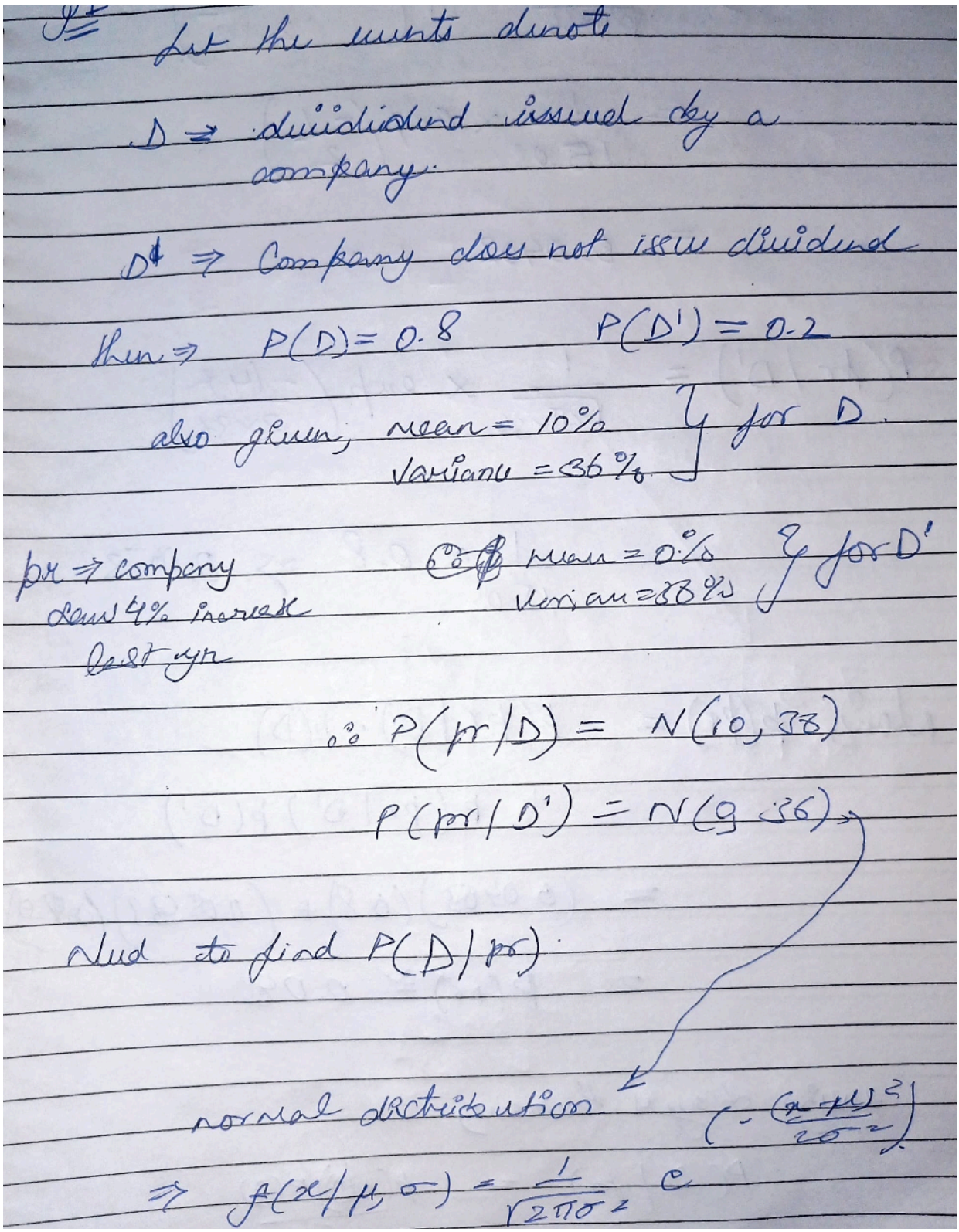
CSE343
Machine Learning
Monsoon 2024

Report for Assignment 2

Shamik Sinha - 2022468

Section A

part a.



$$\therefore P(pr|D) = \frac{1}{\sqrt{2\pi}\times 36} \times \exp\left(\frac{-(4-10)^2}{2\times 36}\right)$$

$$= \frac{1}{15.03} \propto \exp\left(\frac{-1}{2}\right)$$

$$= 0.0403$$

$$P(pr|D') = \frac{1}{\sqrt{2\pi}\times 36} \times \exp\left(\frac{-(4-3)^2}{2\times 36}\right)$$

