Assignment 2 - Naive Bayes, Decision Trees, Random Forests

by Kunal Sharma

Machine Learning (CSE343/ECE343)

Prof. Jainendra Shukla

Indraprastha Institute of Information Technology

Delhi

Oct 02, 2024

Section A (Theoretical)

A)

Combining the normal distribution assumptions given in the problem with Bayes' Theorem, we can determine the probability that a corporation that reported a 4% profit rise the previous year will pay out a dividend this year.

The formula for Bayes' Theorem is:

$$P(D|4\%) \ = \ \frac{P(4\%|D).P(D)}{P(4\%)}$$

Where,

- P(D|4%) represents the probability that the company will issue a dividend this year, given that it reported a 4% profit increase last year.
- P(4%|D) is the likelihood of observing a 4% profit increase assuming the company issues dividends.
- P(D) is the prior probability that a company issues dividends, which is provided as 0.8.
- P(4%) is the total probability of observing a 4% profit increase across all companies, regardless of whether they issue dividends.

To find P(4%), we apply the law of total probability:

$$P(4\%) \ = \ P(4\%|D).P(D) \ + \ P(4\%|ND).P(ND)$$

Where, P(ND) is the probability that a company does not issue dividends, given as 0.2.

Next, we assume that the profit increases follow a normal distribution. Since the variance is 36%, the standard deviation will be $\sqrt{36} \ = \ 6\%$.

For companies that issue dividends, we assume the profit increase follows a normal distribution with a mean of 10% and a standard deviation of 6%. For companies that do not issue dividends, the mean profit increase is 0%, with the same standard deviation:

$$Profit_D \sim N(10\%, 6^2)$$

$$Profit_{ND} \sim N(0\%, 6^2)$$

Now using the normal distribution formula to calculate the probabilities:

$$P(4\%|D) = \frac{e^{-\frac{(4-10)^2}{2.6^2}}}{6.\sqrt{2\pi}} = 0.0403(approx)$$

$$P(4\%|ND) = \frac{e^{-\frac{(4-0)^2}{2.6^2}}}{6.\sqrt{2\pi}} = 0.0532(approx)$$

Using above values, now calculating the total probability of a 4% profit increase:

$$P(4\%) = (0.0403 \cdot 0.80) + (0.0532 \cdot 0.20) = 0.04288$$

Now at last applying Bayes' Theorem to estimate the probability that a company with a 4% profit increase will issue a dividend:

$$P(D|4\%) = \frac{0.0403 \cdot 0.80}{0.04288} = 0.752\ (Approx)$$

Thus, the likelihood that a company with a 4% profit increase last year will issue a dividend this year is approximately 75.2%. This means there is a strong chance that such a company will choose to distribute dividends to its shareholders.

B)

The Target Variable is Attended ML Class (YES/NO)

Step 1: Calculating Entropy for the Entire Dataset

For the dataset S with 7 positive instances and 5 negative instances, we can calculate the entropy Using below formula,

$$Entropy(S) = -p_{yes}\,log_2(p_{yes}) - p_{no}\,log_2(p_{no})$$

Where, $p_{yes} = \frac{7}{12}$ & $p_{no} = \frac{5}{12}$

Putting the above values in the formula we get,

$$Entropy(S) = -\left(\frac{7}{12}\right) log_2\left(\frac{7}{12}\right) - \left(\frac{5}{12}\right) log_2\left(\frac{5}{12}\right) \approx 0.9799$$

Step 2: Entropy for Each Attribute

Class Time:

1. **Morning:** {+3,-1}

$$Entropy(Morning) = -\left(\frac{3}{4}\right) log_2\left(\frac{3}{4}\right) - \left(\frac{1}{4}\right) log_2\left(\frac{1}{4}\right) \approx 0.8113$$

2. **Noon:** {+2, -2}

$$Entropy(Noon) = -\left(\frac{2}{4}\right) log_2\left(\frac{2}{4}\right) - \left(\frac{2}{4}\right) log_2\left(\frac{2}{4}\right) \approx 1$$

3. **Afternoon:** {+2,-2}

$$Entropy(Afternoon) = -\left(\frac{2}{4}\right) log_2\left(\frac{2}{4}\right) - \left(\frac{2}{4}\right) log_2\left(\frac{2}{4}\right) \approx 1$$

Information Gain for Class Time:

$$IG(S, Class\,Time) = Entropy(S) - \left(\frac{4}{12}\right)Entropy(Morning) - \left(\frac{4}{12}\right)Entropy(Noon) - \left(\frac{4}{12}\right)Entropy(Afternoon)$$

$$IG(S, Class\,Time) \approx 0.9799 - \frac{1}{3}(0.8113 + 1 + 1) = 0.0428$$

Had Proper Sleep:

1. **Yes:** {+6, -0} (No uncertainty)

$$Entropy(Yes) = 0$$

2. **No:**

$$Entropy(No) = -\left(\frac{1}{6}\right) log_2\left(\frac{1}{6}\right) - \left(\frac{5}{6}\right) log_2\left(\frac{5}{6}\right) \approx 0.65$$

Information Gain for Had Proper Sleep:

$$IG(S, Had\ Proper\ Sleep) = Entropy(S) - \left(\frac{6}{12}\right)Entropy(Yes) - \left(\frac{6}{12}\right)Entropy(No)$$

$$IG(S, Had\ Proper\ Sleep) \approx 0.9799 - \frac{0.65}{2} = 0.6549$$

Weather:

1. **Cool:** {+4, -1}

$$Entropy(Cool) = -\left(\frac{4}{5}\right) log_2\left(\frac{4}{5}\right) - \left(\frac{1}{5}\right) log_2\left(\frac{1}{5}\right) \approx 0.7219$$

