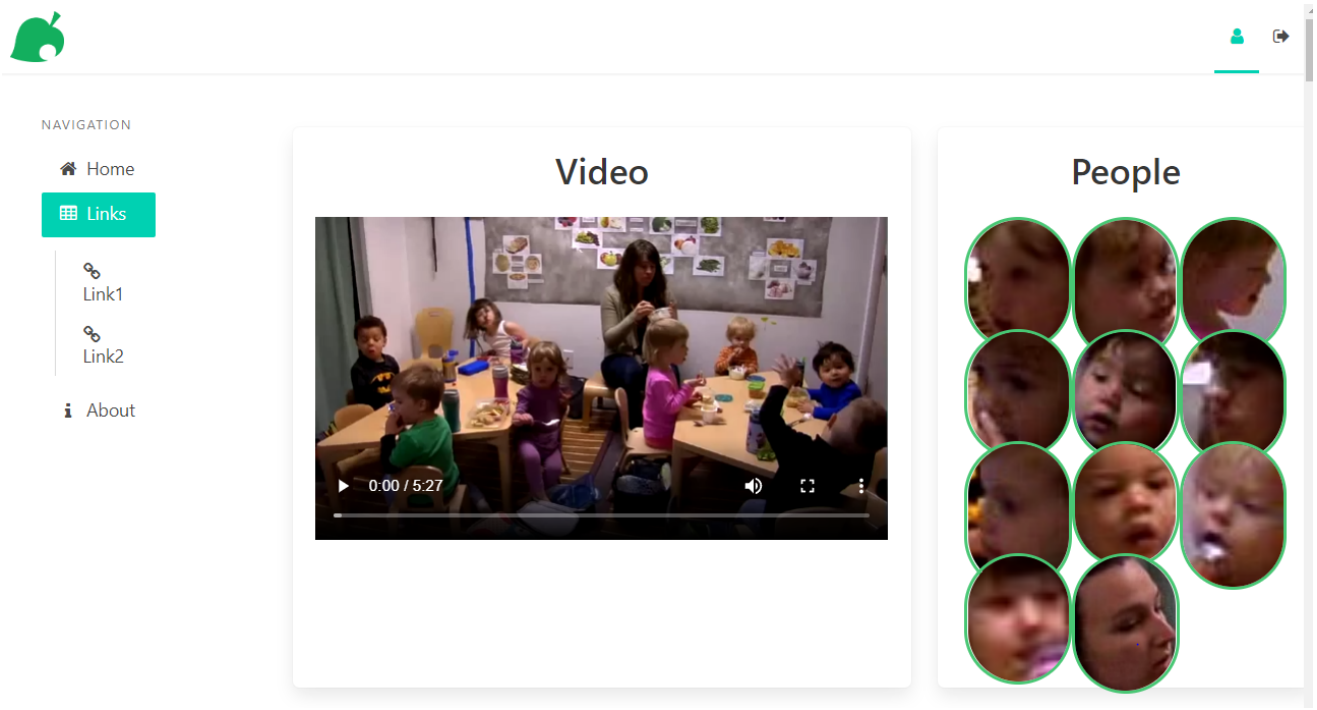

Figure 1

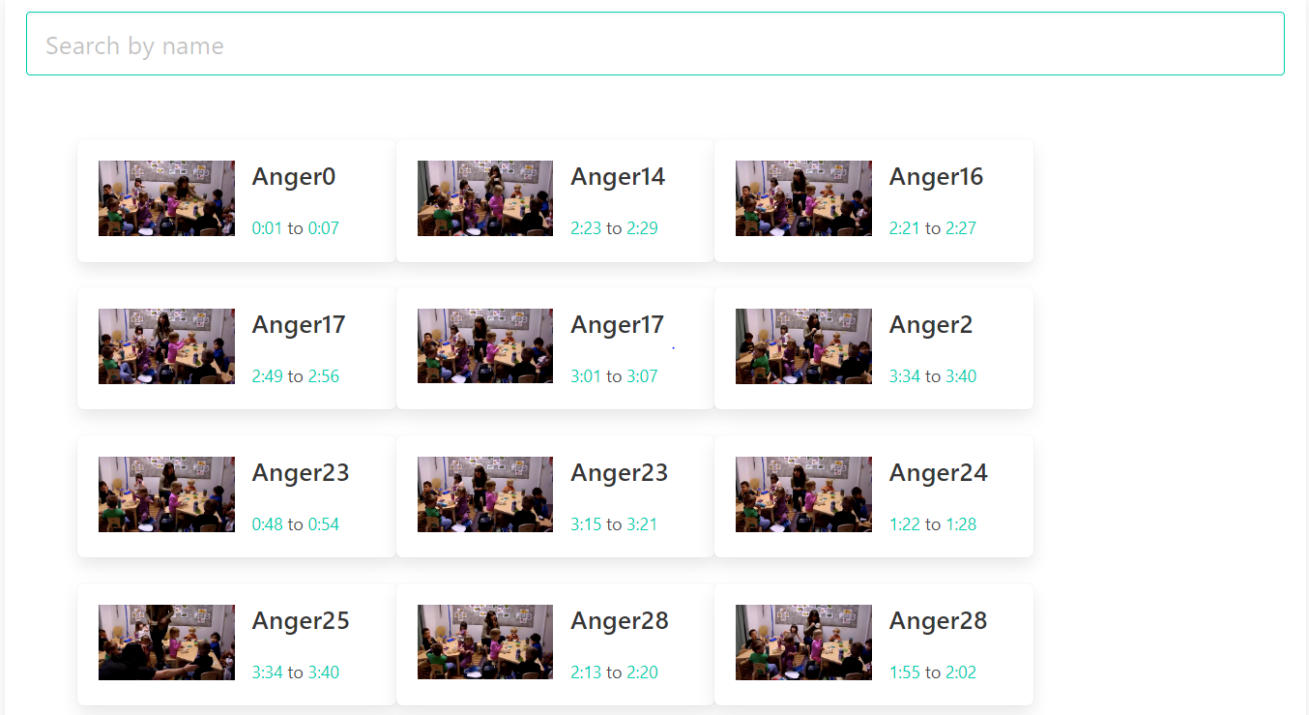

Figure 2

Figure 1. Master video and shortcuts to filter the videos based on subjects (students)

Figure 2. Collection of all subclips and simple text filter to navigate based on names of the subclips

2.3 DATA VISUALIZATION

IBM's definition of data visualization [1]:

"The representation of data through use of common graphics, such as charts, plots, infographics, and even animations. These visual displays communicate complex data relationships and data-driven insights into a way that is easy to understand"

With the rise of data science in recent years and increasingly complex algorithms and endless data and topics of interest, data visualization has become more popular and more complex to meet the demands. While data visualizations are becoming more complex, they are developed to the common goal for a technique to be deemed effective which is to make it as easy to understand as possible for users. Data visualization technique could be as simple as a scatter plot of a static csv file to interactive visualization of the solar systems allowing users to interact with the planets using animations and real time data input.

There are a lot of data visualization tools on the market. Prominent examples include Tableau, Excel, Matlab, D3, Python. Each of them has their own advantages and disadvantages. All of these tools are capable of producing high quality fundamental graphs including scatter plot, bar charts, pie charts, and more. For this project, our goal is to implement a visualization feature for our web application. Thus, D3 and Python are our most sensible choices here for their extensive open-sourced libraries that support both web-based application and data visualization. Because the demo application that had already been developed by the COILS team is written in JavaScript, D3 is preferred to extend on the progress made.

D3 is short for "Data-Driven Documents", a JavaScript library for manipulating documents based on data with HTML, SVG, and CSS. Developed by Mike Bostock, Jeff Heer, and Vadim Ogievetsky, D3 is an open-sourced project [2]. It is free, versatile, and has a wide and active community of users that are knowledgeable about the library.

2.4 CODE REFACTORING

Code refactoring is the practice of rewriting a piece of code to improve its understandability by a human programmer without altering any runtime behavior of that code. This is an important and common practice to implement in software development in general, especially where multiple programmers work on the same code base seperately and successively. However, it is often overlooked [8], especially in

prototype school projects. COILS and ACORN are projects developed asynchronously by multiple teams of students and subjected to further development in the future, which makes code refactoring a vital process to keep the code clean and cohesive for smooth future development.

