Final Exam
Voice Activation System
(15% of Grade)

Instructions

- Read each section carefully.
- Provide thorough explanations for your design choices.
- **Over-communicate**: Explain your thought process in detail
- Provide images for clarity when needed
- Pay attention to precision and accuracy.
- Questions are ambiguous on purpose – think critically and propose ideas.

Objective

In this final exam you will build a voice activation system using machine learning. The system will use your voice to turn on a led, and read and display the current temperature. You will utilize the following sensors:

- Analog Sound Sensor
- RGB LED
- LM35 Temperature
- 4-Digit LED Display

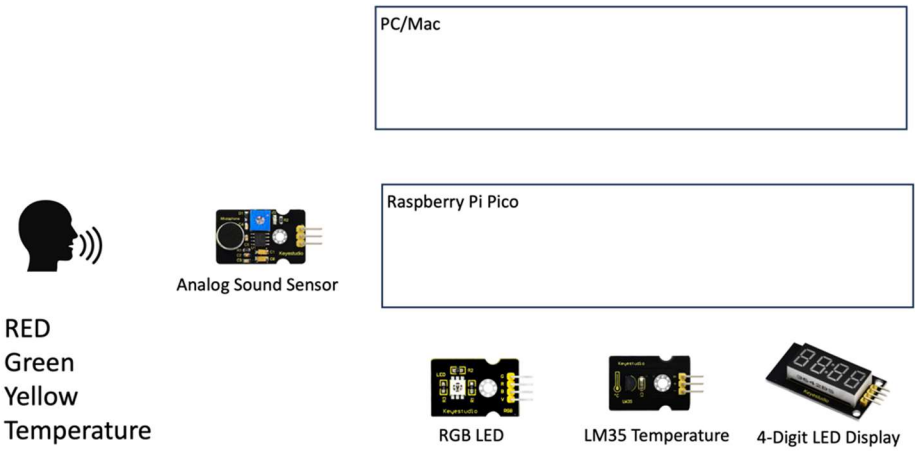


Figure 1.1 System Overview

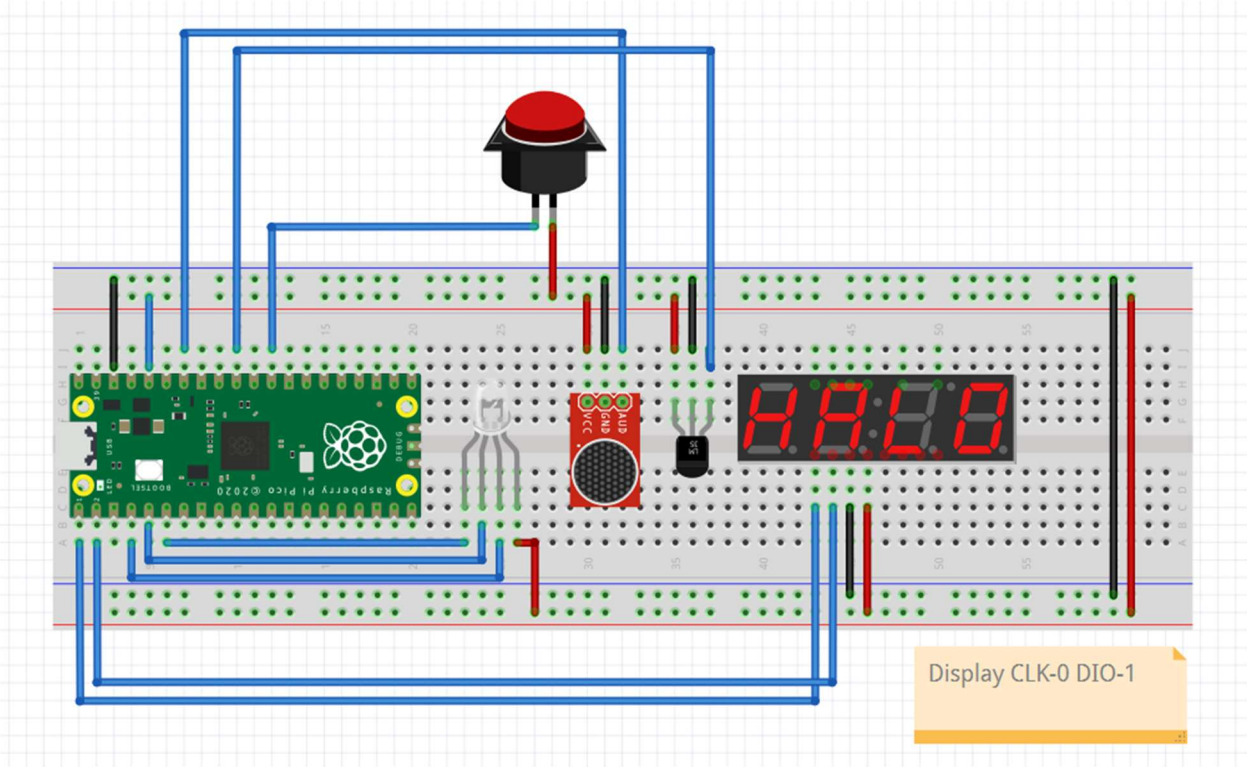
System Behavior

The system will work as follows:

- You will build a machine learning model to classify voice commands
- Voice Activation – the user should be able to say the following commands with their voice: “RED”, “Green”, “Yellow”, “Temperature”
- The system will then turn on the LED to the appropriate color based on the voice command: “RED”, “Green”, “Yellow”
- For the “Temperature” voice command – the system will read the current temperature and display it on the 4-digit LED display

Problem 1 - A

Create a fritzing diagram showing the connections of the sensors to the Raspberry Pi Pico board.



Problem 1 – B

Fill in the box from Figure 1 System Overview above for the Raspberry Pi Pico explaining what needs to happen.

Problem 1 – C

Fill in the box from Figure 1 System Overview above for the PC/Mac explaining what needs to happen.

PC

- Import machine learning model.
- Wait for data over serial.
- Receive data array and store in np array.
- Perform statistics on data.
    - Standard Deviation
    - Kurtosis
    - Skewness
- Use statistics tuple to predict audio sample word.
- Send prediction back over serial to Pico.

Raspberry Pi Pico

- On button click
    - Collect Audio Sample with a sample rate for 1 second.
        - Capture 75 Audio Samples at 441000 Hz for each sample. Sum together
        - Store samples in array.
- Send audio array to computer over Serial Bus.
- Wait for computer to predict word.
- If word is a color,
    - Turn on corresponding color on RGB LED.
- If word is not a color
    - Turn off LED.
    - Read Temperature .
    - Display temp on 7-seg display.
- Repeat on button press

Problem 2

Collect twenty samples of voice data for each voice command.

Upload your data by **Friday April 26, 2024 by Midnight EST** to the following shared folder:

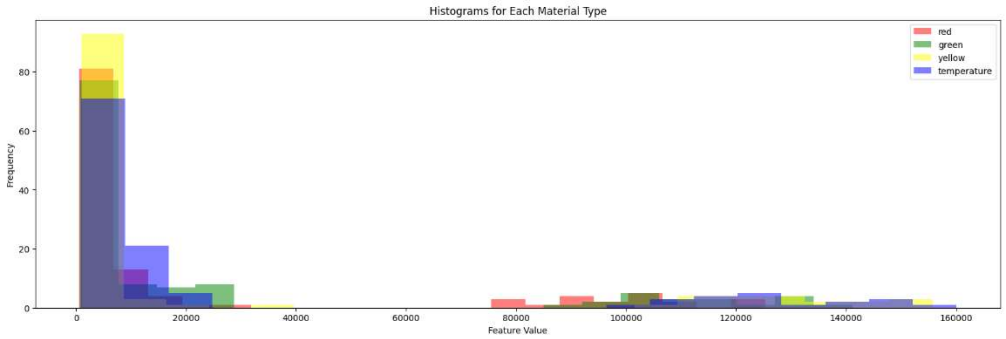
- Final Exam - OneDrive (sharepoint.com)
- **If data is turned in late you will lose 25 points**

When collecting data you will say each command. You will then have a set of values for each command. You will need to average the data such that there is a single value for each command. Thus, your total data set will have 80 samples or 80 rows of data. Upload the data in .csv format.

Problem 3 - A

Build a machine learning model to classify each voice command. Explain how you split the data for testing and training (80/20 or k-fold). Why is one better than the other? Compare your model against other models and provide a comparison table of metrics: precision, recall, f1-score, and accuracy.

Previously, I took data over a period of time and summed it together, using a single value to distinguish between words however, this method did not provide much clustering and only had a max of 40% accuracy when training. Below is the histogram to show distribution.