2. **Rainy:** {+0, -2} (No uncertainty)

$$Entropy(Rainy) = 0$$

3. **Hot:** {+3, -2}

$$Entropy(Hot) = -\left(\frac{3}{5}\right) log_2\left(\frac{3}{5}\right) - \left(\frac{2}{5}\right) log_2\left(\frac{2}{5}\right) \approx 0.9710$$

Information Gain for Had Proper Sleep:

$$IG(S, Weather) = Entropy(S) - \left(\frac{5}{12}\right)Entropy(Cool) - \left(\frac{2}{12}\right)Entropy(Rainy) - \left(\frac{5}{12}\right)Entropy(Hot)$$

$$IG(S, Weather) \approx 0.9799 - \left(\frac{5}{12}\right)(0.7219 + 0.9710) = 0.2745$$

Comparing of Information Gain:

- $G(S, Class\ Time) = 0.0428$
- $IG(S, Had\ Proper\ Sleep) = 0.6549$
- $IG(S, Weather) = 0.2745$

Since Had Proper Sleep has the highest information gain, it will be selected as the root node.

Observations:

When **Had Proper Sleep = Yes**, the target variable **Attended ML Class** is always "YES", resulting in a pure subset with entropy 0. No further splitting is required for this branch.

For **Had Proper Sleep = No**, we need to calculate the next node by repeating the steps for **Class Time** and **Weather.**

Next Node Calculating for "Had Proper Sleep = No":

- **Entropy of Had Proper Sleep = No:** {+1, -5}

$$Entropy(No) = -\left(\frac{1}{6}\right) log_2\left(\frac{1}{6}\right) - \left(\frac{5}{6}\right) log_2\left(\frac{5}{6}\right) \approx 0.65$$

Class Time:

1. **Morning:** {+1, -1}

$$Entropy(Morning) = -\left(\frac{1}{2}\right) log_2\left(\frac{1}{2}\right) - \left(\frac{1}{2}\right) log_2\left(\frac{1}{2}\right) \approx 1$$

2. **Noon:** {+0, -2} (No uncertainty)

$$Entropy(Noon) = 0$$

3. **Afternoon:** {+0, -2} (No uncertainty)

$$Entropy(Afternoon) = 0$$

Information Gain for Class Time:

$$IG(HadProperSleep = No, Class\ Time) = Entropy(No) - \left(\frac{2}{6}\right)Entropy(Morning)$$

$$- \left(\frac{2}{6}\right)Entropy(Noon) - \left(\frac{2}{6}\right)Entropy(Afternoon)$$

$$IG \approx 0.65 - \left(\frac{1}{3}\right)(1) = 0.3167$$

Weather:

1. **Cool:** {+1, -1}

$$Entropy(Cool) = -\left(\frac{1}{2}\right) log_2\left(\frac{1}{2}\right) - \left(\frac{1}{2}\right) log_2\left(\frac{1}{2}\right) \approx 1$$

2. **Rainy:** {+0, -2} (No uncertainty)

$$Entropy(Rainy) = 0$$

3. **Hot:** {+0, -2} (No uncertainty)

$$Entropy(Hot) = 0$$

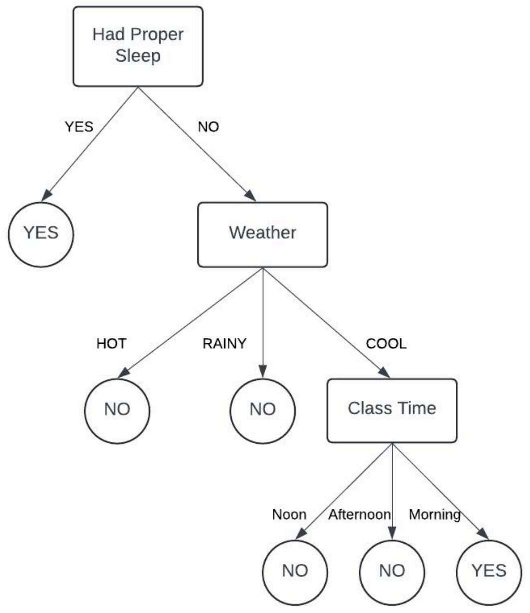
Information Gain for Weather:

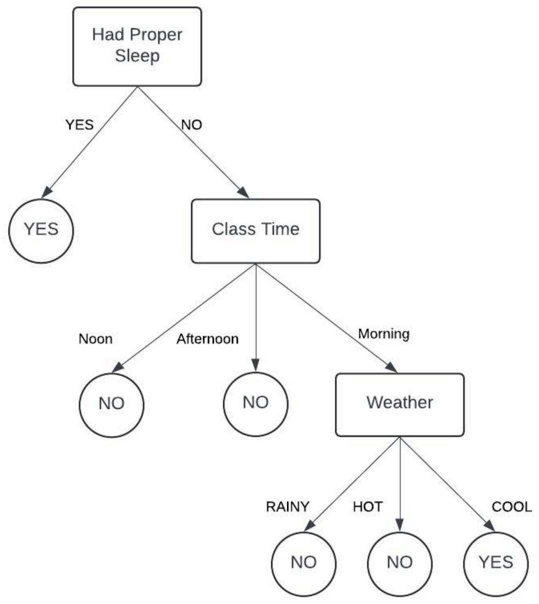
$$IG(HadProperSleep = No, Weather) = Entropy(No) - \left(\frac{2}{6}\right)Entropy(Cool) - \left(\frac{2}{6}\right)Entropy(Rainy) - \left(\frac{2}{6}\right)Entropy(Hot)$$

$$IG \approx 0.65 - \left(\frac{1}{3}\right)(1) = 0.3167$$

Final Decision Tree Construction:

Since both **Class Time** and **Weather** have the same information gain, either can be chosen as the next node in the decision tree. Below are the both possible Decision trees.