Regardless of the code framework and experience of the programmer, a piece of code is almost always refactorable. For example, a common mistake in programming students often make is excessive variable declaration. Excessive variable declaration happens when variables are declared without explicitly using them to contribute to the program functionality. This is often overlooked in learning projects where the priority of the learning students is just to get the code compiled successfully (or to get it work). While it is true that this does not affect performance in short term code running, this problem long term can develop into significant decrease in performance as unused variables could take up a lot of memory space in a computer and running time. This is only a small mistake that could be reverted easily in a short time by simply deleting that declaration. However, there are a lot more ways a piece of code could be suboptimal performance wise and hard to understand. As more code is written, suboptimal implementations could stack up quickly to the point the time required to fix the mistakes would be too long that it would be more efficient to just let the code be unsanitary as it is now. It is vital that code refactor is practiced often to ensure the code is always optimized.

There are a lot of ways to refactor a piece of code. Some common techniques are abstraction, variable naming convention, and delete excess variables. Auto code refactoring is supported by popular editors on the market such as Visual Studio Code (VSC) by Microsoft, JetBrains, Eclipse. While text analysis and AI assisted refactoring systems exist to make it convenient to optimize the code, it is equally important to weigh in the design and purpose of our code base to manually customize the refactoring process.

2.5 IMPACTED STAKEHOLDERS

As we move along this project, we think of new ideas on how the project can benefit our stakeholders. Our stakeholders are teachers from grades K-12 that go through the given COILS training. The project will impact teachers in many ways but the overall goal is to improve their interactions in the classroom. We hope that with the project, teachers can improve their teachings by picking up on classroom dynamics and how students react to teachers' cues. The ability to visualize students' emotions through a variety of clips can help teachers obtain a broader sense of student behaviors. The new UI by letting teachers see snippets of clips would show teachers the actions beforehand and what can trigger

specific emotions within their students and how they react later.

2.6 CONCLUSIONS

Through this project, the users will have the chance to do their research with actual data visualization. The visualization will mainly focus on reporting the class emotion dynamics to the teachers. The UI will also provide relevant statistics of the graphs such as mean, standard deviation, and outliers. These statistics will also be accompanied with simple descriptions to inform the teachers more in depth about the dynamics they wish to research. We hope that the teachers will find the graphs helpful and their experience with the UI pleasant and easy to navigate. The UI and the visualization are prototypes at the time of this report written. They are subjected to further development based on advisor's feedback, users' feedback, and future teams' plan of improvement.

