



University
of Glasgow

Computational Social Intelligence Coursework Report

Contents

Part 1: Laughter events	2
Introduction	2
Methodology	2
• Is the number of laughter events higher for women than for men?	3
• Is the number of laughter events higher for callers than for receivers?	4
• Are laughter events longer for women?	5
• Are laughter events longer for callers?	6
Conclusion	7
Part 2: Facial Expression Classification	8
Introduction	8
Methodology	8
• Parameters in Gaussian Discriminant Analysis	8
• Experimental setup and results	9
Conclusion	10
Appendix	10

Part 1: Laughter events

Introduction

Over the years, technological advancements, ranging from the first cellphone to the revolutionary internet, have significantly transformed our modes of communication. Online interactions have increasingly become the preferred alternative to traditional face-to-face meetings. It is reasonable to assert that future generations may come to perceive nonverbal, digital communication as the norm, a trend largely influenced by their contemporary environment. Nonverbal communication, categorized as a key aspect of human interaction, can be described as a range of signs that add social, psychological, and emotional aspects to spoken conversations. These cues are many and diverse in face-to-face interactions, and they include postures, gestures, and facial emotions. But when it comes to phone calls, the lack of visual clues places all the weight of nonverbal communication on the aural channel, that is, voice.

This study explores a specific aspect of nonverbal communication from a dataset with 60 telephonic interactions involving a total of 120 participants, 57 male and 63 female. The primary nonverbal cue examined in this dataset is the occurrence of laughter. Laughter is a powerful nonverbal communication tool that may express messages beyond spoken words. The objective of this study is to analyze the nuances of laughter within telephonic conversations. This involves a detailed examination of laughter's frequency and duration, and its correlation with variables such as the gender of the participants, their conversational roles (either as caller or recipient), and duration of the laughter events. While performing statistical tests, the aim is to identify patterns and correlations that may reject or accept the null hypothesis.

Methodology

The dataset 'laugheter-corpus.csv' was first subjected to a preliminary analysis to understand its structure. Each laughter event was categorized by the gender of the speaker (Male or Female) and the role (Caller or Receiver), along with the duration of the laughter in seconds. For each of the research questions, a null hypothesis expressing no difference or effect was defined for assessment. The Chi-Square test was chosen to compare observed counts of laughter events against expected counts, assessing the frequency of laughter in relation to gender and role. The student's t-test was selected to compare the means of laughter duration, suitable for assessing the differences between two independent groups. Due to the absence of a direction in the hypotheses regarding laughter duration, the choice of a two-tailed approach allows for the detection of any significant difference in both directions.

The analysis was performed using Python in a Jupyter Notebook format, with calculations done manually to demonstrate an understanding of the statistical methods. The code includes the dataset exploration and descriptive statistics in terms of mathematical test and visualization plots.

- Is the number of laughter events higher for women than for men?

Research Hypothesis: Women laugh more frequently than men during phone conversations.

Null Hypothesis: There is no difference in the frequency of laughter between women and men.

Data description: Out of 120 speakers, 57 are male and 63 females. The total number of laughter events (observed counts) is the sum of events from both genders, that is 842. The expected count for each gender is calculated based on the proportion of that gender in the total population: $E_i = \frac{\text{Total Count in Category} \times \text{Total Count in Group}}{\text{Total Count in Population}}$.

Expected Females $\frac{63 \times 842}{120} = 442.05$ & Males $\frac{57 \times 842}{120} = 399.95$.

	Occurrences	Expected
Gender		
Female	496	442.05
Male	346	399.95

Table 1: Gender-based laughter frequency

Statistical Test: Chi-Square Goodness-of-Fit test is selected to compare the observed number of laughter events against the expected frequencies under the assumption of no gender difference. The degrees of freedom for the test in this two-category comparison is 1. The decision to use a one-tailed test is based on the focused research hypothesis predicting that women laugh more frequently than men during phone conversations.

$$\chi^2 = \sum_{i=1}^N \frac{(O_i - E_i)^2}{E_i} = \frac{(496 - 442.05)^2}{442.05} + \frac{(346 - 399.95)^2}{399.95} = 13.8617$$

$$dof = \text{no of categories} - 1 = 2 - 1 = 1$$

Results:

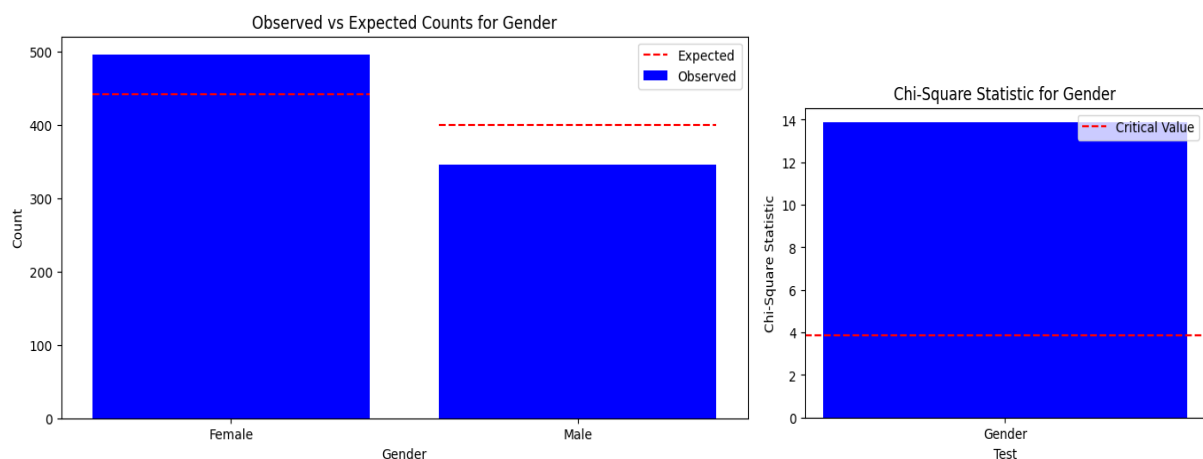


Figure 1: Gender-based laughter frequency bar plots

Since the Chi-Square Statistic $\chi_{obt}^2 = 13.8617$ at 1 degree of freedom is slightly less than the common alpha level $\chi_{0.05}^2 = 3.84$, the null hypothesis can be rejected. This suggests that there is a statistically significant difference in the number of laughter events between women and men, with women subjects laughing more than male ones.

- Is the number of laughter events higher for callers than for receivers?

Research Hypothesis: Callers laugh more frequently than receivers.

Null Hypothesis: There is no difference in the frequency of laughter between callers and receivers.

Data description: Out of 120 speakers, 60 are speakers and 60 are receivers. The total number of laughter events (observed counts) is the sum of events from both roles, that is 842. The expected count for each role is calculated based on the proportion of that role in the total population: $E_i = \frac{\text{Total Count in Category} \times \text{Total Count in Group}}{\text{Total Count in Population}}$.

Expected for both Caller and Receiver $\frac{60 \times 842}{120} = 421$.

	Occurrences	Expected
Gender		
Caller	505	421
Receiver	337	421

Table 2: Role-based laughter frequency

Statistical Test: Chi-Square Goodness-of-Fit test is selected to compare the observed number of laughter events against the expected frequencies under the assumption of no role difference. The degrees of freedom for the test in this two-category comparison is 1. The decision to use a one-tailed test is based on the focused research hypothesis predicting that callers laugh more frequently than receivers during phone conversations.

$$\chi^2 = \sum_{i=1}^N \frac{(O_i - E_i)^2}{E_i} = \frac{(505 - 421)^2 + (337 - 421)^2}{421} = 33.5201$$

$$dof = \text{no of categories} - 1 = 2 - 1 = 1$$

Results:

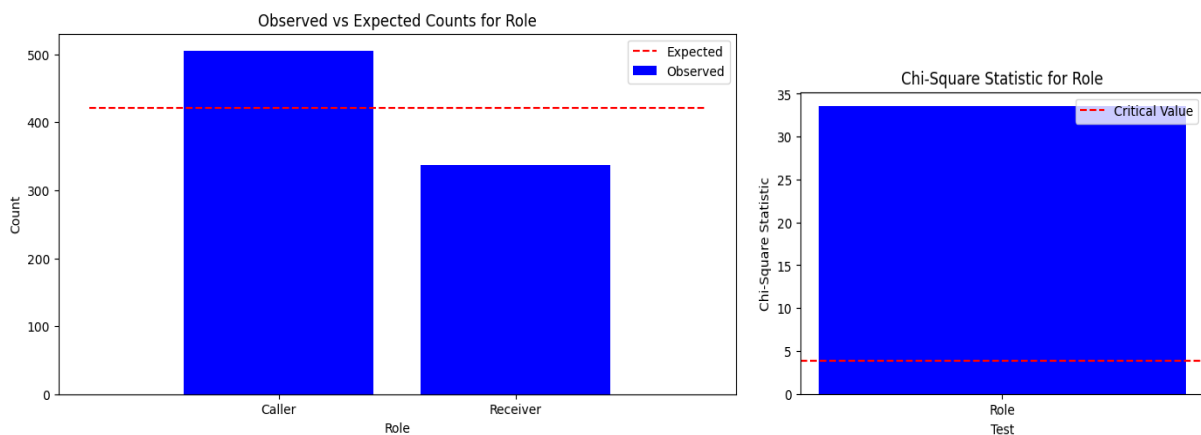


Figure 2: Role-based laughter frequency bar plots

The Chi-Square Statistic $\chi_{obt}^2 = 33.5201$ at 1 degree of freedom, which it is less than common alpha level $\chi_{0.05}^2 = 3.84$, indicates that the difference in the number of laughter events between callers and receivers is statistically significant. The null hypothesis can be rejected and conclude that callers subjects tend to laugh more than receivers ones.

- Are laughter events longer for women?

Research Hypothesis: Laughter events are longer in duration for women compared to men.

Null Hypothesis: There is no difference in the duration of laughter events between women and men.

Data description: The Student's t-test is selected to compare the means and standard deviations of laughter duration, suitable for assessing the differences between two independent groups.

	Mean Duration	Standard Deviation
Gender		
Male	0.606231	0.404662
Female	0.709685	0.441615

Table 3: Gender-based laughter duration

Statistical Test: The t-statistic is a measure of the difference between the two group means in units of the standard error. The formulas below,

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}, \text{ and } dof = \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{\frac{\left(\frac{s_1^2}{n_1}\right)^2}{n_1 - 1} + \frac{\left(\frac{s_2^2}{n_2}\right)^2}{n_2 - 1}}$$

where \bar{X}_1, \bar{X}_2 are the sample means of the two groups, s_1, s_2 are the sample variances (square of the standard deviations) of the two groups and n_1, n_2 are the sample sizes of the two groups are used for manual.

Results:

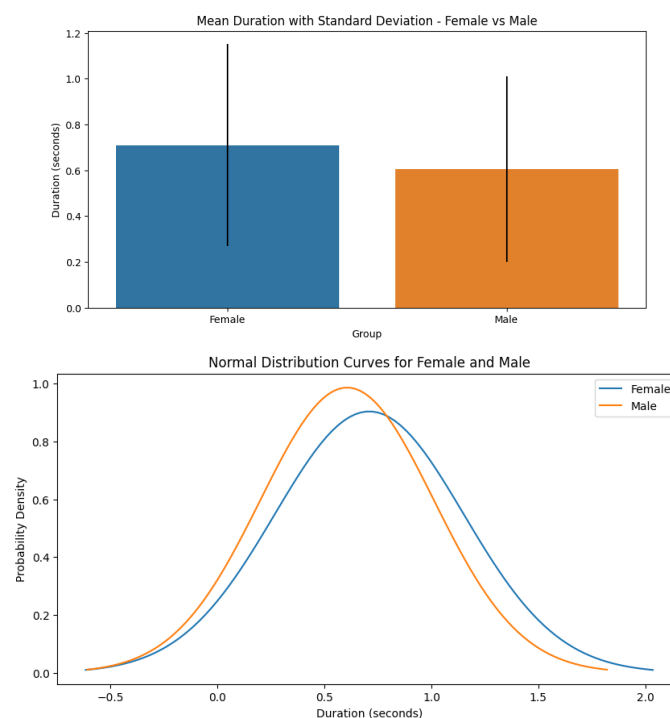


Figure 3: Gender-based laughter duration bar plots and distribution curves

In the process of determining the p-value corresponding to the calculated t-statistic and the degrees of freedom, reference was made to the t-distribution table. For a one-tailed test, the t-statistic of 3.5146, with an approximate degree of freedom (dof) of 781 – (dof) of ∞ would be considered – gives from the table a p-value significantly lower than the conventional alpha level of 0.05 having 1.645, compelling the rejection of the null hypothesis. In the context of a two-tailed test, the p-value derived from the one-tailed test doubled (as both tails are considered from the distribution), as it accounts for both tails of the distribution. Even with this adjustment, given the high t-statistic, the resulting p-value remains below the 0.05 threshold of 1.960. This outcome suggests a statistically significant difference in the duration of laughter events between women and men, that is laughter events are longer for women than for men.

- **Are laughter events longer for callers?**

Research Hypothesis: Laughter events are longer in duration for callers compared to receivers.

Null Hypothesis: There is no difference in the duration of laughter events between callers and receivers.