C)

To establish the mistake bound for this specific type of Perceptron algorithm, we follow an approach similar to the original Perceptron proof but adapt it to account for margin mistakes.

We start with a sequence of labeled examples S, which is consistent with a linear threshold function. The true weight vector w* has unit length, and γ represents the minimum margin over all examples in S Specifically:

$$\gamma = min\left(\frac{|w^{*} . x|}{||x||}\right)$$

where each x ∈ S is normalized to have Euclidean length 1.

n this modified algorithm, the weight vector w_t is updated when there is either:

- A classification mistake, or
- A margin mistake, where the margin is less than $\gamma/2$

The objective is to show that the total number of updates (or mistakes) is at most $\frac{8}{\gamma^2}$

Step 1: Progress per Update

As in the original Perceptron algorithm, each update increases, $w_t . w^{*}$ by at least γ However, the complexity here lies in bounding the growth of $||w_t||$ because updates occur even when the angle between w_t and x is more than 90 degrees (margin mistakes).

In the original Perceptron algorithm, we had:

$$||w_{t+1}||^2 <= ||w_t||^2 + 1$$

This leads to:

$$||w_{t+1}|| <= ||w_t|| + 1$$

However, in this new algorithm, the update rule is modified due to margin mistakes, and we obtain the following bound:

$$||w_{t+1}|| \;<=\; ||w_t|| \;+\; \frac{1}{2}||w_t|| \;+\; \frac{\gamma}{2}$$

This is derived by decomposing x into its orthogonal and parallel parts. The parallel part adds at most $\gamma/2$ to the length of w_t.

Step 2: Simplifying the Bound

To simplify, we observe that if $||w_t||$ >= 2/γ, then:

$$||w_{t+1}|| \;<=\; ||w_t|| \;+\; \frac{3\gamma}{4}$$

After M updates, the inequality becomes:

$$||w_{M+1}|| \;<=\; \frac{2}{\gamma} \;+\; \frac{3M\gamma}{4}$$

Step 3: Solving for the Number of Mistakes

Solving the inequality:

$$M\gamma \;-\; \frac{3M\gamma}{4} \;<=\; \frac{2}{\gamma}$$

Simplifying this:

$$\frac{M\gamma^2}{4} \;<=\; 2$$

$$M \;<=\; \frac{8}{\gamma^2}$$

We can generalize for $(1 \ - \ \epsilon)\gamma$

The analysis proceeds similarly, with the difference that updates now depend on the margin $(1 \ - \ \epsilon)\gamma$

Therefore,

$$M \ <= \ \frac{8}{(1-\epsilon)^2\gamma^2}$$

D)

a) Assuming Independence of Features: "buy" and "cheap"

We need to calculate the probabilities for both "spam" and "non-spam" classes, using both regular probability estimates and m-estimate smoothing (with m = 2)

We have formula for m-estimate:

$$P(A_i \mid C) \ = \ \frac{N_{ic} + mp}{N_c + m}$$

1. **Probability of "buy=1" given "spam":**

- **Without Smoothing:**

$$P(buy \ = \ 1|spam) \ = \ \frac{2}{2} \ = \ 1$$

- **With m-estimate smoothing:**

$$P(buy \ = \ 1|spam) \ = \ \frac{2 + 0.5 \times 2}{2 + 2} \ = \ \frac{3}{4} \ = 0.75$$

2. **Probability of "cheap=1" given "spam":**

- **Without Smoothing:**

$$P(cheap = 1|spam) = \frac{1}{2} = 0.5$$

- **With m-estimate smoothing:**

$$P(cheap = 1|spam) = \frac{1 + 0.5 \times 2}{2 + 2} = 0.5$$

The probability remains the same with and without smoothing, because the dataset contains all possible records for this feature.

3. **Probability of "buy=1" given "non-spam":**

- **Without Smoothing:**

$$P(buy = 1|non - spam) = \frac{1}{2} = 0.5$$

- **With m-estimate smoothing:**

$$P(buy = 1|non - spam) = \frac{1 + 0.5 \times 2}{2 + 2} = 0.5$$

Again, the probability remains unchanged due to the dataset having all possible records for this feature.

4. **Probability of "cheap=1" given "non-spam":**

- **Without Smoothing:**

$$P(cheap = 1|non - spam) = \frac{1}{2} = 0.5$$

- **With m-estimate smoothing:**

$$P(cheap = 1|non - spam) = \frac{1 + 0.5 \times 2}{2 + 2} = 0.5$$

5. **Probability of "buy=0" given "spam":**

- **Without Smoothing:**

$$P(buy\ =\ 0|spam)\ =\ \frac{0}{2}\ =\ 0$$

- **With m-estimate smoothing:**

$$P(buy\ =\ 0|spam)\ =\ \frac{0 + 0.5 \times 2}{2 + 2}\ =\ \frac{1}{4}\ =\ 0.25$$

6. **Probability of "cheap=0" given "spam":**

- **Without Smoothing:**

$$P(cheap\ =\ 0|spam)\ =\ \frac{1}{2}\ =\ 0.5$$

- **With m-estimate smoothing:**

$$P(cheap\ =\ 0|spam)\ =\ \frac{1 + 0.5 \times 2}{2 + 2}\ =\ 0.5$$

The probability is unchanged because the dataset contains all possible records.