CHAPTER 3. METHODOLOGY

3.1 Objectives

Our main goal in this project is to put more data visualizations on the user interface (UI) of the application. The project was at an early stage of development at that point. The UI developed by the COILS team previously focused primarily on ease of access of the parent video and the subclips.

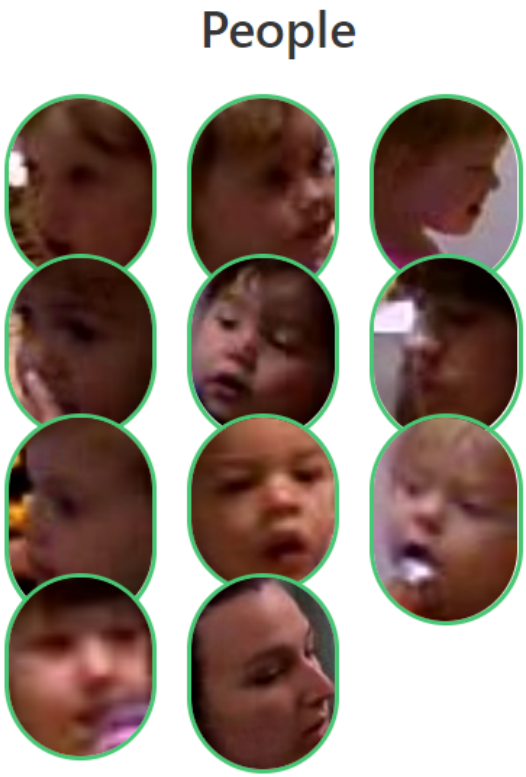

Figure 3. Subclips filter based on students

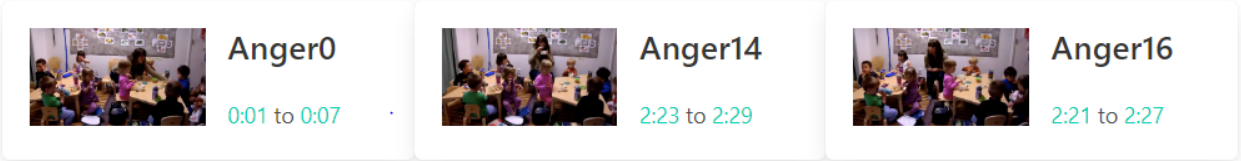

Figure 4. Example of subclips' names

Figure 3 and Figure 4 demonstrate the most significant progress on the UI side towards the goal of the project. In figure 3, the user can click on an avatar and the website will filter all the subclips

containing the subject corresponding to the avatar. In figure 4, we see the descriptive name of the clips. Names consist of emotion attached with numbered intensity such as “Happy5”, “Sad10”, or “Anger2”. This could be considered a “text thumbnail” for the user to navigate the subclips video collection. This is useful in the case of the teacher wanting to isolate their research intention to an individual at a very specific emotion case. However, this is also not sufficient if a teacher wants some kind of overview for the class dynamic in emotions.

While progress has been made, there is little to no data visualization that actually conveys useful information to the user. The key goal of our project initially was to bring data visualization to helping the process of teaching. To achieve this, we want to come up with concrete UI features that are common data visualizations. They should be visually informative, easy to use and understand from the users’ perspective, and most importantly, they should be useful. Basic statistics concepts like mean or standard deviation though common and easy to compute, can bring up useful patterns in the data. We will attempt to visualize basic statistics to convey information to users.

As discussed in section 2.1, understanding the class’ emotion dynamic is essential to teachers. In statistics, there is a common fitting concept called distribution. According to Britannica [3], a distribution describes the probability that a system takes on a specific value or set of values.

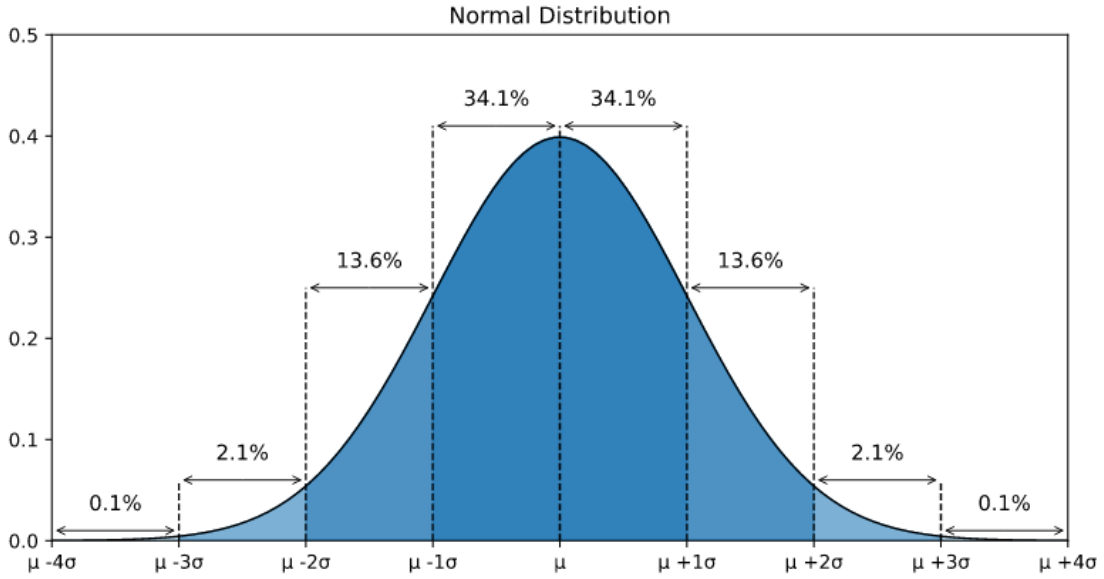

Figure 5. Example of normal distribution [4]

To explain simply, a distribution is useful in this particular case in the sense that it can tell the teacher how many students express a certain level of emotional intensity. This gives a sense of generality about the class dynamic. Most distributions can be effectively presented in the form of traditional bar charts. Here is an example.