Data description: The Student's t-test is selected to compare the means and standard deviations of laughter duration, suitable for assessing the differences between two independent groups.

	Mean Duration	Standard Deviation
Gender		
Caller	0.745766	0.461145
Receiver	0.549401	0.346070

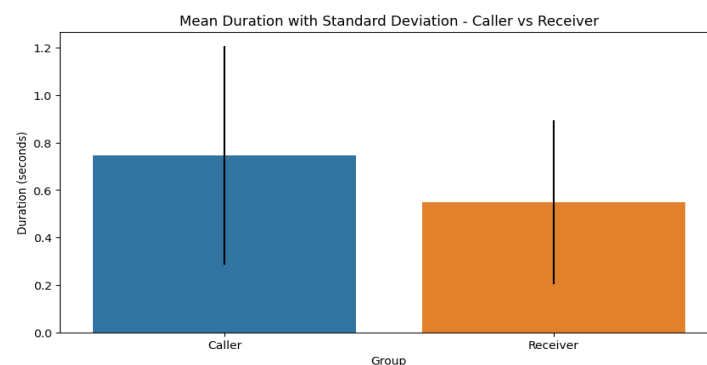
Table 4: Role-based laughter duration

Statistical Test: The t-statistic is a measure of the difference between the two group means in units of the standard error. The formulas below,

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}, \text{ and } dof = \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{\frac{\left(\frac{s_1^2}{n_1}\right)^2}{n_1 - 1} + \frac{\left(\frac{s_2^2}{n_2}\right)^2}{n_2 - 1}}$$

where \bar{X}_1, \bar{X}_2 are the sample means of the two groups, s_1, s_2 are the sample variances (square of the standard deviations) of the two groups and n_1, n_2 are the sample sizes of the two groups are used for manual.

Results:



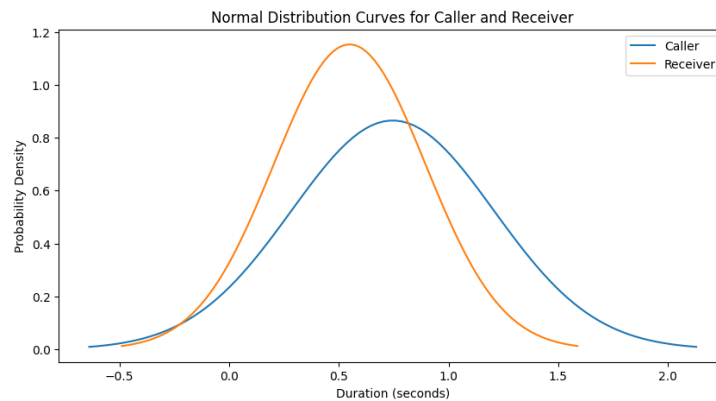


Figure 4: Role-based laughter duration bar plots and distribution curves

For the one-tailed test, the t-statistic of 7.0469, with an approximate degree of freedom (dof) of 829 – (dof) of ∞ would still be considered – gives from the table a p-value significantly lower than the conventional alpha level of 0.05 having 1.645, compelling the rejection of the null hypothesis. In the context of a two-tailed test, the resulting p-value remains below the 0.05 threshold of 1.960 again. This outcome suggests a statistically significant difference in the duration of laughter events between women and men, that laughter events are longer for callers.

Conclusion

This part of the study explored laughter events as a nonverbal communication cue in telephonic interactions, analyzing its frequency and duration relative to participant gender and conversational role (caller or receiver). The statistical analysis, employing both the Chi-Square Goodness-of-Fit test and Student's t-test, offered robust insights into the dynamics of laughter and a better understanding of how it is calculated. Key findings revealed that women laugh more frequently and for longer durations during phone conversations than men, highlighting gender differences in communication styles. Additionally, callers tend to laugh more frequently and for longer periods than receivers, suggesting the influence of conversational dynamic on laughter from the research perspective.

Part 2: Facial Expression Classification

Introduction

Facial expression analysis brings key insights into human emotions, with applications, for instance, in security, entertainment, and mental health. This field faces the challenge of accurately classifying subtle and complex emotions which may raise difficulty in understanding human responses accurately. This part of the study focuses on the computational analysis of two universally recognized expressions: smiles and frowns. The dataset includes 52 feature vectors, each representing activation levels of 17 Action Units (AUs) associated with these expressions. The objective lies around using Gaussian Discriminant Functions to develop a capable classifier that distinguishes between these fundamental expressions. Despite the interdependence of facial muscles, we assume feature independence to simplify our model and avoid covariance complexity, focusing on methodological exploration and practical application in facial expression classification.