$$\frac{1}{15.03} \propto 0.8 \Rightarrow 0.0533$$

Now, $P(pr) = P(pr|D)\cdot P(D) + P(pr|D')P(D')$

$$= (0.0403)(0.8) + (0.0533)(0.20)$$

$$= P(pr) \simeq 0.043$$

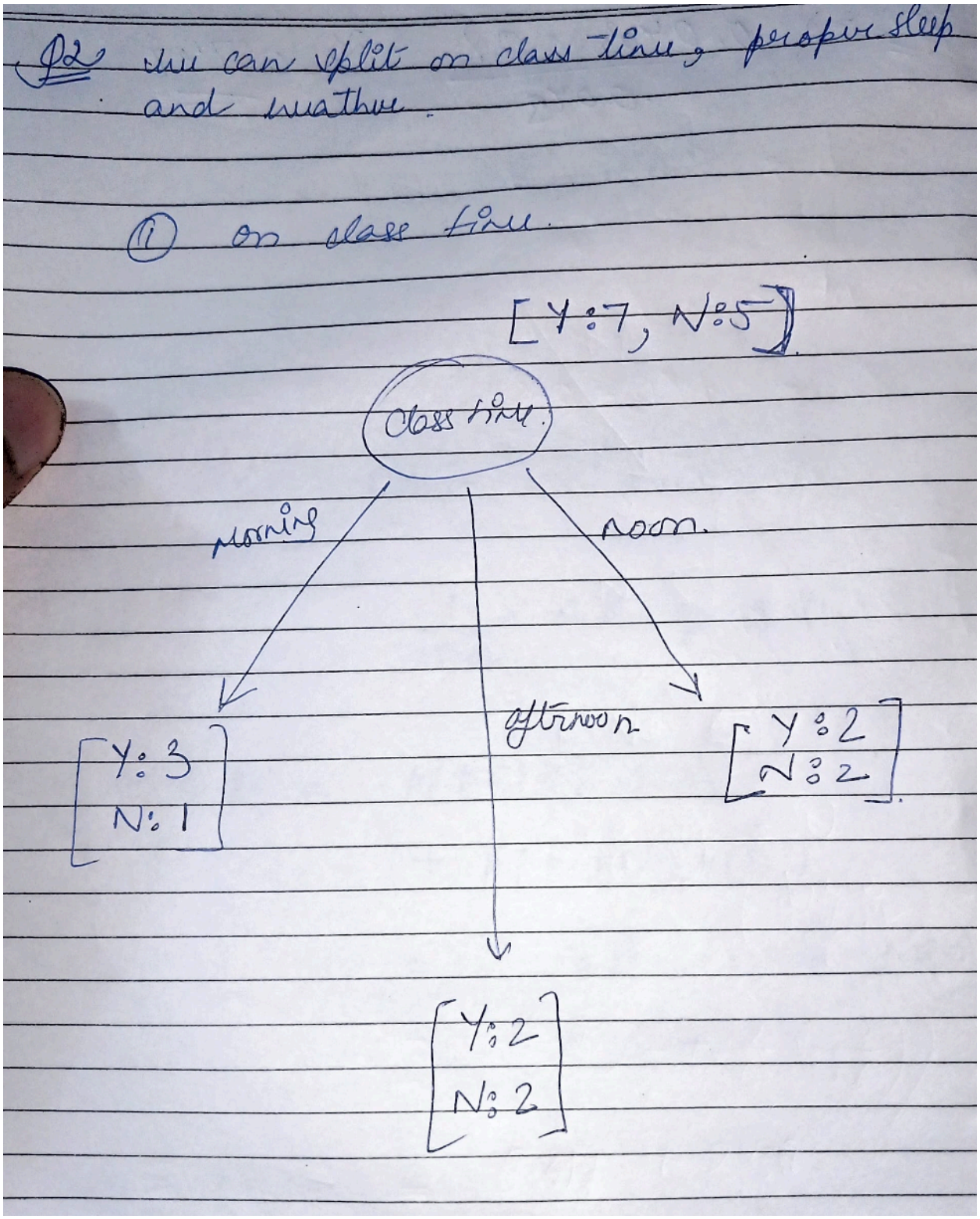
using Bayes theorem,

$$P(D|pr) = \frac{P(pr|D)P(D)}{P(pr)}$$

Date ..../..../.....

$$\Rightarrow \frac{0.0403 \times 0.8}{0.043} \approx 0.747$$

part b.


Q2 we can split on class time, proper sleep and weather.
① on class time.
[Y:7, N:5]
Class time.
morning
noon.
afternoon
[Y:3 N:1]
[Y:2 N:2]
[Y:2 N:2]

IG = E(parent) − weighted avg of E(children).

$$E(\text{parent}) = -\left(\frac{7}{12}\left(\log\frac{7}{12}\right) + \frac{5}{12}\log\frac{5}{12}\right)$$

$$\Rightarrow -(-0.453 + -0.526)$$

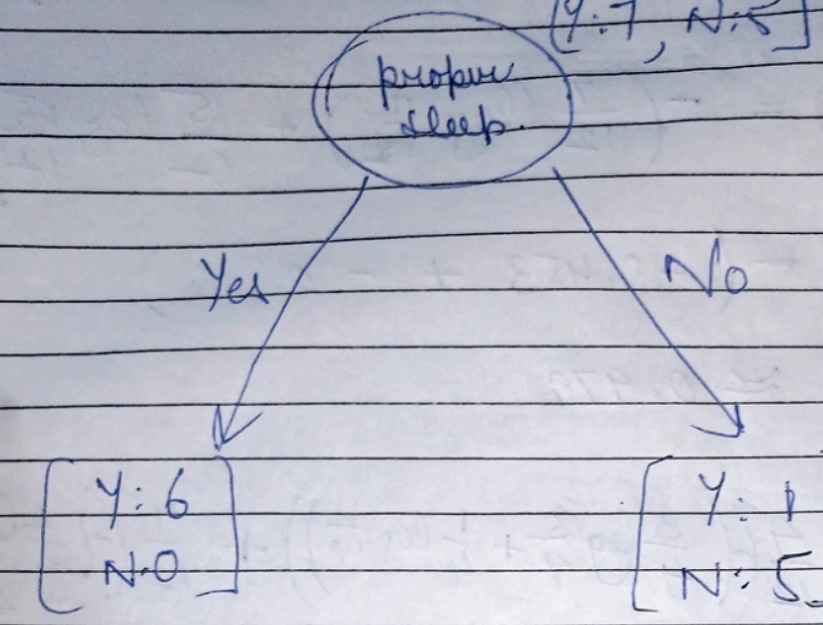
$$\Rightarrow \simeq 0.979$$

weighted avg. $\Rightarrow \frac{4}{12}\left(-\left(\frac{3}{4}\log\frac{3}{4} + \frac{1}{4}\log\frac{1}{4}\right)\right) + \frac{4}{12}\left(-\left(\frac{2}{2}\log\frac{2}{4} + \frac{2}{2}\log\frac{2}{4}\right)\right) + \frac{4}{12}\left(-\left(\frac{2}{4}\log\frac{2}{4} + \frac{2}{4}\log\frac{2}{4}\right)\right)$

$$\Rightarrow \frac{4}{12}(0.311 + 0.5) \Rightarrow 0.27 + 0.33 + 0.53$$

$$\therefore \boxed{IG = \quad} \Rightarrow 0.042$$

(2) on bad propure sleep



E(parent) = 0.979

Weighted Avg = $\frac{6}{12}\left(-\left(\frac{6}{6}\log\frac{6}{6}+\frac{0}{6}\log 0\right)\right) = 0$ (pure split)

$+\frac{6}{12}\left(-\left(\frac{5}{6}\log\frac{5}{6}+\frac{1}{6}\log\frac{1}{6}\right)\right)$

0.219 + 0.430

$\frac{6}{12}(0.649)$

2) 0.324

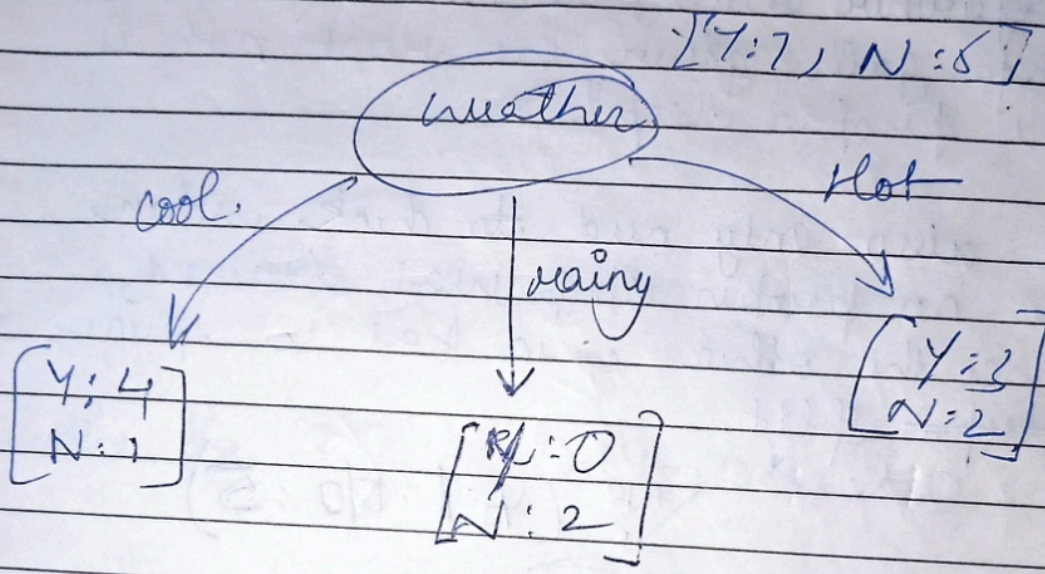
Spiral

Date ..../..../.....