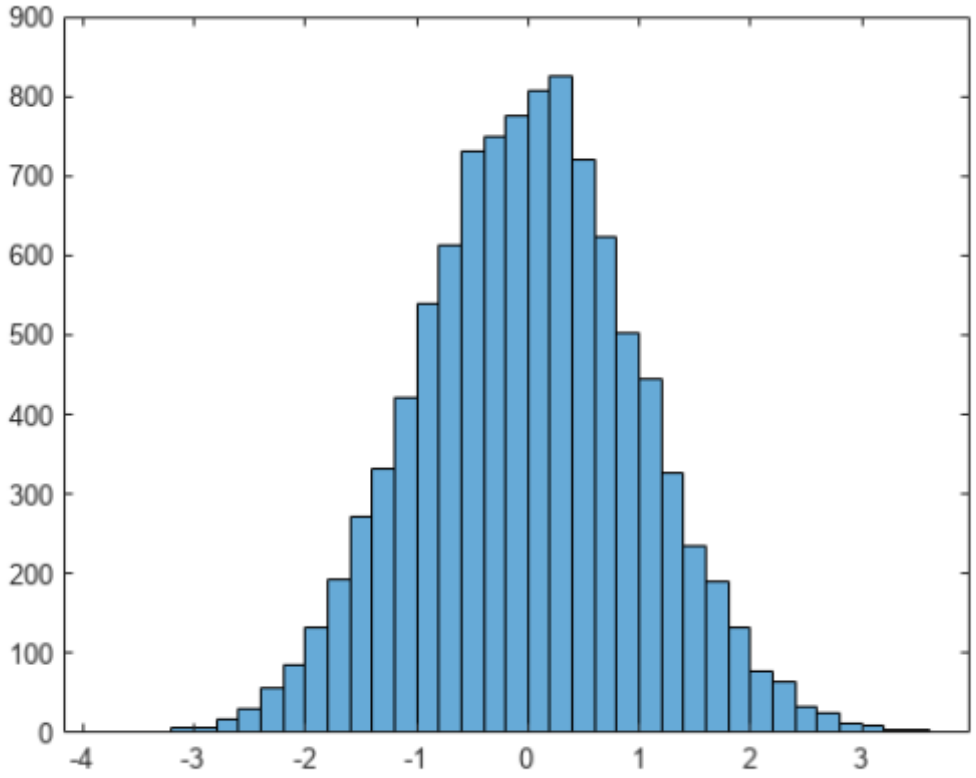

Figure 6. Normal distribution illustrated by MATLAB bar chart function [5]

Another statistical concept that could be useful to the teachers are the outliers. Outliers are extreme cases that have extremely small chances, often less than 1 percent, to happen. In this case, certain extremely high intensity emotions are of great interest by teachers, for the extreme cases especially negative emotions are signals of the source individuals are requiring special attention from the caregiver.

For these reason, the concrete features that we have set out to implement in the existing UI are:

- A graphical representation of the distribution for emotions of the entire class.
- A feature highlighting the emotional outliers amongst the students.

In addition to these, we are also aware that this is an ongoing collaborative effort to develop this tool. Thus, we want the team who works on this project in the future to focus on making real progress on the tool and not spending time trying to understand what happened here in the codebase. In short, we want to keep the code clean and understandable. For this, we also decided to refactor the existing code base. To achieve this, we analyze the code, discuss with our advisor about the possible deficiencies in the code base, and revise them according to the feedback.

3.2 IMPLEMENTATION

Our goal is to have a graphical representation of the distribution for emotions of the entire class. In other words, that would mean counting the amount of times a certain emotion intensity appears in the subclips population. We first started by defining exactly how our visualization looks.