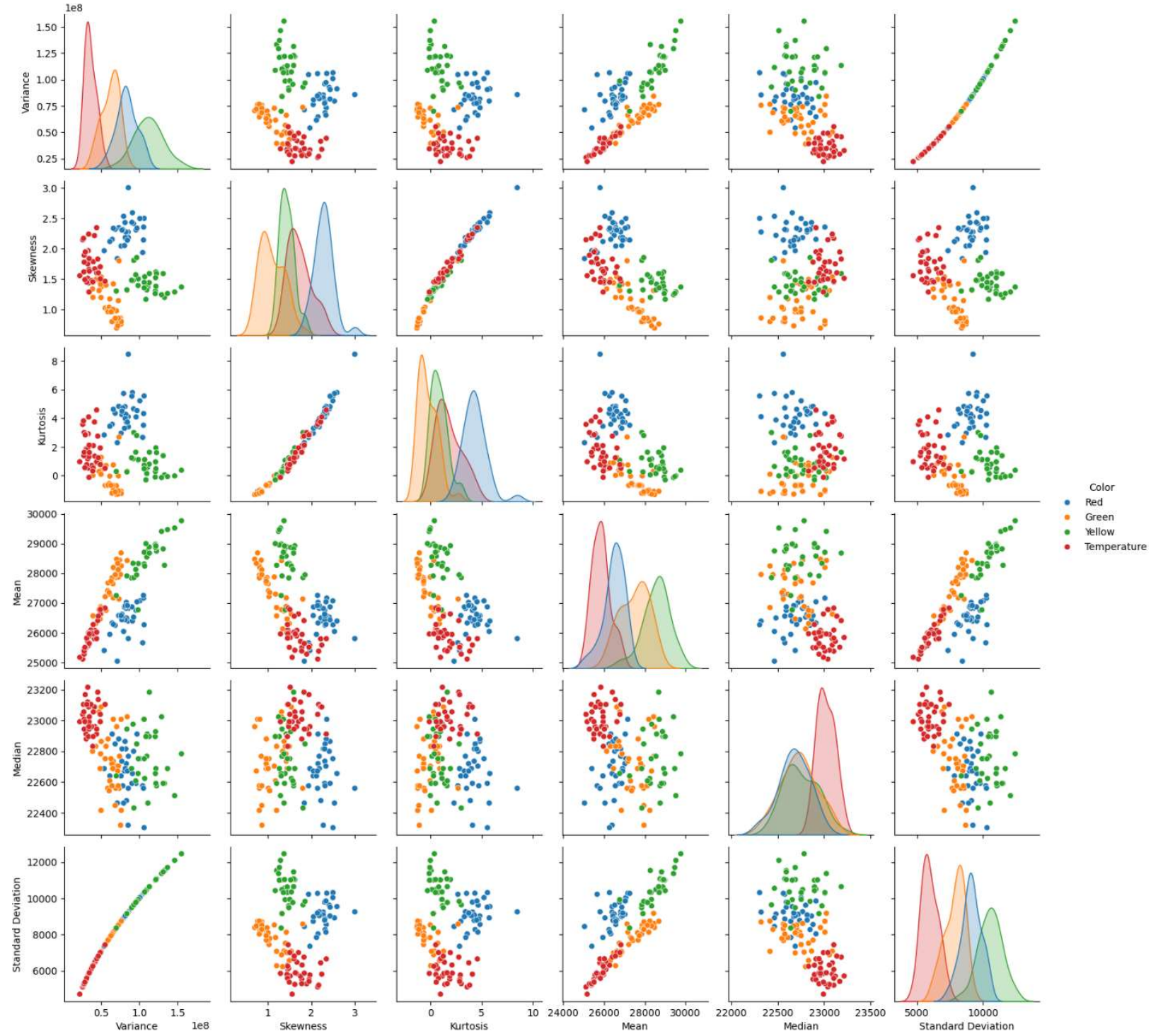


What I ended up doing was adding a button that would collect summed data at a specific sample rate then add it to an array for some period of time. That way each word had the same data time and amount collected. (See pair plot on next page for details)

What I noticed was if I summed all the values together they would be indistinguishable, but if I took that array of collected samples and looked at the statistics of them such as skew and standard deviation it provided features that could be used! This is because while the sum of the volume was the same, the spikes at which the "loudest" part of the word was different for each one if pronounced correctly.

Doing all this I was able to get a trained model of 92% using three statistical features but didn't use any of the other student's data since I am using a different model. However, I had 4 members of my household collect voice data so I would have a range of inputs as I would if I used other students data.