$\Rightarrow IG_T = 0.979 - 0.824$

$= 0.155$

(2) On weather.



$E(parent) = 0.979$

$\Rightarrow \frac{5}{12}\left(-\frac{4}{5}\log\frac{4}{5} + \frac{1}{5}\log\frac{1}{5}\right) + 0 + \frac{5}{12}\left(\frac{3}{5}\log\frac{3}{5} + \frac{2}{5}\log\frac{2}{5}\right)$

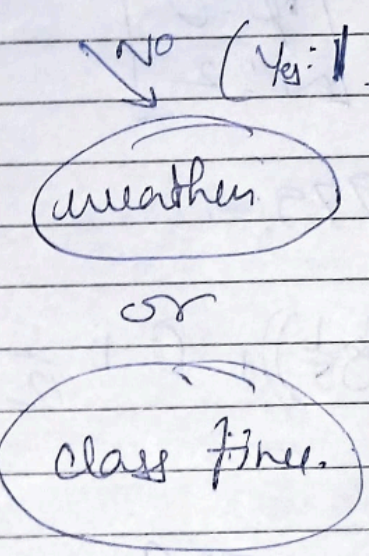
$IG \Rightarrow 0.979 - 0.749 \Rightarrow 0.26$

so we get the highest info gain in weather proper sleep

2) Therefore we split at the root node as weather proper sleep

for further levels, we need to find the Info gain given the root not is "bad proper sleep".

we also only need to check when the no proper sleep was obtained, since the other case has a pure ~~sleep~~ split.



E(parent) = 0.64

Date ..../..../.....

Weighted Avg. ① weather

$= \frac{2}{6}(1) + \frac{2}{6}(0) + \frac{2}{6}(0)$

$\Rightarrow$ 0.33

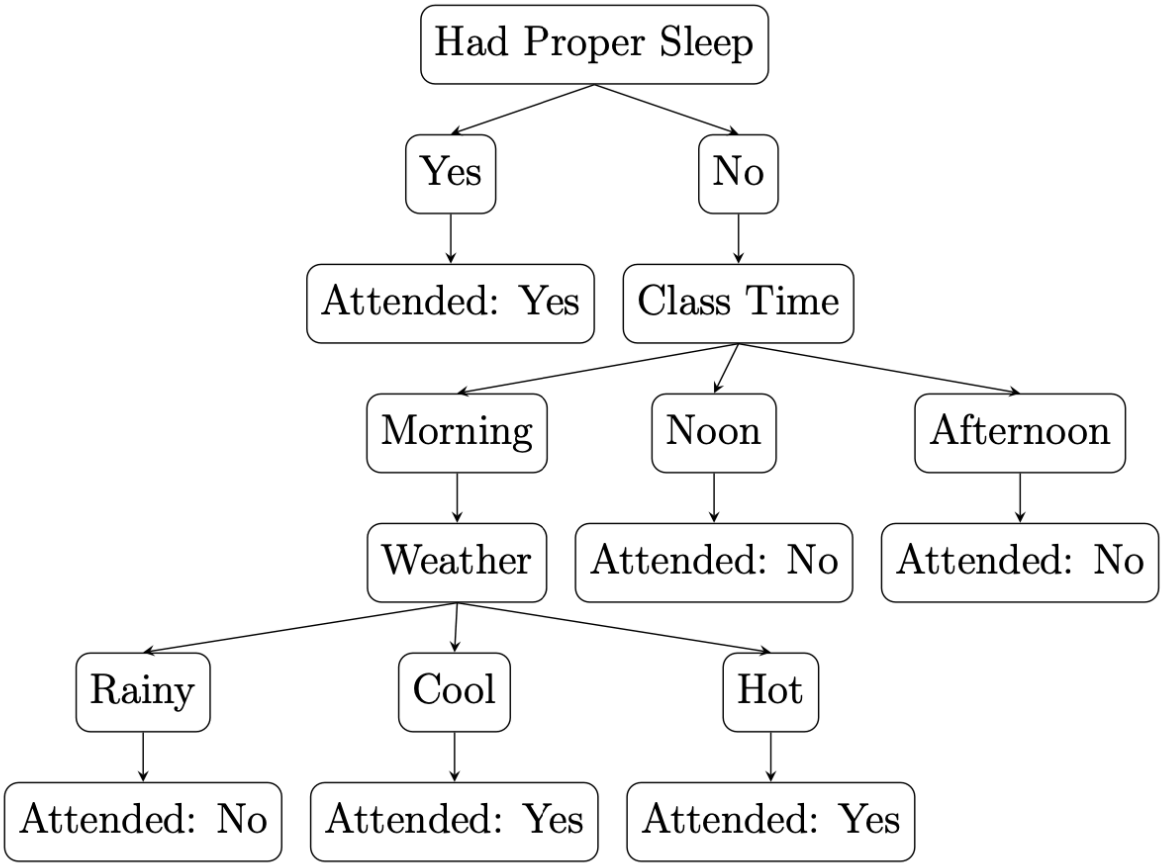
② class time

$= \left[\frac{2}{6}(1) + 0 + \frac{2}{6}(0)\right]$

$\Rightarrow \left(\frac{1}{3}\right) \Rightarrow 0.33$

Both the $IG_s = 0.64 - 0.33 = 0.31$

are the same hence we may use any to split the nodes further.


Had Proper Sleep
Yes
No
Attended: Yes
Class Time
Morning
Noon
Afternoon
Weather
Attended: No
Attended: No
Rainy
Cool
Hot
Attended: No
Attended: Yes
Attended: Yes

part c.

We begin with the standard Perceptron algorithm, which updates the weight vector w_t whenever it makes a mistake on an example x. Our goal is to bound the number of updates.

Step 1: Increase in dot product. Every time a mistake is made, the dot product $w_t \cdot w^*$ increases by at least γ. Specifically, we claim that:

$$w_{t+1} \cdot w^* \geq w_t \cdot w^* + \gamma$$

This is because, for any correct update:
- If the mistake is on a positive example x, we update $w_{t+1} = w_t + x$. Thus:

$$w_{t+1} \cdot w^* = (w_t + x) \cdot w^* = w_t \cdot w^* + x \cdot w^* \geq w_t \cdot w^* + \gamma$$

- Similarly, if the mistake is on a negative example, we update $w_{t+1} = w_t - x$, which also increases the dot product by at least γ.

Proof: if $\mathbf{x}$ was a positive example, then we get $\mathbf{w}_{t+1} \cdot \mathbf{w}^* = (\mathbf{w}_t + \mathbf{x}) \cdot \mathbf{w}^* = \mathbf{w}_t \cdot \mathbf{w}^* + \mathbf{x} \cdot \mathbf{w}^* \geq \mathbf{w}_t \cdot \mathbf{w}^* + \gamma$ (by definition of γ). Similarly, if $\mathbf{x}$ was a negative example, we get $(\mathbf{w}_t - \mathbf{x}) \cdot \mathbf{w}^* = \mathbf{w}_t \cdot \mathbf{w}^* - \mathbf{x} \cdot \mathbf{w}^* \geq \mathbf{w}_t \cdot \mathbf{w}^* + \gamma$.