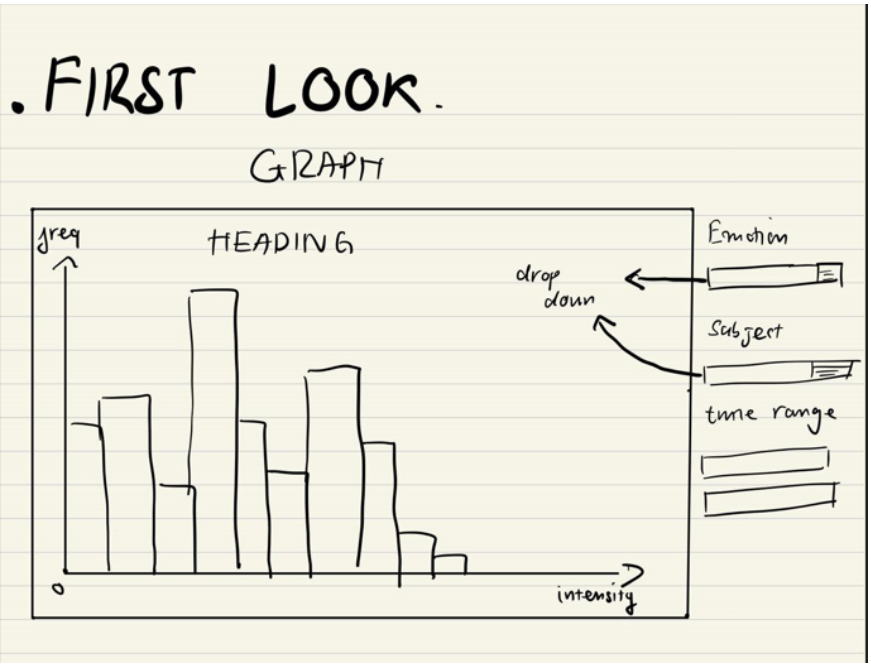

Figure 7. First sketch of the visualization

The visualization includes the bar chart of intensity over frequency. There are also extra features to interact with the visualization including emotion selection, subject selection, and time range. After discussion with our advisor and our available time for this project, we have decided to limit our chart to the most important features which are the bar chart and emotion selection. The user interaction with the graph is simple. When the user chooses an emotion in the dropdown, the bar chart will automatically refresh and display the new distribution graph accordingly. After the priorities are set, we attempt to build

the graph. For ease of demonstration, we will explain the building process according to the user interaction flow.

Firstly, When the user clicks on the emotion dropdown and selects an emotion, the website will take that value and use it as a filter to filter out all the subclips in the collection using their names. The clips are stored as objects of our custom JavaScript class, which stores useful information about the clips including our fields of interest in this case: emotion and intensity. The fields of interest are stored in Java script arrays and are counted as frequency.

```
class Clip {
    constructor() {
        this.name = "Clip Default";
        this.videoPath = "";
        this.timestamp = [];
        this.emotion = "Emotion Default";
        this.emotionValue = 0;
```

Figure 8. Part of "Clip" class' structure

The data are then fed into the bar chart graph default model. Once the data is loaded, the small elements of the bar chart are gradually added: the axes, the bars, and basic statistics of the data being included in the graph . After when all the elements are in place, we added support for the graph to change on the event of value change in emotion selection.