Methodology

- **Parameters in Gaussian Discriminant Analysis**

For Gaussian Discriminant Analysis, the mean (μ) and variance (σ^2) define the Gaussian Density Function for each class and feature. This function calculates a probability score for how likely a given feature value is, assuming it comes from a distribution characterized by the resulted means and variances. The formula,

$$P(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

represents the gaussian density function, where 'x' is the feature vector. For each feature vector it is assumed independence to compute the likelihood of the entire feature. The total likelihood for a feature vector belonging to a particular class is the cumulative product of the individual probabilities for each feature. Mathematically, for a vector $x = [x_1, x_2, x_3, \dots, x_n]$, the likelihood for a class is calculated as,

$$L(x|c) = \prod_{i=1}^n P(x_i|\mu_i, \sigma_i^2)$$

Here, $P(x_i|\mu_i, \sigma_i^2)$ represents the probability of the i-th feature given the mean and variance for that feature in a class, also referred as posterior probability. Utilizing Bayes' Rule, the posterior probability is related to the likelihood as follows:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

In practical scenarios, the comparison of the numerator across classes is normal since the denominator $P(x)$ remains constant for all classes. The prior probability of each class, $P(c)$,

can be estimated based on the class frequency in the dataset. For classifying a new feature vector as a decision rule, the procedure involves identifying the highest posterior probability across classes. Often, this classification is simplified to the comparison of the likelihoods multiplied by the class priors, especially when the priors are equal, or the primary goal is to ascertain the most likely class.

• Experimental setup and results

The original dataset comprised 52 feature vectors extracted from face images, each representing the activation levels of 17 distinct Action Units (AUs). The dataset was evenly split into two classes, 'smile' and 'frown', with 26 instances each, for training purposes. The training set would be used to estimate the parameters of the Gaussian models by calculating the mean and variance of the activation levels for each AU within each class. Given the number of features and treating the AUs as independent features conditioned on the class label, the Gaussian models were constructed under the Naive Bayes assumption. For each instance there would be a function that implements the gaussian model from scratch.

The classifier's performance was measured by its error rate over the test set, defined as the proportion of misclassified instances. The final results of the classification process achieved an error rate of 12.5%, indicating that it incorrectly classified 2 out of the 16 instances in the test set. A closer inspection revealed that the misclassifications were not concentrated in one class but distributed between 'smile' and 'frown', suggesting that the classifier did not exhibit a systematic bias toward one particular class. To understand the source of these errors, the instances have been examined along with their respective posterior probabilities, finding that the misclassified instances had feature values that fell within overlapping regions of the distributions for 'smile' and 'frown'.

The distributions for AU06_r and AU26_r as seen in Figure 5 provided insights into the decision-making process of the classifier. For AU06_r, the 'smile' class showed a high frequency of activation at higher values, while the 'frown' class clustered at the lower end. For AU26_r, there was a notable spike for 'frown' around certain values, which may have contributed to the misclassification of a 'smile' instance.