[1][2][3]

Step 2: Bounding the norm of w_t. After each update, the norm $\|w_t\|$ increases by at most 1. Specifically, we claim that:

$$\|w_{t+1}\|^2 \leq \|w_t\|^2 + 1$$

This follows from the update rule and the fact that each example x is normalized to have a Euclidean length of 1:
- If $w_{t+1} = w_t + x$, then:

$$\|w_{t+1}\|^2 = \|w_t + x\|^2 = \|w_t\|^2 + 2(w_t \cdot x) + \|x\|^2 \leq \|w_t\|^2 + 1$$

- The same holds if the update is $w_{t+1} = w_t - x$.

Step 3: Final bound on M. From Step 1, after M mistakes, we have:

$$w_M \cdot w^* \geq M\gamma$$

From Step 2, after M mistakes, we have:

$$\|w_M\|^2 \leq M$$

Proof: if $\mathbf{x}$ was a positive example, we get $||\mathbf{w}_t + \mathbf{x}||^2 = ||\mathbf{w}_t||^2 + 2\mathbf{w}_t \cdot \mathbf{x} + ||\mathbf{x}||^2$. This is less than $||\mathbf{w}_t||^2 + 1$ because $\mathbf{w}_t \cdot \mathbf{x}$ is negative (remember, we made a mistake on $\mathbf{x}$). Same thing (flipping signs) if $\mathbf{x}$ was negative but we predicted positive.

[1][3]

Now, for our special type of Perceptron algorithm, we follow a similar procedure as we did in the case of the normal perception with slight changes.

1. *Update Process and Increase in Dot Product:* Each update of the weight vector w_t increases the dot product $w_t \cdot w^*$ by at least γ. After M mistakes, we have:

$$w_M \cdot w^* \geq M\gamma$$

2. *Growth of the Norm of w_t:* For the original Perceptron, we had the inequality:

$$\|w_{t+1}\|^2 \leq \|w_t\|^2 + 1$$

For the Margin Perceptron, since we update even when the margin is violated (i.e., less than $\frac{\gamma}{2}$), we get the bound:

$$\|w_{t+1}\| \leq \|w_t\| + \frac{1}{2\|w_t\|} + \frac{\gamma}{2}$$

[1]

3. *Bounding the Norm After M Updates:* Initially, $\|w_1\| = 0$. If $\|w_t\| \geq \frac{2}{\gamma}$, then:

$$\|w_{t+1}\| \leq \|w_t\| + \frac{3\gamma}{4}$$

After M updates, we get:

$$\|w_{M+1}\| \leq \frac{2}{\gamma} + \frac{3M\gamma}{4}$$

4. *Solving for the Maximum Number of Mistakes:* Using the fact that $w_M \cdot w^* \geq M\gamma$ and applying the Cauchy-Schwarz inequality $w_M \cdot w^* \leq \|w_M\|$, we get:

$$M\gamma \leq \frac{2}{\gamma} + \frac{3M\gamma}{4}$$

Rearranging and solving for M:

$$M - \frac{3M}{4} \leq \frac{2}{\gamma^2}$$

Simplifying:

$$\frac{M}{4} \leq \frac{2}{\gamma^2}$$

Finally:

$$M \leq \frac{8}{\gamma^2}$$

Thus, the number of updates (including margin mistakes) made by the Margin Perceptron algorithm is at most $\frac{8}{\gamma^2}$.

Proof for the case where the margin threshold is replaced by $(1-\epsilon)\gamma$:
We extend the analysis of the margin-based Perceptron algorithm by considering updates made on examples for which the margin is less than $(1-\epsilon)\gamma$.

1. *Increase in dot product:* After each update, the dot product $w_t \cdot w^*$ increases by at least $(1-\epsilon)\gamma$. Thus, we have:

$$w_{t+1} \cdot w^* \geq w_t \cdot w^* + (1-\epsilon)\gamma$$

2. *Bounding the norm of w_t:* After M updates, the norm $\|w_M\|^2$ satisfies:

$$\|w_M\|^2 \leq M$$

3. *Cauchy-Schwarz inequality:* From the Cauchy-Schwarz inequality, we know:

$$M(1-\epsilon)\gamma \leq \|w_M\| \leq \sqrt{M}$$

Squaring both sides gives:

$$M^2(1-\epsilon)^2\gamma^2 \leq M$$

Dividing by M yields:

$$M \leq \frac{1}{(1-\epsilon)^2\gamma^2}$$

Therefore, the number of updates is at most $\frac{1}{(1-\epsilon)^2\gamma^2}$.

As we can see, the bounds obtained are polynomial in $\frac{1}{\epsilon\gamma}$

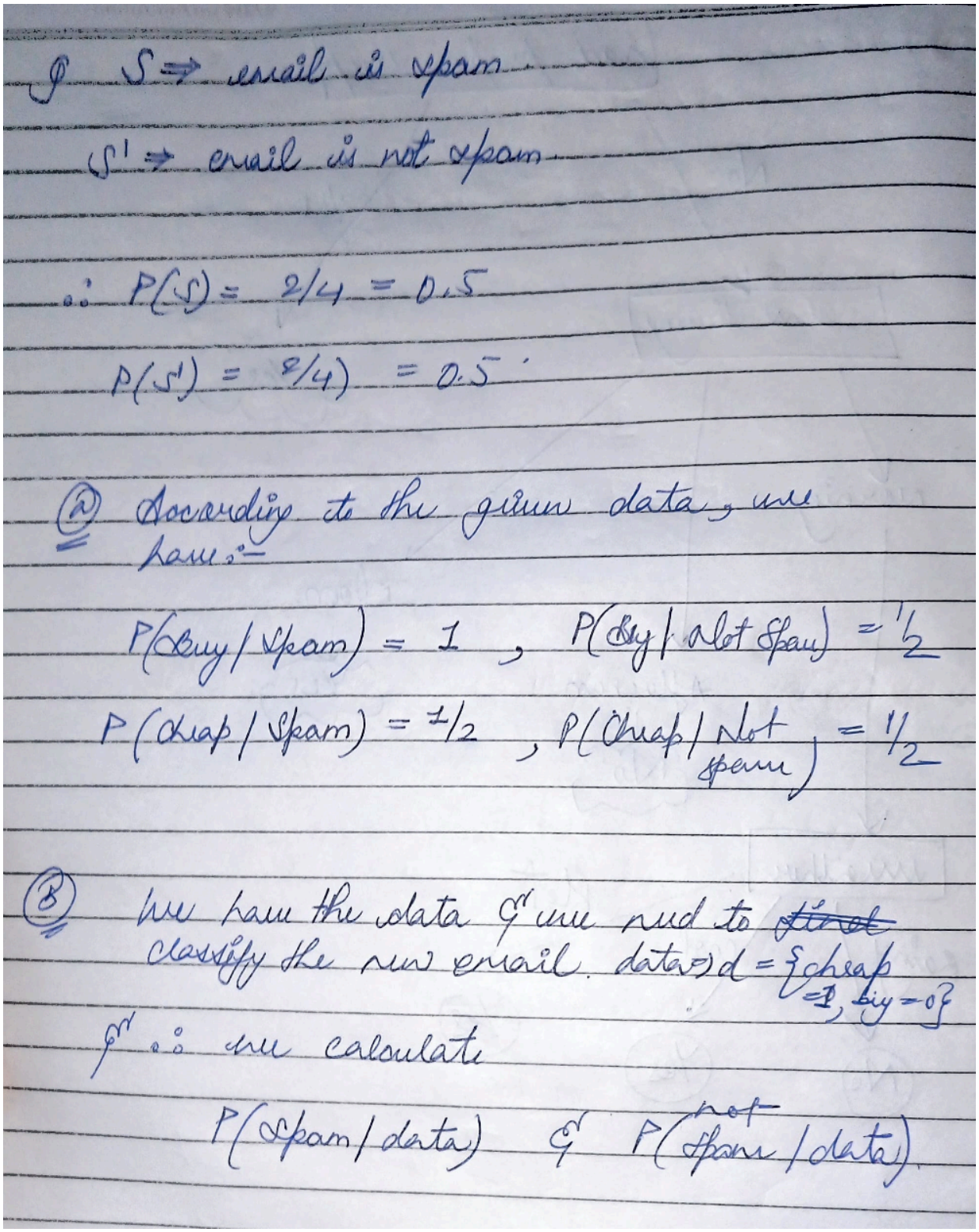
References for part c:

[1] K. Weinberger, "Lecture 3: The Perceptron." Accessed: Oct. 01, 2024. [Online]. Available: https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote03.html

[2] Kilian Weinberger, Lecture 5 "Perceptron" -Cornell CS4780 SP17, (Jul. 10, 2018). Accessed: Oct. 01, 2024. [Online Video]. Available: https://www.youtube.com/watch?v=wl7gVvI-HuY

[3] A. Blum, "15-859(B) Machine Learning Theory," CMU School of Computer Science.

part d.
\


S ⇒ email is spam
S' ⇒ email is not spam
∴ P(S) = 2/4 = 0.5
P(S') = 2/4) = 0.5
(a) According to the given data, we have :-
P(Buy / Spam) = 1 , P(Buy / Not Spam) = 1/2
P(Cheap / Spam) = 1/2 , P(Cheap / Not Spam) = 1/2
(B) We have the data & we need to classify the new email data) d = {cheap =1, buy = 0}
for ∴ we calculate
P(Spam / data) & P(Not Spam / data)

$$p(s/d) = \frac{p(d/s) \cdot p(s)}{p(d)}$$

$$= \frac{p(\text{cheap}=1, \text{buy}=0 / \text{spam}) \cdot p(s)}{p(\text{cheap}=1, \text{buy}=0)}$$

$$= p(b=0/s)\, p(c=1/s)\, p(s)$$

$$= 0 \quad -(1)$$

Since, $p(b=0/s) = 0$.

Similarly,

$$p(s'/d) = \frac{p(c=1/s')\, p(b=0/s')\, p(s')}{p(c=1, b=0)}$$

$$= 1/2 \times 1/2 \times 1/2 \Rightarrow \frac{1}{8} \quad -(2)$$

not calculating denominator.

$\therefore (2) > (1)$

$\therefore$ the new email with given data is most likely not spam.

(2) The problem with zero probability is in the fact that NB model cannot classify data that has not been seen before.

As the conditional probability for such a prob. will be zero.

In this case

$$p(\text{Sky}=0 \mid \text{Rain}) = 0$$

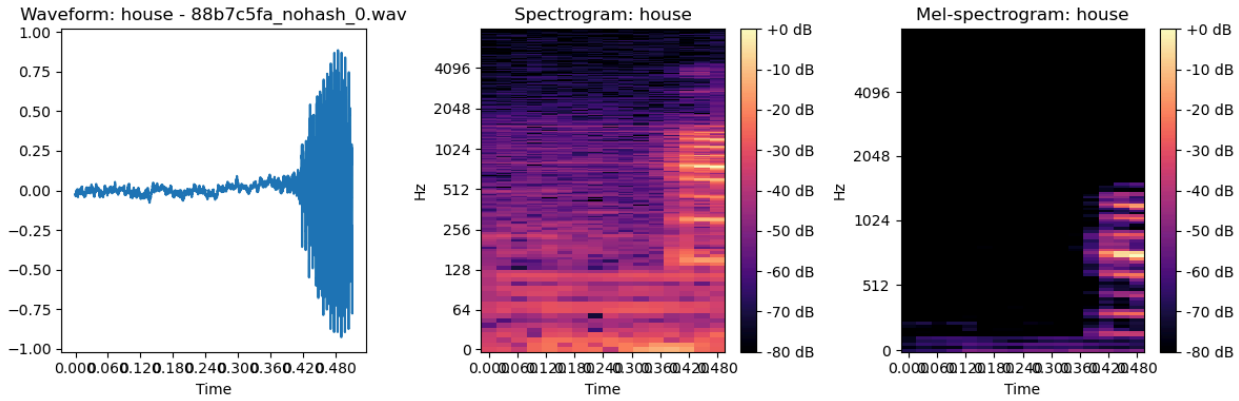
To address this we can use m-estimate or laplace techniques for smoothing

Section B

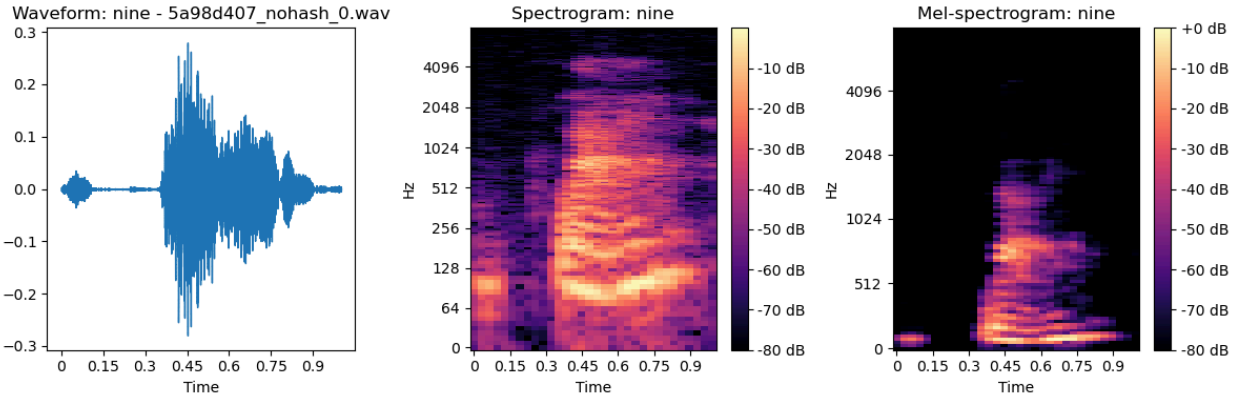
part a.