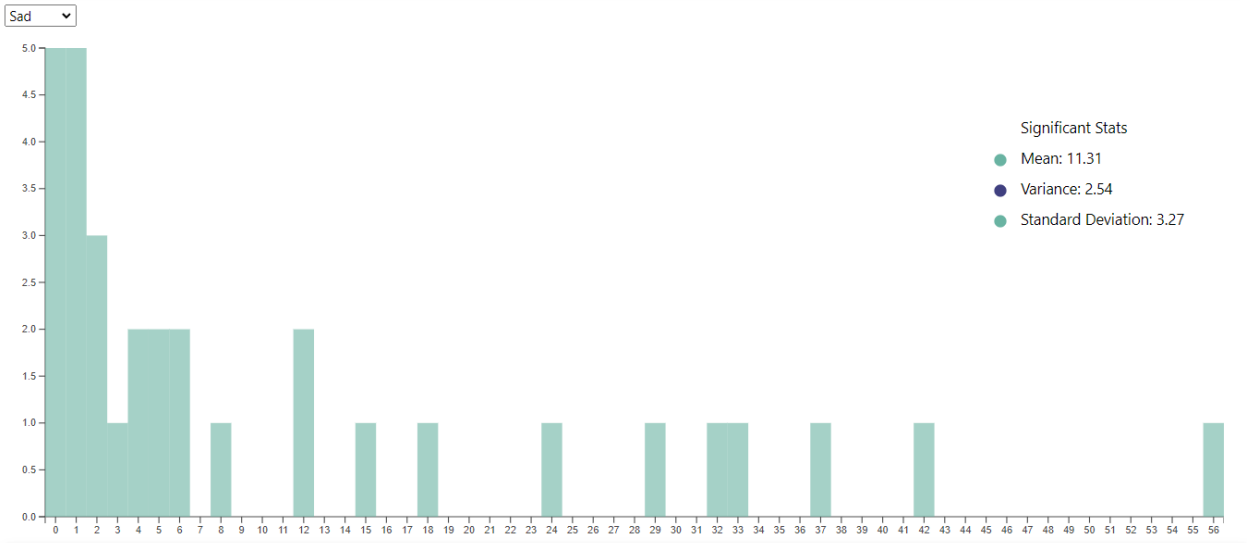

Figure 9. Finished graph of "Sad" emotion distribution

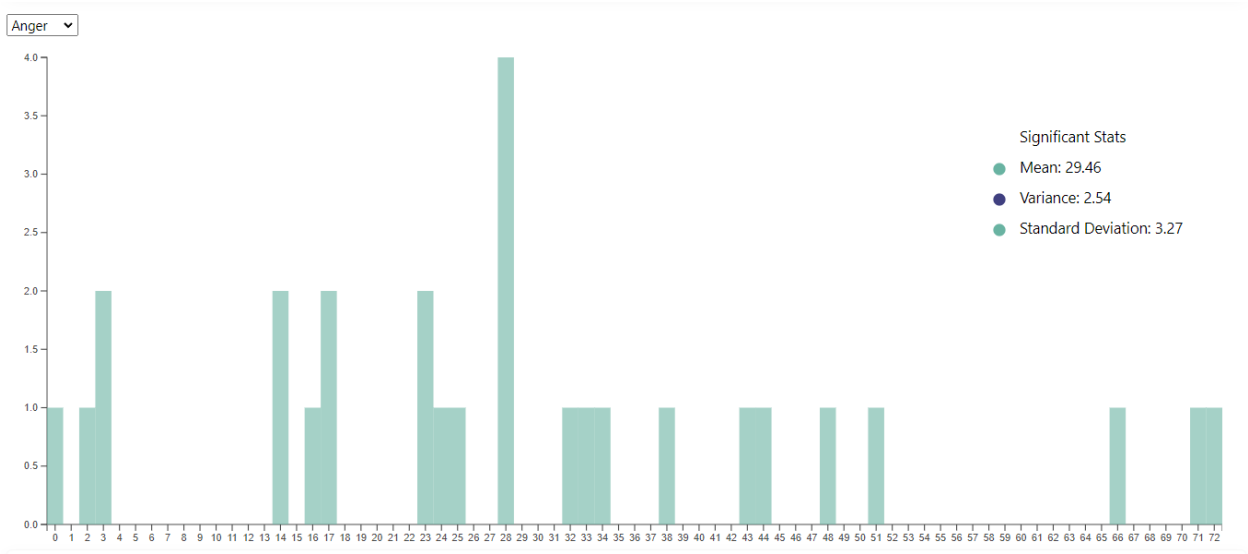

Figure 10. Finished graph of "Angry" emotion distribution

From the statistical standpoint, for a data point to be considered outliers for a distribution, it has to deviate from the rest of the dataset to a "great extent" [9]. Different distributions will have different ways of determining that extent. For example, according to the z-score method, data points in a normal distribution will be considered outliers if they are 3 or more standard deviations away from the mean [10]. In these two examples figure 9 and figure 10, based on the definition, we could observe that there are significant outliers or extreme data points existing in our distribution. However, we realize that a regular kindergarten classes' size typically has less than 30 students, which would mean outliers by statistical definition may not always exist because of small sample size . Hence, we will redefine our outliers to be just the most intense emotions. This redefinition will fit the teachers' needs better as the numeric rating is only a relative stats that are prone to incorrect revaluation from the modeling. It is crucial to remember that this tool will only serve as a tool of suggestion for teachers not as a form of definitive judgment.

For displaying the outliers in our dataset, we decided to pick the top 3 most intense subclips of each emotion and group them into a tab-like feature. Each of the emotions would have its own tab, and the user can click on each tab to view the top most intense reaction for that emotion.

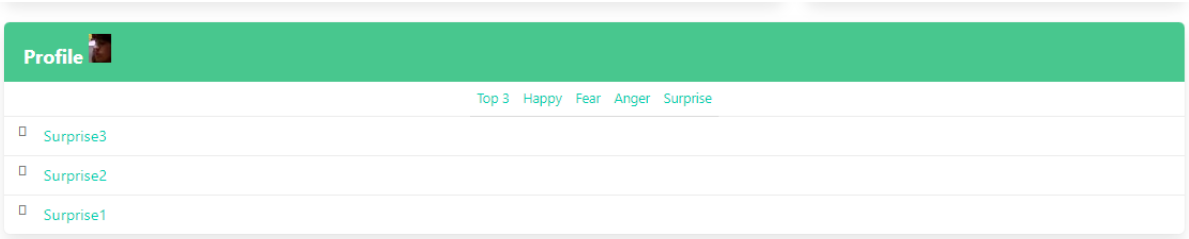

Figure 11. Top 3 Feature

3.3 CODE REFACTORING

To yield the most effective refactoring, we set the goals according to the advisor's feedback on the prototyping nature and the ongoing development plan of the project. Most likely, a team of undergraduate students will take on the task of developing the project further in the future. We reasonably assume that the students will come into the project with basic programming functionality including variables, functions, scope, and so on. Students are exposed to web development (typical file structures, available services) but are not expected to know framework-specific syntaxes. Due to limited time available and the UI-oriented nature of this project, we will primarily focus on refactoring the Java Script file and HTML file.

The refactoring process happens throughout the development process. For ease of demonstration, we will discuss the process based on these three significant timelines:

- Before Implementation: Read, understand the code base, then optimize
- During Implementation: Practicing clean code writing procedure while writing new code
- After Implementation: Revision

Before the implementation, we inspect the code to identify how the previously implemented the user interface. We started by skimming through basic files of web development including HTML, JavaScript, and CSS. The focus is on making the code purpose as transparent as possible. Here are a few prominent examples of what we've done to improve code's transparency.

```
/*
    Helper function for "displayTop3ClipAccordingtoEmotion". Setting Headline for Top 3 region.

    @params: {image} image: image of subject in top 3 region
*/
function setHeadLineTop3Region(image) {
    var currentPerson = findCurrentPersonInTop3Region(image);
    var cln = document.getElementById("area").cloneNode(true);
    document.getElementsByClassName("panel-heading")[0].innerHTML = "Profile: " + currentPerson.name + " ";
    document.getElementsByClassName("panel-heading")[0].appendChild(cln);
}

/*
    Renders Top 3 region

    @params: {Person} currentPerson: person's emotion being considered for top 3
    @params: {String} emotion: emotion being considered
*/

function displayTop3ClipAccordingtoEmotion(currentPerson, emotion) {
    var formattedCurrentPersonName = currentPerson.name.slice(0, 6) + '-' + currentPerson.name.slice(6);

    // filter top 3 that has currentPerson and emotion
    var top3;
    if (emotion == "Top-3") {
        top3 = exampleClips.filter(clip => clip.videoPath.includes(formattedCurrentPersonName));
    } else {
        top3 = exampleClips.filter(clip => clip.name.includes(emotion) && clip.videoPath.includes(formattedCurrentPersonName));
    }
```