7. **Probability of "buy=0" given "non-spam":**

- **Without Smoothing:**

$$P(buy\ =\ 0|non\ -\ spam)\ =\ \frac{1}{2}\ =\ 0.5$$

- **With m-estimate smoothing:**

$$P(buy\ =\ 0|non\ -\ spam)\ =\ \frac{1 + 0.5 \times 2}{2 + 2}\ =\ 0.5$$

8. **Probability of "cheap=0" given "non-spam":**

- **Without Smoothing:**

$$P(cheap\ =\ 0|non\ -\ spam)\ =\ \frac{1}{2}\ =\ 0.5$$

- **With m-estimate smoothing:**

$$P(cheap = 0|non - spam) = \frac{1 + 0.5 \times 2}{2 + 2} = 0.5$$

b) Assuming Independence of Features: "buy" and "cheap"

We need to calculate the probabilities for "spam" and "non-spam" emails, both with and without m-estimate smoothing (using m = 2).

1. Prior Probabilities:

- **Probability of Spam:**

$$P(Spam) = \frac{Number\ of\ spam\ emails}{Total\ emails} = \frac{2}{4} = 0.5$$

- **Probability of Non-Spam:**

$$P(Non - Spam) = \frac{Number\ of\ non-spam\ emails}{Total\ emails} = \frac{2}{4} = 0.5$$

2. Prior Probabilities:

- **Probability of "cheap":**

$$P("cheap") = \frac{2}{4} = 0.5$$

- **Probability of "not buy":**

$$P("not\ buy") = \frac{1}{4} = 0.25$$

3. Probability of "cheap" but "not buy" given Spam::

- **Without smoothing:**

$$P(\text{"}cheap\text{"}\ but\ \text{"}not\ buy\text{"} \mid Spam) = P(\text{"}cheap\text{"} \mid Spam) \times P(\text{"}not\ buy\text{"} \mid Spam)$$

$$= 0\times 0.5 = 0$$

- **With m-estimate smoothing:**

$$P(\text{"}cheap\text{"}\ but\ \text{"}not\ buy\text{"} \mid Spam) = 0.5 \times 0.25 = 0.125$$

4. Probability of "cheap" but "not buy" given Non-Spam:

- **Without and with smoothing (same for both):**

$$P(\text{"}cheap\text{"}\wedge\text{"}not\ buy\text{"}|Non - Spam) = P(\text{"}cheap\text{"} \mid Non - Spam)$$

$$\times P(\text{"}not\ buy\text{"} \mid Non - Spam) = 0.5 \times 0.5 = 0.25$$

5. Total Probability of "cheap" but "not buy":

- **Without smoothing:**

$$P(\text{"}cheap\text{"}\ but\ \text{"}not\ buy\text{"}) = P(\text{"}cheap\text{"}\ but\ \text{"}not\ buy\text{"} \mid Spam) \times P(Spam)$$

$$+ P(\text{"}cheap\text{"}\ but\ \text{"}not\ buy\text{"} \mid Non - Spam) \times P(Non - Spam)$$

$$P(\text{"}cheap\text{"}\ but\ \text{"}not\ buy\text{"}) = 0 \times 0.5 + 0.25 \times 0.5 = 0.125$$

- **With m-estimate smoothing:**

$$P(\text{"}cheap\text{"}\wedge\text{"}not\ buy\text{"}) = 0.125\times 0.5 + 0.25\times 0.5 = 0.0625 + 0.125$$

$$= 0.1875$$

6. Posterior Probability of Spam given "cheap" but "not buy":

- **Without smoothing:**

$$P(Spam \mid \text{"}cheap\text{"}\ but\ \text{"}not\ buy\text{"}) = \frac{P(\text{"}cheap\text{"}\ but\ \text{"}not\ buy\text{"} \mid Spam) \times P(Spam)}{P(\text{"}cheap\text{"}\ but\ \text{"}not\ buy\text{"})}$$

$$= \frac{0 \times 0.5}{0.125} = 0$$

- **With m-estimate smoothing:**

$$P(Spam \mid \text{"}cheap\text{"}\ but\ \text{"}not\ buy\text{"}) = \frac{0.125 \times 0.5}{0.1875} = \frac{0.0625}{0.1875} = 0.333$$

7. Posterior Probability of Non-Spam given "cheap" but "not buy":

- **Without smoothing:**

$$P(Non - Spam \mid \text{"}cheap\text{"}\ but\ \text{"}not\ buy\text{"}) = \frac{P(\text{"}cheap\text{"} \wedge \text{"}not\ buy\text{"} \mid Non{-}Spam) \times P(Non{-}Spam)}{P(\text{"}cheap\text{"}\ but\ \text{"}not\ buy\text{"})}$$

$$= \frac{0.25 \times 0.5}{0.125} = 1$$

- **With m-estimate smoothing:**

$$P(Non - Spam \mid \text{"}cheap\text{"}\ but\ \text{"}not\ buy\text{"}) = \frac{0.25 \times 0.5}{0.1875} = \frac{0.125}{0.1875} = 0.667$$

C) The issue with zero probabilities arises when a feature combination has never been observed in the training data for a particular class. In such cases, the likelihood of that class for that feature combination becomes zero, causing Bayes' Theorem to return a posterior probability of zero for that class, regardless of other supporting evidence. This is problematic, especially when the unseen combination may still provide valuable information.

For instance, if there are no examples of emails marked as spam with the combination of "cheap" but not "buy" in the training data (as seen in the case when we don't apply smoothing), the computed probability for P(“cheap” but not “buy” | Spam) becomes 0. Consequently, the posterior probability for Spam given this combination is also zero. This happened because every spam email in the dataset contained the word "buy," leading to a zero likelihood for the "cheap but not buy" combination under the "Spam" class.