Observations:
- Most audio classes have relatively low amplitude means, suggesting that the recordings are generally quieter.
- The average duration across the classes shows slight variation, but for most, the mean duration hovers around 0.98 seconds.
- While there is some variability in the standard deviation of the duration, it remains fairly consistent across the different classes.
- The class "forward" stands out with the highest amplitude mean and standard deviation, whereas "nine" has the lowest amplitude mean.
- The class "six" has the longest average duration among all classes.

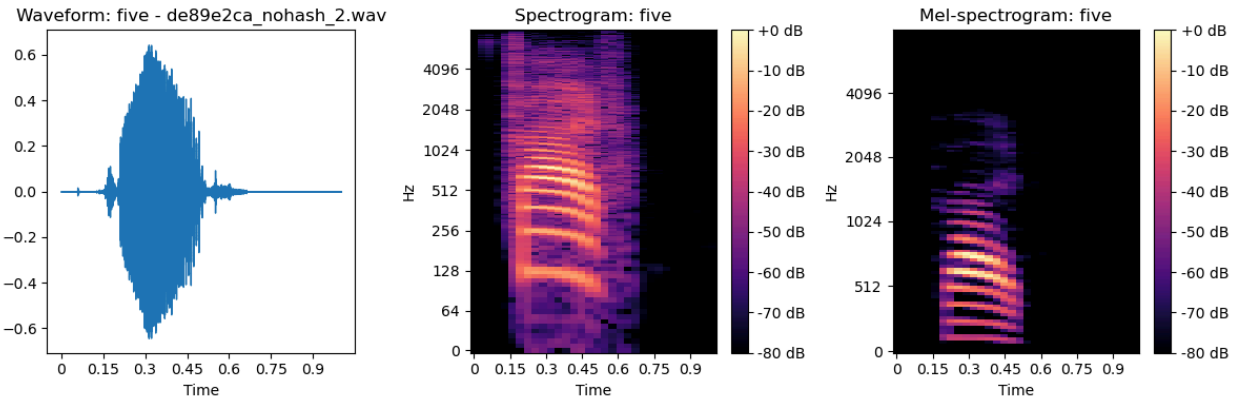
b)
Class : house



Class : nine



Class : five



1. “House” class:

Waveform:

- The waveform shows a prominent peak around 0.2 seconds, suggesting a short, sharp sound.

Spectrogram:

- A concentrated band of energy around 1000 Hz is observed. This is indicative of a dominant frequency component, likely associated with the vowel sound in "house."

Mel-spectrogram:

- Similar to the spectrogram, with a focus on the same frequency range. However, the mel-spectrogram might provide more detail in the lower frequency regions due to its emphasis on human hearing perception.

Interpretation:

- The "house" audio file likely represents a clear, isolated utterance of the word "house." The concentrated energy around 1000 Hz is consistent with the vowel sound in this word.

2. “Nine” class:

Waveform:

- A single, prominent peak suggests a short, sharp sound.

Spectrogram:

- A concentrated band of energy around 1000 Hz indicates a dominant frequency component.

Mel-spectrogram:

- Similar to the spectrogram, suggesting a focus on a specific frequency range.

Interpretation:

- The "nine" audio file likely represents a clear, isolated utterance of the number "nine." The concentrated energy around 1000 Hz is consistent with the pronunciation of this digit.

3. “Five” Class:

Waveform:

- Multiple peaks suggest a more complex or drawn-out sound.

Spectrogram:

- A wider range of frequencies suggests a richer sound quality.

Mel-spectrogram:

- Similar to the spectrogram, but potentially with more detail in the lower frequency regions due to the mel-scale's emphasis on human hearing perception.

Interpretation:

- The "five" audio file may represent a more varied or drawn-out utterance of the number "five." The wider frequency range could indicate additional phonetic elements or variations in pronunciation.

Based on the provided spectrograms and mel-spectrograms, here are some key observations:

Frequency Content:

- **"nine":** A concentrated band of energy around 1000 Hz suggests a dominant frequency associated with the vowel sound in "nine."
- **"five":** A wider range of frequencies indicates a more complex sound, possibly due to additional phonetic elements or variations in pronunciation.
- **"house":** Similar to "nine," a concentrated band of energy is observed around 1000 Hz, likely associated with the vowel sound in "house."

Duration:

- **"nine" and "house":** Both appear to be relatively short and focused sounds.
- **"five":** The waveform suggests a longer, more drawn-out sound.

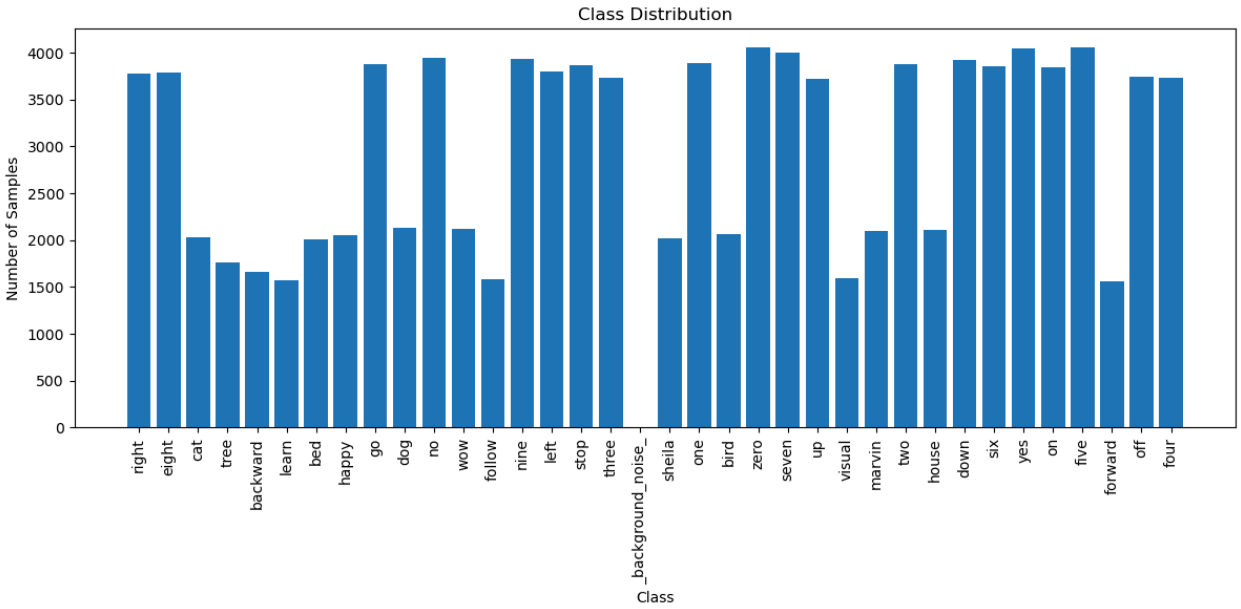
Mel-Spectrogram Analysis:

- **Overall:** The mel-spectrograms reinforce the findings from the spectrograms, providing additional detail in the lower frequency regions.
- **"five":** The mel-spectrogram might reveal more information about the lower frequency components of the sound.