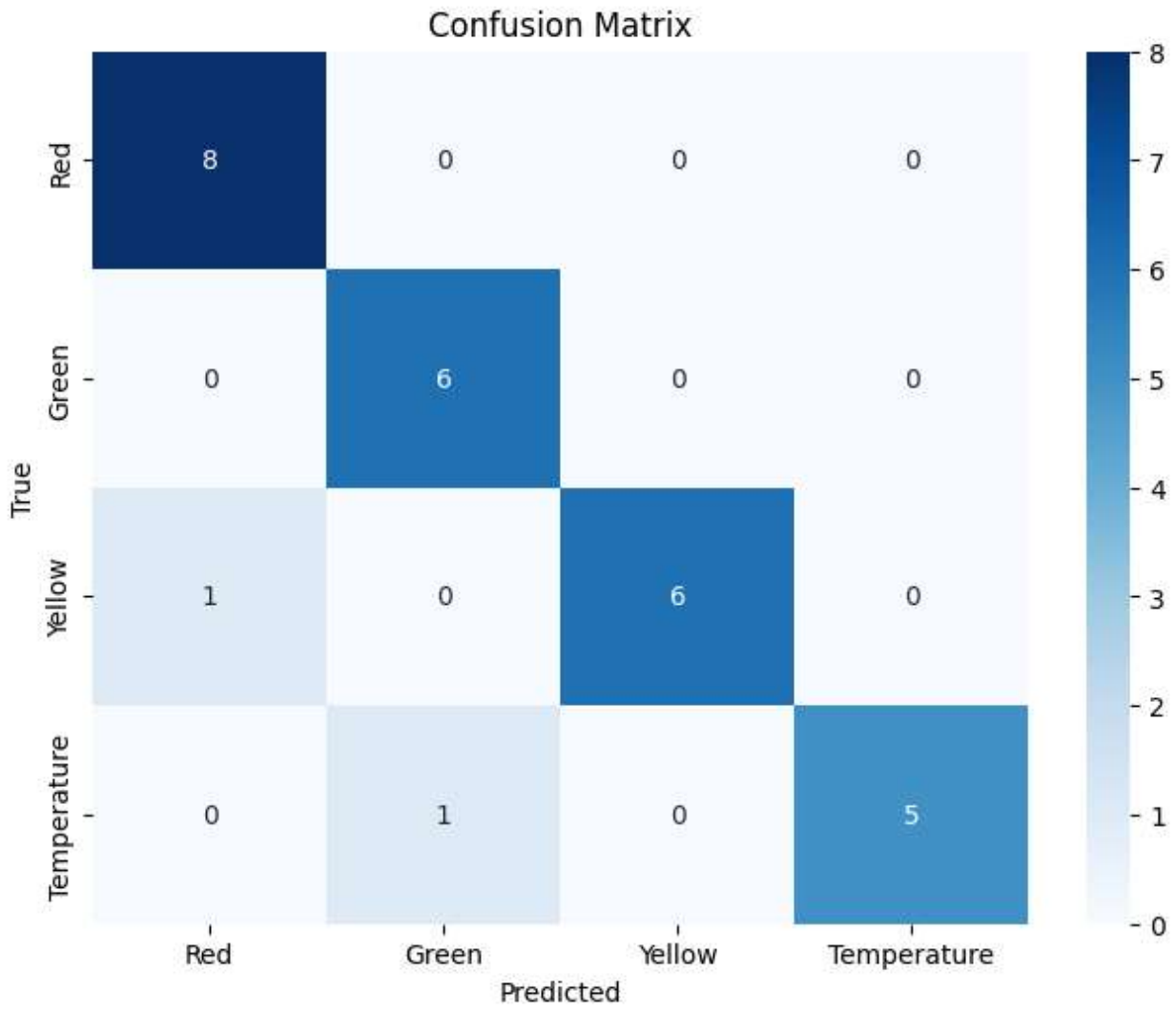
	Model	Accuracy	Precision	Recall	F1 Score
0	SVC(kernel='linear'), ALL	0.888889	0.911523	0.888889	0.887413
1	SVC(), ALL	0.666667	0.718230	0.666667	0.674504
2	KNeighborsClassifier(n_neighbors=10), ALL	0.703704	0.748148	0.703704	0.702469
3	DecisionTreeClassifier(max_depth=5, random_sta...	0.814815	0.878189	0.814815	0.809731
4	GaussianNB(), ALL	0.925926	0.935332	0.925926	0.925332
5	RandomForestClassifier(max_depth=5, max_featur...	0.851852	0.893004	0.851852	0.850472
6	MLPClassifier(alpha=1, max_iter=1000, random_s...	0.296296	0.087791	0.296296	0.135450
7	AdaBoostClassifier(algorithm='SAMME', random_s...	0.629630	0.554497	0.629630	0.544856



Overall, the Gaussian model had the highest accuracy and was used for the rest of the project. This is probably because Gaussian is robust to noisy data and is less sensitive to irrelevant features.

Problem 3 – B

For your best selected model (high performance) based off the metrics you used for evaluation, plot the confusion matrix below.



Problem 4

Build the system with your model. Provide your code and a video demo of the system working.

- Upload all files to Canvas (.py and exported model)
- Include a README file with instructions for running the system
- Upload your video demo of the system working to Canvas