To handle this, we can apply methods such as Laplace Smoothing or m-estimate smoothing.

1. Laplace Smoothing (Additive Smoothing):

Laplace Smoothing adds a small constant (typically 1) to each feature count in every class to avoid zero probabilities. This ensures that even features that haven’t appeared in the training data have non-zero probabilities. By doing this, no feature can have a probability of zero.

Formula:

$$P(A_i \mid C) = \frac{N_{ic} + 1}{N_c + k}$$

Where, N_{ic} is the count of feature A_i in class C, N_c is the total count of all instances in class C and k is the number of unique features.

This approach ensures every feature has a non-zero probability, even if it hasn't appeared in the training data for a specific class.

2. m-Estimate Smoothing:

The m-estimate smoothing is a more generalized form of Laplace Smoothing. Instead of adding a constant 1, it incorporates a parameter m, which controls how much influence the prior probability of the feature has on the final estimate.

Formula:

$$P(A_i \mid C) = \frac{N_{ic} + m \, . \, p}{N_c + m}$$

Where, N_{ic} is the count of feature A_i in class C, N_c is the total number of instances in class C, p is the prior probability of feature A_i and m is a parameter controlling the amount of smoothing.

In the above formula:

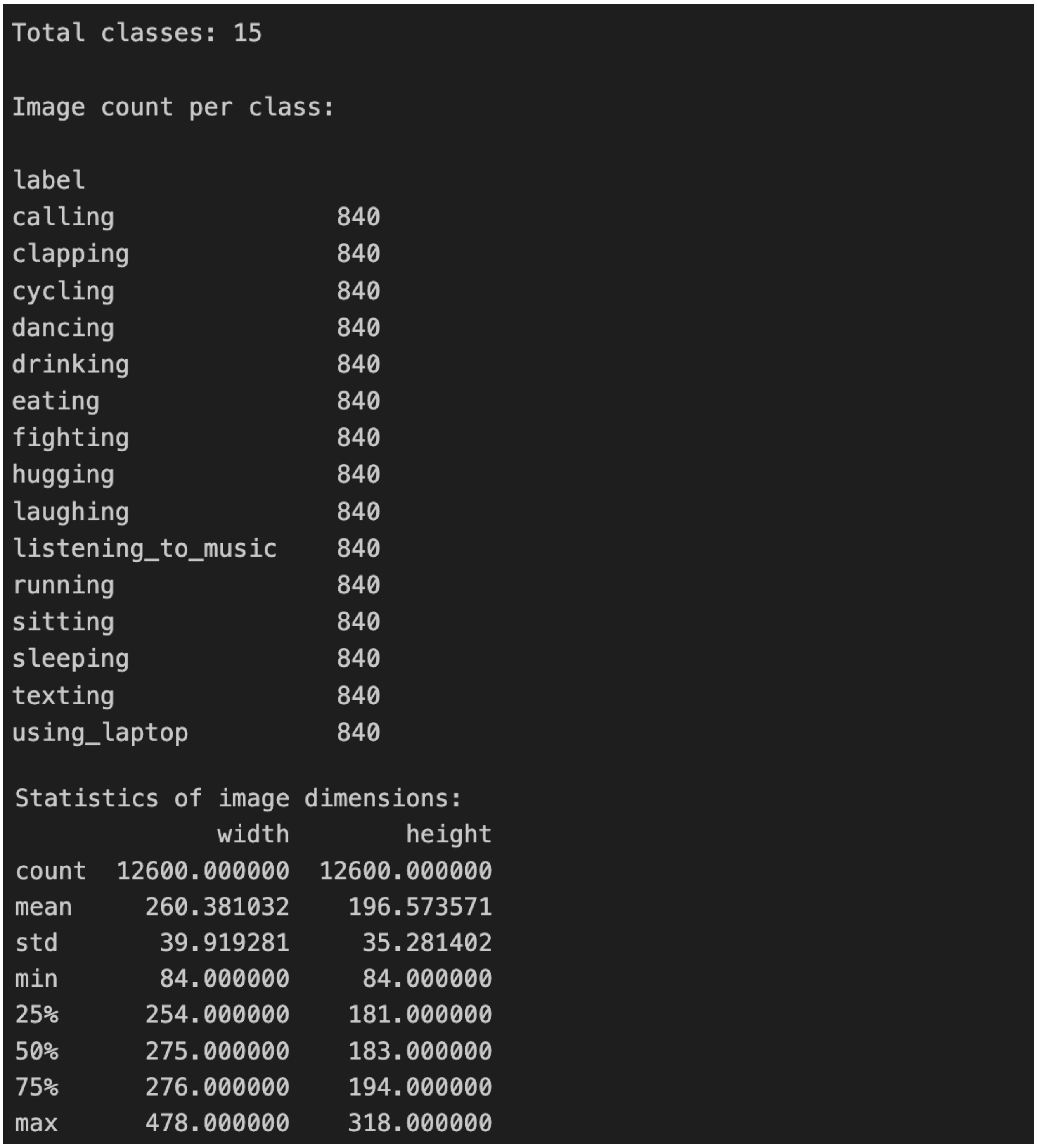
- If m = 0, no smoothing is applied.
- A higher mm gives more weight to the prior pp, thus influencing the final estimate based on prior knowledge of the feature's distribution.

By adjusting m, we can control how much smoothing to apply, which can be useful when we have limited or sparse training data.