Overall Observations:

- **"nine" and "house":** These words share a similar frequency characteristic, likely due to the presence of the vowel sound in both. They also appear to be shorter in duration.
- **"five":** This word exhibits a wider range of frequencies and a longer duration, suggesting a more complex or varied pronunciation.

part c.

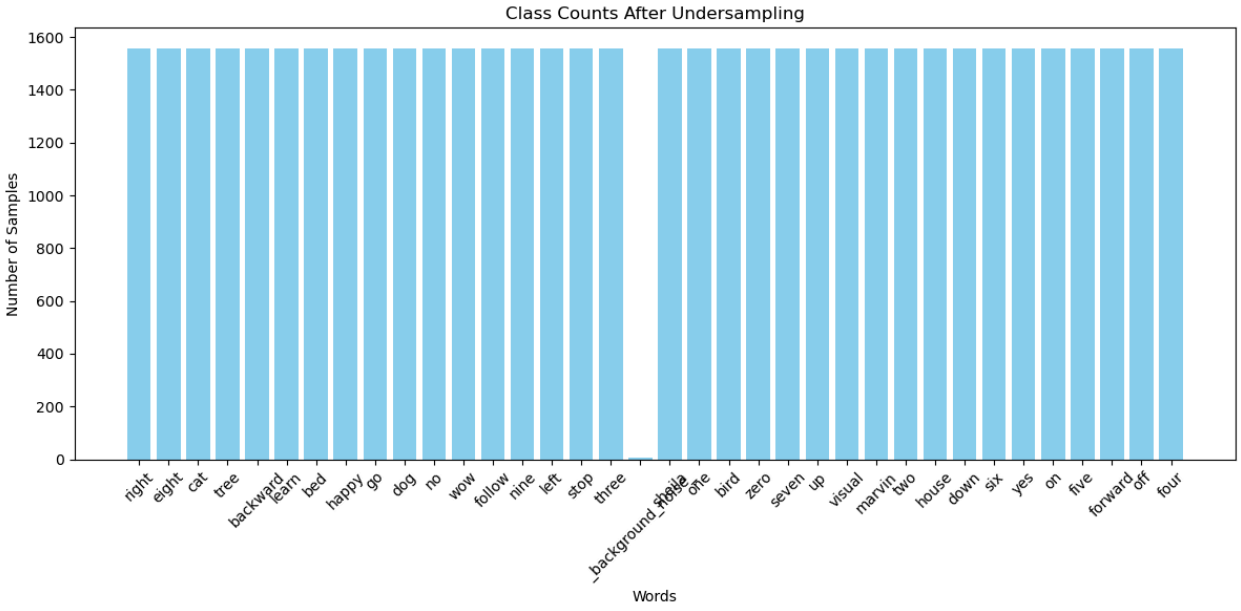


Overall Observations:

- **Class Imbalance:** The plot clearly shows that some classes have significantly more samples than others. This is a common issue in many machine learning datasets and can impact model performance.
- **Dominant Classes:** The classes "eight," "follow," "nine," "left," "stop," "three," and "house" appear to have the highest number of samples, potentially making them easier for a model to learn.
- **Underrepresented Classes:** Classes like "background_noise," "sheila," "bird," and "zero" have significantly fewer samples, which could make them more challenging for a model to learn accurately.

Specific Insights:

- **Words vs. Commands:** The plot shows that words like "eight," "nine," and "house" have a higher number of samples compared to commands like "up," "down," "left," and "right." This might suggest that the dataset was collected with a focus on recognizing words over commands.
- **Noise Class:** The "background_noise" class is present, indicating that the dataset includes samples with background noise. This could be beneficial for training a model to be more robust to noise.



Analyzing the Class Distribution Plot After Undersampling

Overall Observations:

- **Reduced Class Imbalance:** The plot shows a significant improvement in the class distribution compared to the original plot. The heights of the bars are now much more similar, indicating a more balanced dataset.
- **Consistent Sample Counts:** The majority of classes have approximately the same number of samples, which is a positive outcome of undersampling.
- **Potential Outlier:** The "background_noise" class still appears to have a slightly lower number of samples than the others. However, the difference is less pronounced compared to the original plot.

Specific Insights:

- **Undersampling Effectiveness:** The undersampling technique seems to have successfully reduced the number of samples in the overrepresented classes, leading to a more balanced distribution.
- **Class Representation:** All classes are now reasonably represented, which should improve the model's ability to learn and generalize to different classes.

Evaluating the selected models

I used two models, Naive Bayes and Perceptron, and implemented them from scratch in this section.

The Naive Bayes model achieved a test accuracy of 19.33%, indicating it could classify some audio samples, but still struggled due to issues like feature representation and possible class imbalance. In contrast, the Perceptron model had an accuracy of 10%. This poor performance is likely because the Perceptron isn't well-suited for multiclass classification without modifications, as it's primarily designed for binary classification tasks. This highlights the need for better feature engineering and potentially using more suitable models for handling multiple classes in future attempts.

Section C

part 1.

a. The dataset consists of 12,600 images, classified into 15 different human activity classes. Each class contains an equal number of 840 images, ensuring a balanced distribution across the dataset. The activity labels include actions such as **sitting**, **using a laptop**, **hugging**, **sleeping**, **dancing**, and more.

Image Size and Resolution

The image dimensions vary, with the width ranging from **84 pixels to 478 pixels** and the height ranging from **84 pixels to 318 pixels**. Most images (50th percentile) have dimensions close to 275 pixels in width and 183 pixels in height.