Figure 12. Adding comments for undocumented functions in index.js

```
131                         </span>                                 131                         </span>
132                     </p>                                        132                     </p>
133                     <p class="panel-tabs">                      133                     <p class="panel-tabs">
134 -                       <a class="is-active" id="top3">Top 3</a>134 +                       <a id="top-3-button">Top 3</a>
135 -                       <a id="happy-button-navbar"             135 +                       <a id="happy-button">Happy</a>
    onclick="displayTop3ClipAccordingtoEmotion(this)">Happy</a>
136 -                       <a>PlaceHolder</a>                      136 +                       <a id="fear-button">Fear</a>
137 -                       <a>PlaceHolder</a>                      137 +                       <a id="anger-button">Anger</a>
138 -                       <a>PlaceHolder</a>                      138 +                       <a id="surprise-button">Surprise</a>
139                     </p>                                        139                     </p>
```

Figure 13. Edit examples in HTML files. These changes were recorded on our history of commits on GitHub. Code in red is original code, while code in green is edited code. Changes we've made here include renaming HTML elements to better reflect their purpose, and moving events from HTML file to index.js file for unified style of design.

During implementation, we try to add comments appropriately so that the code is informative but not overflowing with unnecessary comments that makes the code harder to read. That's why we go with the approach of dissecting a big function into sub-tasks of length around 10 to 20 lines and adding a short line of comment on each of those sub-tasks. The function for rendering the visualization is a great example. Once development is done, we start revising the code by deleting unused function, variable, and excess comments. We also use the refactoring feature built-in Visual Studio Code to rename variables where variable name are not informative enough.

```
// Filter videos with emotion similar to "keyword" parameter
var svg = d3.select('svg');
var exampleClips0 = [];
for (let i = 0; i < exampleClips.length; i++) {
    var emotion = exampleClips[i].emotion;
    if (emotion.includes(keyWord)) {
        exampleClips0.push(exampleClips[i]);
    }
}

var freq = [];
var check = false;


// Adding frequncy for each intensity level
for (let i = 0; i < exampleClips0.length; i++) {
    const key = parseInt(exampleClips0[i].emotionValue);
    check = false;
    for (let j = 0; j < freq.length; j++) {
        if (freq[j].value == key) { // key already exist
            check = true;
            freq[j].count += 1
        }
    }
```

Figure 14. Example for our refactoring method

Chapter 4. RESULTS AND RECOMMENDATION

4.1 RESULTS

The prototype of the UI has been improved significantly in terms of actual visualization showing on the page. The user could interact with the graph to see different distributions of each emotion in their class dynamics.