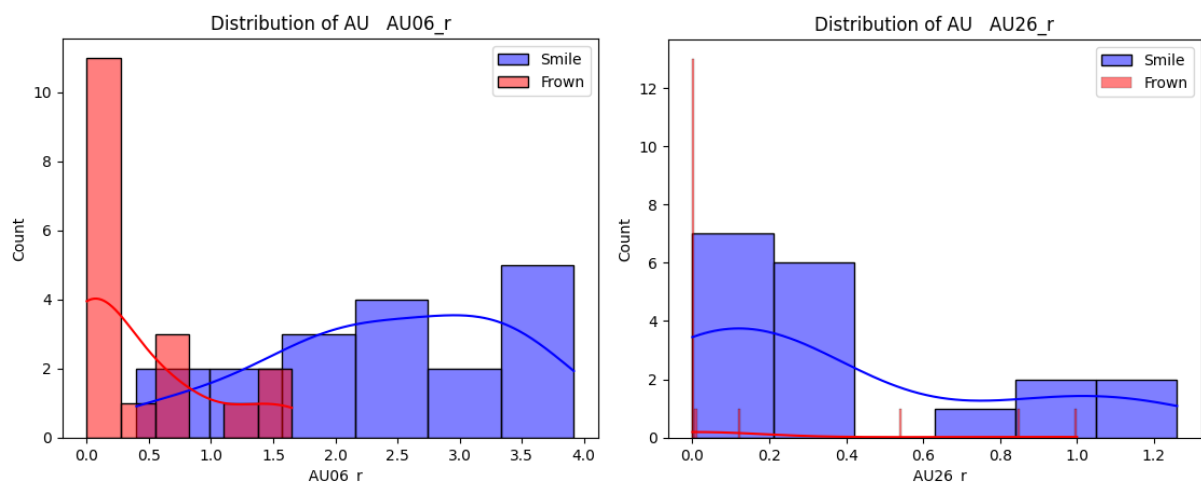


Figure 5: Distribution of the action units that were incorrectly classified

Conclusion

This study's part investigation into the computational analysis of smiles and frowns using Gaussian Discriminant Functions illustrated a significant step in emotion recognition through facial expressions. The approach employed has enabled the development of a manual classifier distinguishing these expressions with a tolerable 12.5% error rate. The observed misclassifications indicate an absence of systematic bias in the classifier and possible limitations of assuming feature independence. Overall, this study affirms the viability of Gaussian Discriminant Functions in facial expression analysis, paving the way for further advancements in emotion recognition technologies.

Appendix

The code snippets of the Jupyter notebook presented below can also be found here:
<https://github.com/AlexandruRO45/CSI>

As a quick note, some of the code such as generating the plots and adjusting some functions to calculate the statistical test has been generated by using GitHub version of Copilot is offers to be a great assistant at helping me fixing logical errors and add certain visualization features to understand the data.

run_csi

December 4, 2023

1 Part 1

```
[ ]: # Import necessary libraries
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns

# Load the dataset
file_path = 'csv/laughter-corpus.csv' # Replace with your file path
data = pd.read_csv(file_path)

# Display the first few rows for a quick overview
data.head()
```

```
[ ]:   Gender      Role  Duration
0  Female    Caller    0.961
1    Male  Receiver    0.630
2  Female    Caller    1.268
3    Male  Receiver    0.146
4  Female    Caller    0.276
```

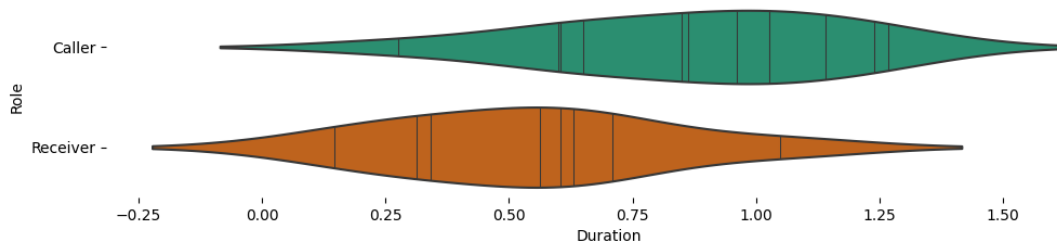
```
[ ]: data.head(20)
```

```
[ ]:   Gender      Role  Duration
0  Female    Caller    0.961
1    Male  Receiver    0.630
2  Female    Caller    1.268
3    Male  Receiver    0.146
4  Female    Caller    0.276
5    Male  Receiver    0.562
6  Female    Caller    1.141
7  Female    Caller    0.600
8  Female    Caller    1.239
9  Female    Caller    0.850
10   Male  Receiver    0.605
11  Female    Caller    1.026
```