Section C (Algorithm implementation using packages)

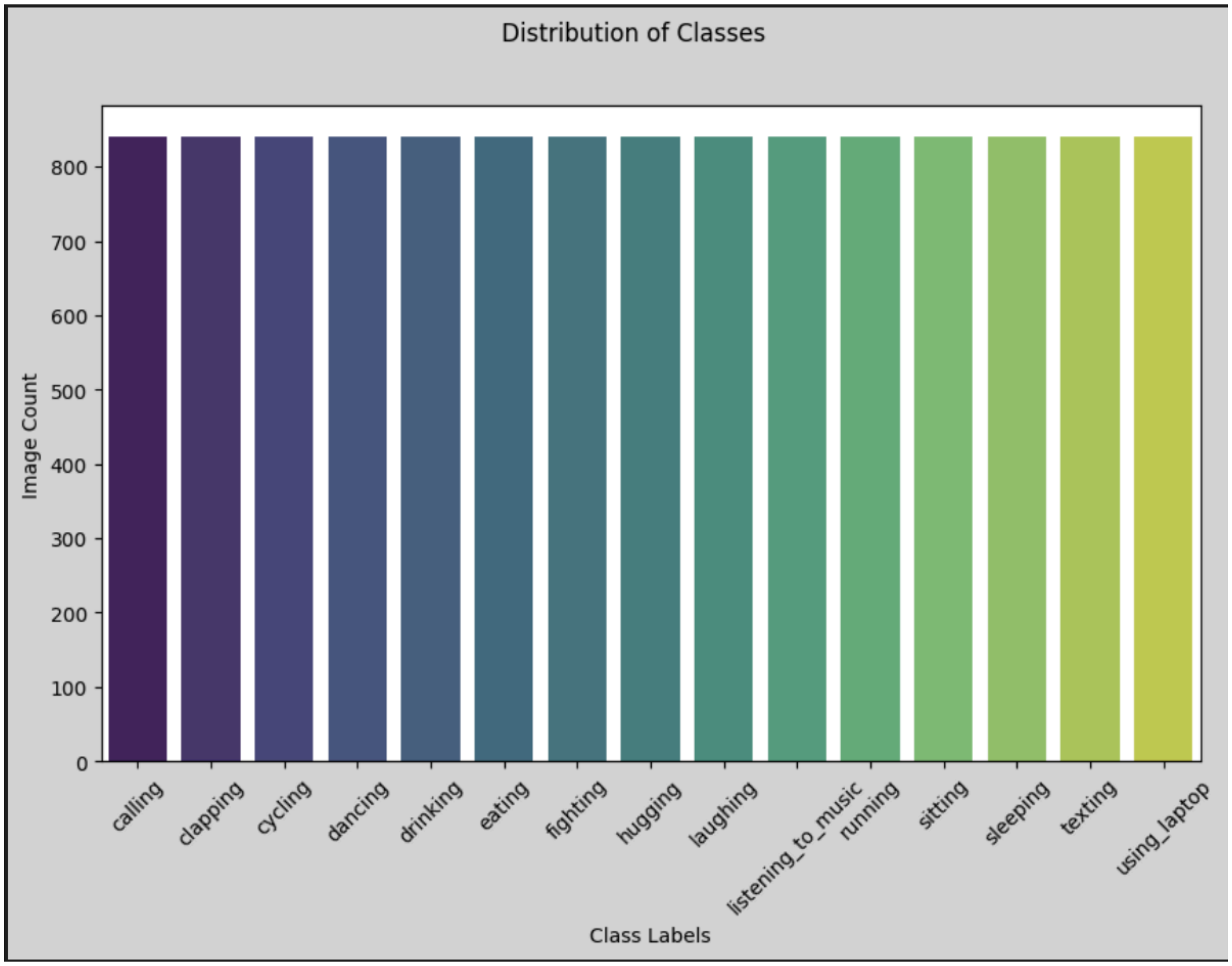
1)

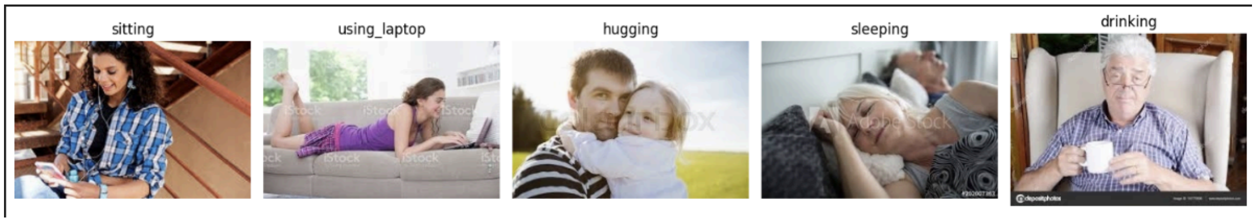
a) There are fifteen classes in all in our dataset. It is discovered that there are 840 photos in each class, and this quantity is the same for all classes. Most of the image sizes are the same. The photos' average width and height are 260 and 196 pixels, respectively. The outputs listed below show the precise statistics:

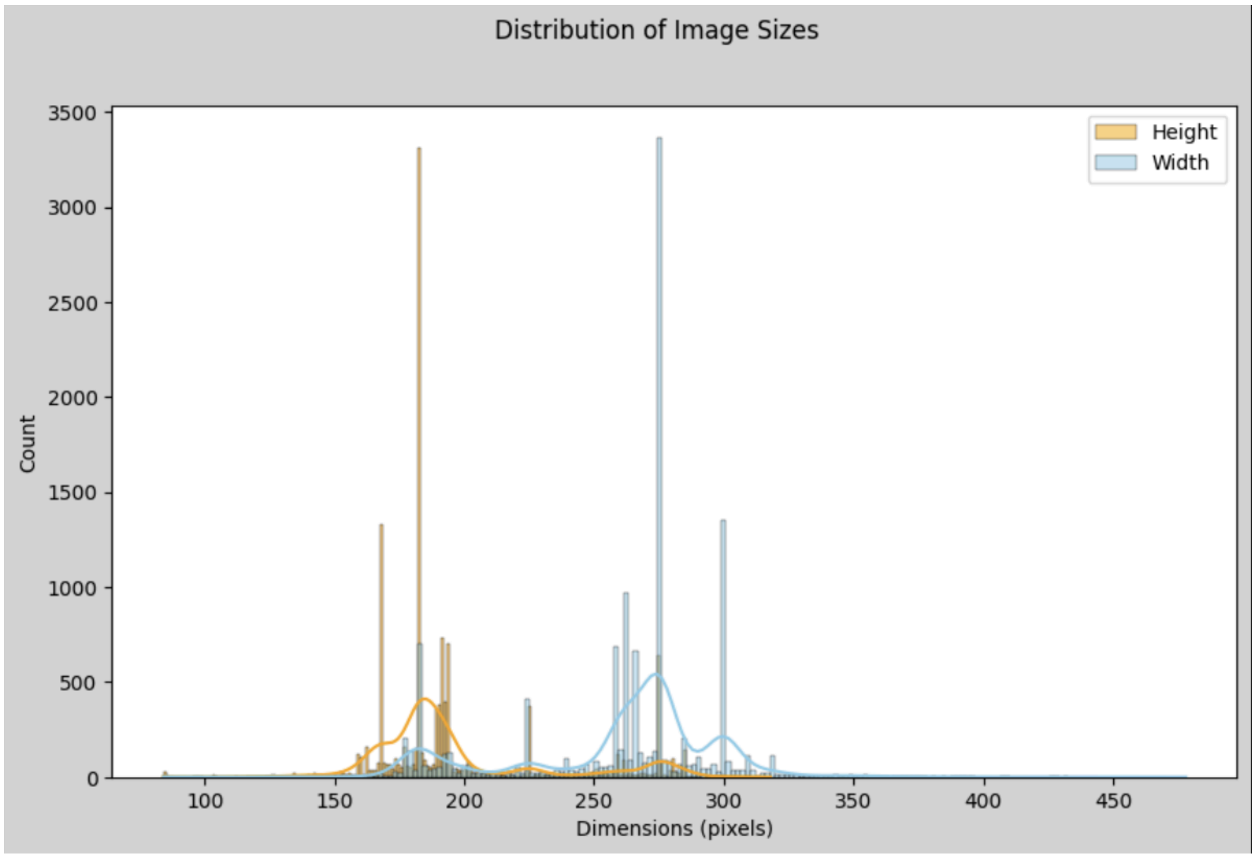


```
Total classes: 15

Image count per class:

label
calling               840
clapping              840
cycling               840
dancing               840
drinking              840
eating                840
fighting              840
hugging               840
laughing              840
listening_to_music    840
running               840
sitting               840
sleeping              840
texting               840
using_laptop          840

Statistics of image dimensions:
              width        height
count  12600.000000  12600.000000
mean     260.381032    196.573571
std       39.919281     35.281402
min       84.000000     84.000000
25%      254.000000    181.000000
50%      275.000000    183.000000
75%      276.000000    194.000000
max      478.000000    318.000000
```

b) It is determined that there are 840 photos in each class, and this amount is the same for all classes, as the distribution of classes makes clear. Most of the image sizes are the same. The maximum number of photos is 270 pixels wide and 180 pixels tall. Based on the sample photographs, it appears that the classification is rather accurate, which will be useful in developing a predictive model. Reliable data is essential to developing an accurate model.

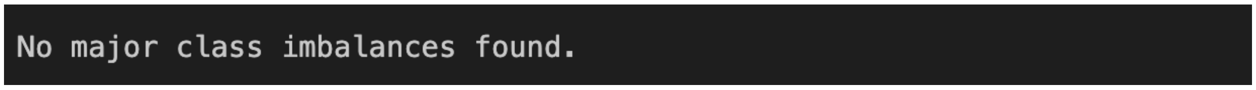






C) Since each class in the provided data has the same number of photos, there are no class imbalances when looking at the class distribution. Therefore, there is no need to add to the data. To determine the imbalance threshold, take 10% of the average number of photos in each class.

The maximum permissible imbalance is 10% of the average, which is calculated by dividing the total number of photographs by the number of classes. This yields the average number of images per class.

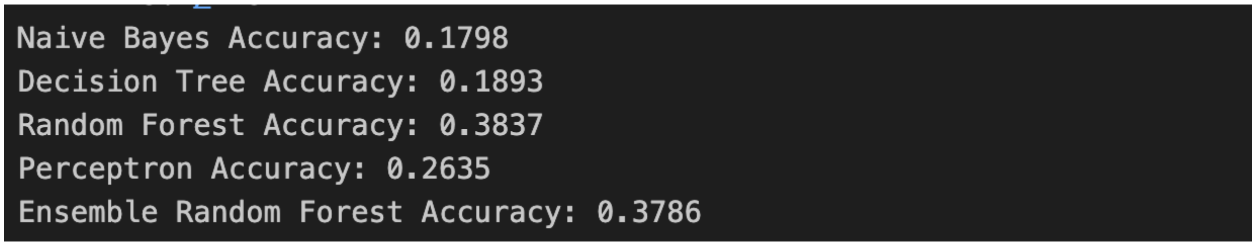


```
No major class imbalances found.
```

2) I took out five different kinds of features: edge density features, local binary patterns (LBP) features, histogram of oriented gradients (HOG) features, color histogram features, and gray-level co-occurrence matrix (GLCM) features. HOG captures the distribution of gradient orientations (edges) in an image.Regardless of the shape or orientation of the image, color histograms provide information about the color composition by capturing the distribution of colors in the image. Local Binary Pattern (LBP) is a texture descriptor that describes the local texture by capturing the relationship between a pixel and its neighbors.The percentage of edge pixels in an image is measured by edge density. By figuring out how frequently pairings of pixel values with particular intensities and spatial relationships occur in a picture, GLCM is able to extract texture information.