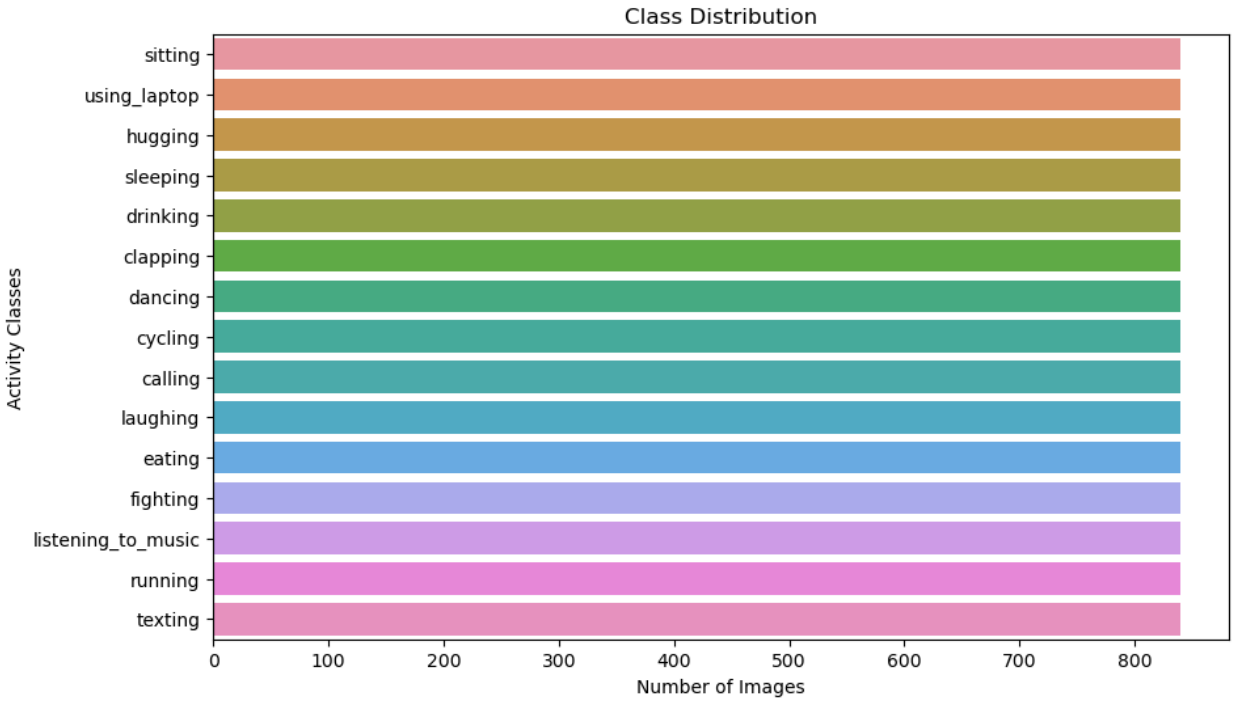
Aspect Ratio

The aspect ratio of the images, defined as the width divided by the height, has an **average value of 1.39**, indicating that most images are slightly wider than they are tall. The aspect ratio ranges from 0.45 (taller images) to 4.55 (wider images), with a majority of the images having an aspect ratio close to 1.5.

File Size Distribution

The file sizes of the images range from **3.6 KB to 44.1 KB**, with an average size of approximately 17.8 KB. The majority of the images fall between 13.6 KB and 21.3 KB, reflecting a relatively consistent image quality and compression level across the dataset.

b.



Balanced Distribution: Based on the bar plot obtained, it seemed the dataset is well-balanced, with each of the 15 activity classes containing exactly 840 images. This uniform distribution across classes ensures that the dataset is suitable for training and evaluating models without bias toward any particular class.

There are no significant discrepancies in the number of images per class. This balance is crucial for model training as it helps prevent overfitting to any one class and ensures that the model generalizes well across all activities.

c. Upon investigating the class distribution in the dataset, it was observed that all 15 activity classes are represented equally, each containing 840 images. This balanced distribution indicates that there are no significant class imbalances. However, should future analyses reveal any disparities or if model performance issues arise, data augmentation and resampling techniques can be employed. Data augmentation, such as rotation and flipping, can introduce variability and improve model robustness, especially if image similarity is high.

part 2.

The feature extraction process resulted in a combined feature vector of 306 dimensions per image, following are the observation regarding features:

- **HOG Features (144 dimensions)**: Histogram of Oriented Gradients (HOG) captures edge-based information by encoding gradients and orientation, effectively representing the shape and structural elements of objects within the images.
- **RGB Histogram (48 dimensions)**: The RGB histogram provides a distribution of colour intensities across the three primary colour channels, capturing essential colour features.
- **HSV Histogram (48 dimensions)**: By converting the images to the HSV (Hue, Saturation, Value) colour space, the HSV histogram adds robustness to colour variations, enhancing the colour representation.
- **Lab Histogram (48 dimensions)**: The Lab colour space histogram contributes an additional layer of colour description, which is particularly useful for representing perceptual differences between colours.
- **Haralick Features (13 dimensions)**: Haralick features are derived from the gray-level co-occurrence matrix (GLCM) and capture texture information from images. These features quantify the texture by examining the spatial relationship of pixels in grayscale images, providing insights into the structure and patterns within the textures present.
- **GLCM Features (4 dimensions)**: GLCM (Gray-Level Co-occurrence Matrix) features include contrast, correlation, energy, and homogeneity. These metrics quantify the texture of the image based on the distribution of pixel pairs with specific values and spatial relationships, allowing for a detailed texture analysis that aids in differentiating between various surfaces and patterns.
- **Edge Density (1 dimension)**: Edge density measures the proportion of edges detected in the image. By applying the Canny edge detection algorithm, this feature quantifies the presence of edges, which can be indicative of object boundaries and structural details, enhancing the representation of the image's geometrical characteristics.

HOG features have a higher dimension count because they extensively capture edge orientations and gradients, providing a rich representation of shapes and structural details. In comparison, RGB, HSV, and Lab histograms focus on color distributions and thus yield fewer dimensions. Haralick and GLCM features quantify texture but are limited to specific metrics, making HOG more effective for tasks requiring detailed shape analysis.

part 3.

Model Evaluation Results

1. **Naive Bayes**
    - **Training Accuracy:** 0.2484
    - **Testing Accuracy:** 0.2306
    - **Observation:** The model shows low accuracy on both training and testing datasets, indicating that it may not be capturing the underlying patterns in the data effectively.
2. **Decision Tree**
    - **Training Accuracy:** 0.6463
    - **Testing Accuracy:** 0.1718
    - **Observation:** While the training accuracy is relatively high, the significantly low testing accuracy suggests overfitting. The model performs well on training data but fails to generalize to unseen data.
3. **Perceptron**
    - **Training Accuracy:** 0.2671
    - **Testing Accuracy:** 0.2246
    - **Observation:** Similar to Naive Bayes, the Perceptron shows low accuracy across both datasets, indicating limited effectiveness in learning from the provided features.
4. **Random Forest (RF)**
    - **Training Accuracy:** 1.0000
    - **Testing Accuracy:** 0.3702
    - **Observation:** The Random Forest model achieves perfect accuracy on the training dataset but struggles with testing accuracy, indicating potential overfitting. While it can fit the training data very well and in fact is clearly overfitting, it does not generalize effectively to new data.
5. **LightGBM (LGB)** (Ensemble of Decision Trees)
    - **Training Accuracy:** 0.8850
    - **Testing Accuracy:** 0.4171
    - **Observation:** The LGB model, which is an ensemble of decision trees, demonstrates better performance compared to other models like

Naive Bayes and Perceptron, achieving a testing accuracy of 41.71%. This indicates a potential improvement in capturing patterns within the data. Largely, attributed to the boosting algorithm employed in its training process.

Based on the evaluation of various models, the **LightGBM (LGB)** (Ensemble of Decision Trees) was ultimately chosen for its relatively higher testing accuracy of **0.4171** compared to the other models. Despite achieving perfect training accuracy, which indicates a tendency for overfitting, the LGB model still outperformed Naive Bayes, Decision Tree, Random Forest, and Perceptron regarding its ability to generalize to unseen data. Thus, it is considered the most promising model for this task, warranting further optimization and validation to enhance its performance.