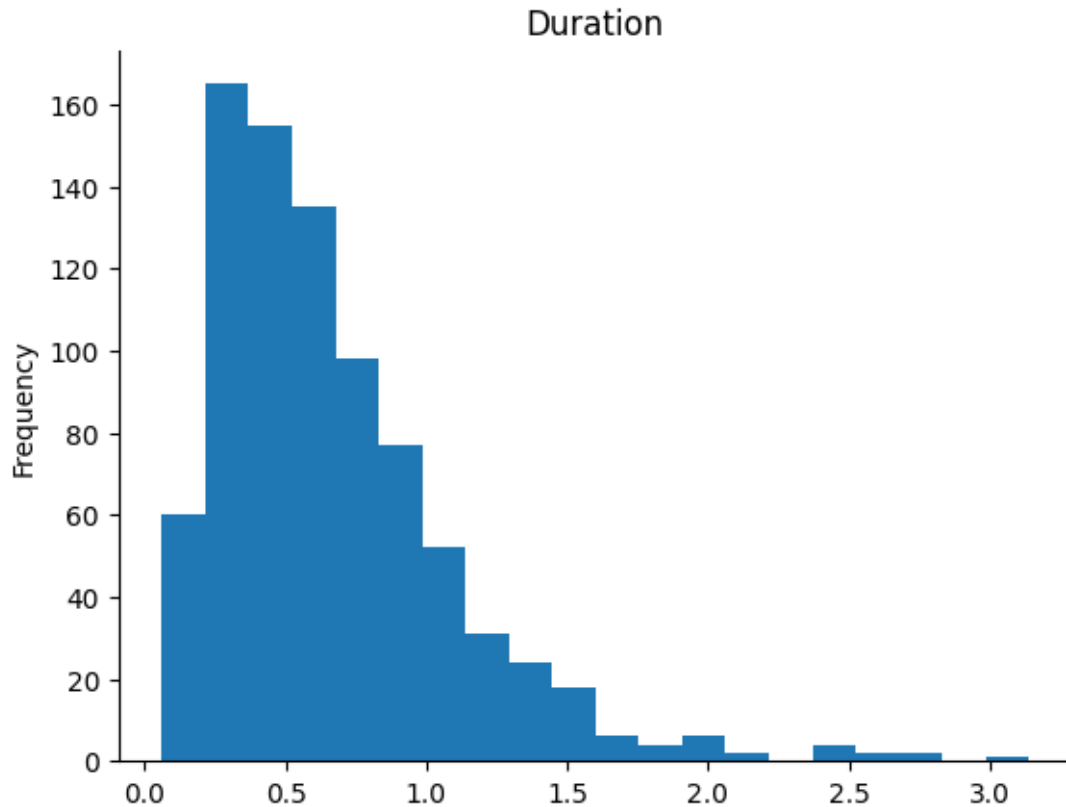
12	Male	Receiver	0.314
13	Female	Caller	1.026
14	Female	Caller	0.605
15	Male	Receiver	0.710
16	Female	Caller	0.862
17	Male	Receiver	0.341
18	Female	Caller	0.651
19	Male	Receiver	1.048

```
[ ]: figsize = (12, 1.2 * len(data['Role'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(data.head(20), x='Duration', y='Role', inner='stick',
               palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)
```

```
/home/alex/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
    if pd.api.types.is_categorical_dtype(vector):
/home/alex/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
    if pd.api.types.is_categorical_dtype(vector):
/home/alex/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498:
FutureWarning: is_categorical_dtype is deprecated and will be removed in a
future version. Use isinstance(dtype, CategoricalDtype) instead
    if pd.api.types.is_categorical_dtype(vector):
```



```
[ ]: data['Duration'].plot(kind='hist', bins=20, title='Duration')
plt.gca().spines[['top', 'right']].set_visible(False)
```



1.1 Chi-Square Test for Count Data (Questions 1 and 2)

- Question 1: Number of Laughter Events - Women vs. Men Is the number of laughter events higher for women than for men? H1: Women have a higher number of laughter events than men. H0: There is no significant difference in the number of laughter events between women and men.
 - Observed Frequencies: Counts of laughter events for each gender.
 - Expected Frequencies: Assuming no gender difference, we expect the counts to be proportional to the number of speakers of each gender.
- Question 2: Number of Laughter Events - Callers vs. Receivers Is the number of laughter events higher for callers than for receivers? H1: Callers have a higher number of laughter events than receivers. H0: There is no significant difference in the number of laughter events between callers and receivers.
 - Observed Frequencies: Counts of laughter events for each role.
 - Expected Frequencies: Assuming no role difference, we expect the counts to be equal for callers and receivers.

```
[ ]: # Count the number of laughter events for each gender and role
gender_counts = data['Gender'].value_counts()
role_counts = data['Role'].value_counts()
```

```
[ ]: # Occurance of gender
pd.DataFrame(gender_counts)
```

```
[ ]:          count
Gender
Female    496
Male      346
```

```
[ ]: # Occurance of role
pd.DataFrame(role_counts)
```

```
[ ]:          count
Role
Caller    505
Receiver   337
```

```
[ ]: # Creating contingency tables for Chi-Square tests
gender_cross_tab = pd.DataFrame({
    'Count': gender_counts, # Count Occurances
    'Total': [63, 57] # Total number of female and male speakers (in this order)
})
role_cross_tab = pd.DataFrame({
    'Count': role_counts, # Count Occurances
    'Total': [60, 60] # Total number of callers and receivers (in this order)
})
total = gender_cross_tab['Count'].values.sum()
total_pop = gender_cross_tab['Total'].values.sum()
```

```
[ ]: gender_cross_tab
```

```
[ ]:          Count  Total
Gender
Female    496      63
Male      346      57
```

```
[ ]: role_cross_tab
```

```
[ ]:          Count  Total
Role
Caller    505      60
Receiver   337      60
```

```
[ ]: # Function to calculate Chi-Square statistic
def chi_square_calc(observed, expected):
    return np.sum((observed - expected) ** 2 / expected)

# Function to calculate degrees of freedom for Chi-Square test
```

```
def chi_square_dof(observed):
    return (observed.shape[0] - 1) * (observed.shape[1] - 1)
```

```
[ ]: # Manual calculation of the Chi-Square statistic for gender

# Expected counts assuming no gender difference
expected_counts_gender = [gender_cross_tab['Total'].iloc[0] / total_pop * total,
                           gender_cross_tab['Total'].iloc[1] / total_pop * total,
                           ]

# Create a dataframe with gender counts and expected counts gender
gender_counts_df = pd.DataFrame({
    'Occurrences': gender_counts,
    'Expected': expected_counts_gender
})
print(gender_counts_df)

# Chi-Square statistic calculation
chi2_gender = chi_square_calc(gender_counts, expected_counts_gender)
chi2_dof_gender = chi_square_dof(gender_cross_tab)
print(f"\nQ1:Chi-Square Statistic = {chi2_gender}\nQ1:Degrees of Freedom = \u2192{chi2_dof_gender}")
```