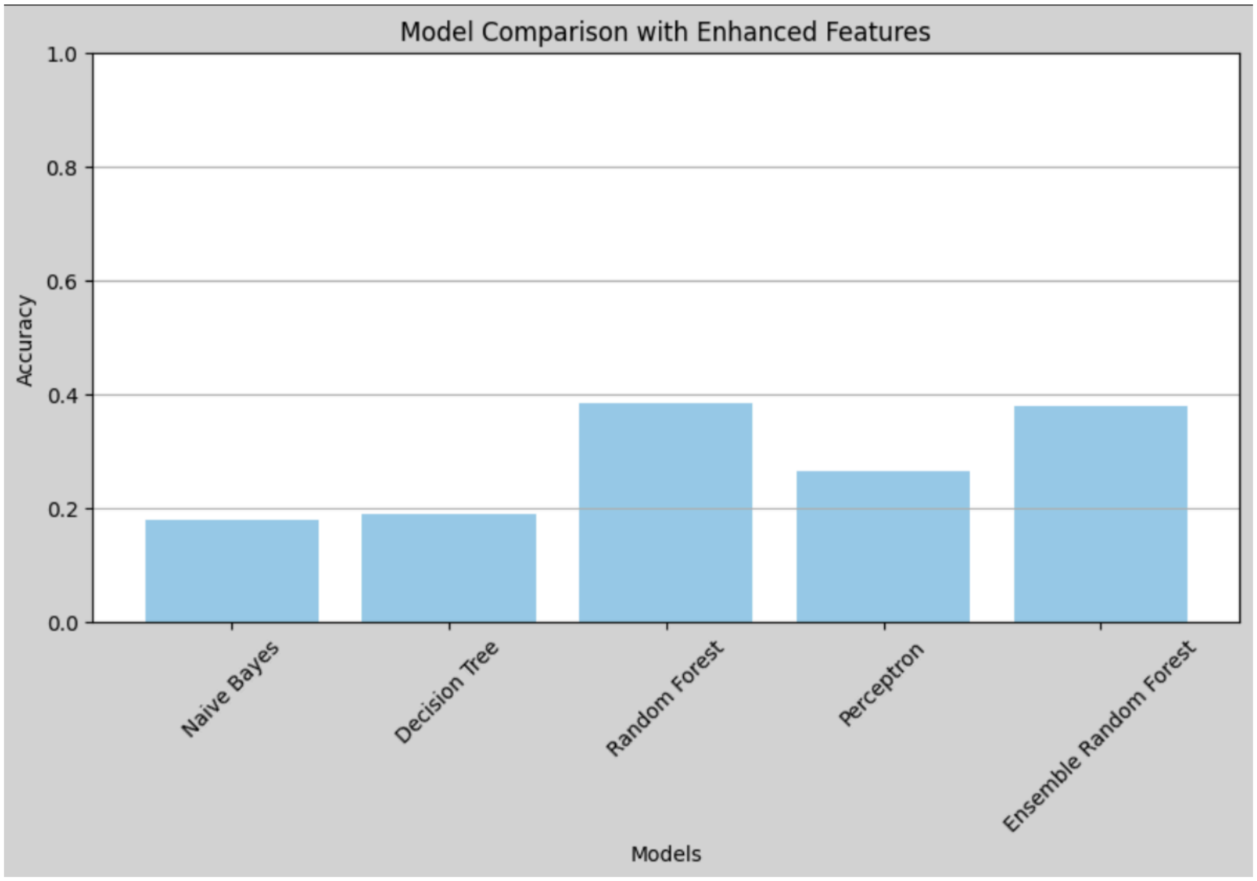
3)

a) I experimented with 5 distinct models and verified each one's accuracy. The output graphic below shows us that our random forest model produced the best accuracy which is 0.3837 or 38.37%.



```
Naive Bayes Accuracy: 0.1798
Decision Tree Accuracy: 0.1893
Random Forest Accuracy: 0.3837
Perceptron Accuracy: 0.2635
Ensemble Random Forest Accuracy: 0.3786
```

b) Given the class label, Naive Bayes assumes that all features are conditionally independent, which is rarely the case in picture data. When training on high-dimensional data, such as photographs, where each pixel or extracted feature may function as a separate attribute, decision trees have a tendency to overfit the data. Using an ensemble approach called Random Forest, overfitting is minimized and generalization is enhanced by constructing several Decision Trees and aggregating their predictions. A Perceptron is a simple linear classifier that operates by learning weights for features and making judgments based on a weighted sum. Although linear, it outperforms Naive Bayes and Decision Trees since it can still identify some relevant patterns from the retrieved visual characteristics. Now talking about the random forest ensemble because of its excessive complexity and tree count, the random forest ensemble overfits the training set, leading to poor generalization on the test set.


Model Comparison with Enhanced Features
Accuracy
Naive Bayes
Decision Tree
Random Forest
Perceptron
Ensemble Random Forest
Models