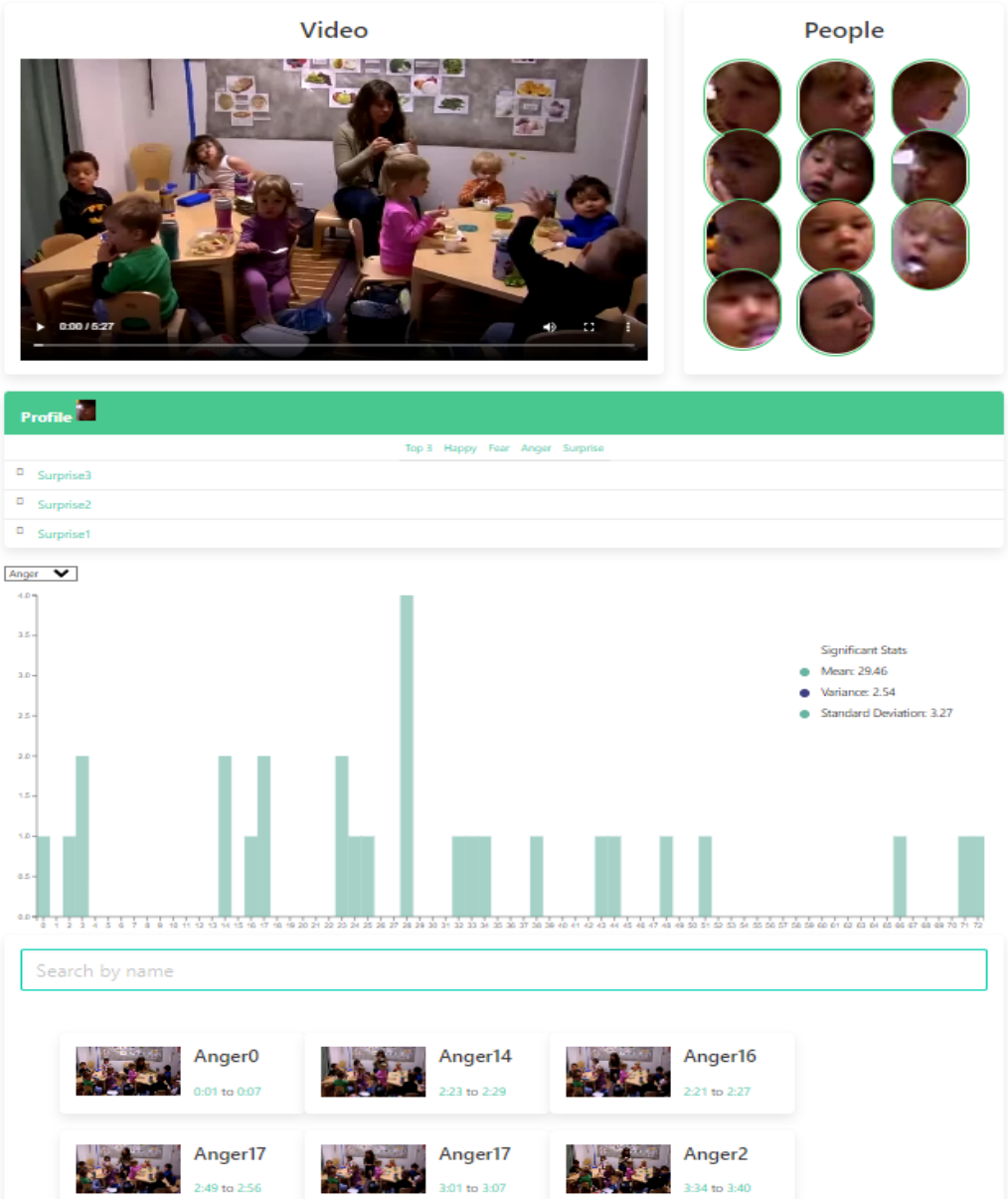

Figure 12. UI Overall look after finished development

4.2 RECOMMENDATIONS

The project is still in early stage of development and experimentation, thus room for improvement is ample. We have multiple plans to further develop COILS but haven't had the chance to implement them. For the bar chart, we are aware that teachers aren't always used to the terms of statistics such as variance, and standard deviation or different distributions. There could be a small feature on the UI page that explains to the user what those statistics mean. For example, we could add an explanation to the terms "standard deviation" and explain how small standard deviation of intensity would mean generally there are small differences between how the students express a particular emotions in the considered data. There are also improvements we could have made on the interactivity of the graph. For instance, the bars in the chart could be hovered over by the mouse, and information related to that bar would pop up as a small pop-up window. The possibility of the information on that pop-up window is huge, but one idea we have come up with is to list the names of the subclips corresponding to that intensity. With this, the teacher would have an easier time researching the students of their concerns if they were to use the bar chart feature to isolate the cases that would require their attention more.

Chapter 5. CONCLUSION

Over the course of the project, we have learned a lot about data visualizations. We learned that for a data visualization to be successful, we have to start from defining clear goals and expectations we have for the project. We can achieve this by a lot of research on the needs of the users who would benefit from the final product and reflect on our budget of time and technical skills. We were also exposed to new technology involving data visualization on a web application, particularly D3 including what web framework it supports and what visualization techniques it offers. The visualization part of this project is considered one of the more basic types of graphs that D3 supports, but it could be challenging for students who could be getting into D3 and Javascript for the first time. Thus, it is important to be patient and keep in mind that the first few visuals on the product might not be as expected.

6. BIBLIOGRAPHY

1. Cloud Education, IBM. “What Is Data Visualization?” IBM, IBM, 10 Feb. 2021, https://www.ibm.com/cloud/learn/data-visualization.
2. M. Bostock. (2012) Data driven documents. [Online]. Available: https://d3js.org/
3. Zeidan, Adam. “Distribution Function.” Encyclopædia Britannica, Encyclopædia Britannica, Inc., 16 June 2005, https://www.britannica.com/science/distribution-function.
4. https://www.w3schools.com/statistics/statistics_normal_distribution.php
5. Matlab. Figure Histogram of Vector. “Histogram.” Histogram Plot - MATLAB, 8 Oct. 2014, https://www.mathworks.com/help/matlab/ref/matlab.graphics.chart.primitive.histogram.html.
6. J. Lin and H. Zhang, "Data Structure Visualization on the Web," 2020 IEEE International Conference on Big Data (Big Data), 2020, pp. 3272-3279, doi: 10.1109/BigData50022.2020.9378249.
7. Y. Miao, "Research on Kindergarten Science Field Teaching Based on Big Data Analysis," 2019 International Conference on Communications, Information System and Computer Engineering (CISCE), 2019, pp. 592-594, doi: 10.1109/CISCE.2019.00138.
8. M. M. Maw and D. K. K. Oo, "Analysis of Refactored Code Vulnerability and Assessment," 2020 IEEE 9th Global Conference on Consumer Electronics (GCCE), 2020, pp. 272-273, doi: 10.1109/GCCE50665.2020.9291710.
9. E. Muller, I. Assent, U. Steinhausen and T. Seidl, "OutRank: ranking outliers in high dimensional data," 2008 IEEE 24th International Conference on Data Engineering Workshop, 2008, pp. 600-603, doi: 10.1109/ICDEW.2008.4498387.
10. F. Yoseph, M. Heikkilä and D. Howard, "Outliers Identification Model in Point-of-Sales Data Using Enhanced Normal Distribution Method," 2019 International Conference on Machine Learning and Data Engineering (iCMLDE), 2019, pp. 72-78, doi: 10.1109/iCMLDE49015.2019.00024.
11. Association, National Education. “Early Childhood Education.” NEA, 12 Feb. 2021, https://www.nea.org/advocating-for-change/action-center/our-issues/early-childhood-education.