	Occurrences	Expected
Gender		
Female	496	442.05
Male	346	399.95

Q1:Chi-Square Statistic = 13.861744622839753
 Q1:Degrees of Freedom = 1

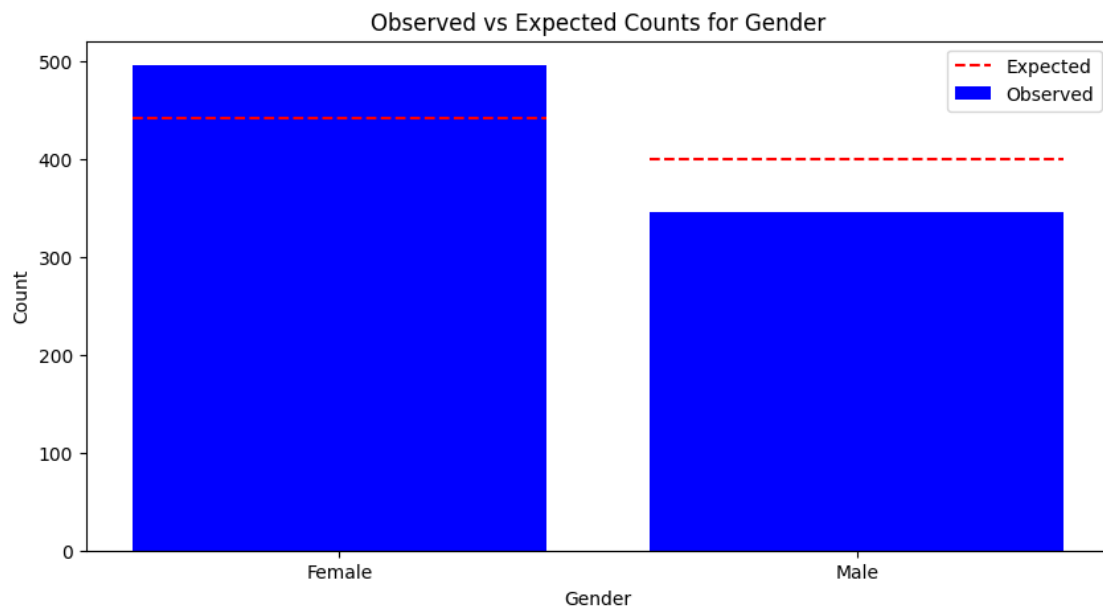
```
[ ]: # Plotting the observed and expected counts for gender
plt.figure(figsize=(10, 5))
observed_bars = plt.bar(gender_counts_df.index,\u2192
    \u2192gender_counts_df['Occurrences'], label='Observed', color='blue')
bar_width = observed_bars[0].get_width()
x_positions = [bar.get_x() + bar_width/2 for bar in observed_bars]
plt.hlines(y=gender_counts_df['Expected'].iloc[0], xmin=x_positions[0] - \u2192
    \u2192bar_width/2, xmax=x_positions[0] + bar_width/2, color='red', \u2192
    \u2192linestyles='--', label='Expected')
plt.hlines(y=gender_counts_df['Expected'].iloc[1], xmin=x_positions[1] - \u2192
    \u2192bar_width/2, xmax=x_positions[1] + bar_width/2, color='red', linestyle='--')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.title('Observed vs Expected Counts for Gender')
plt.legend()
plt.show()
```

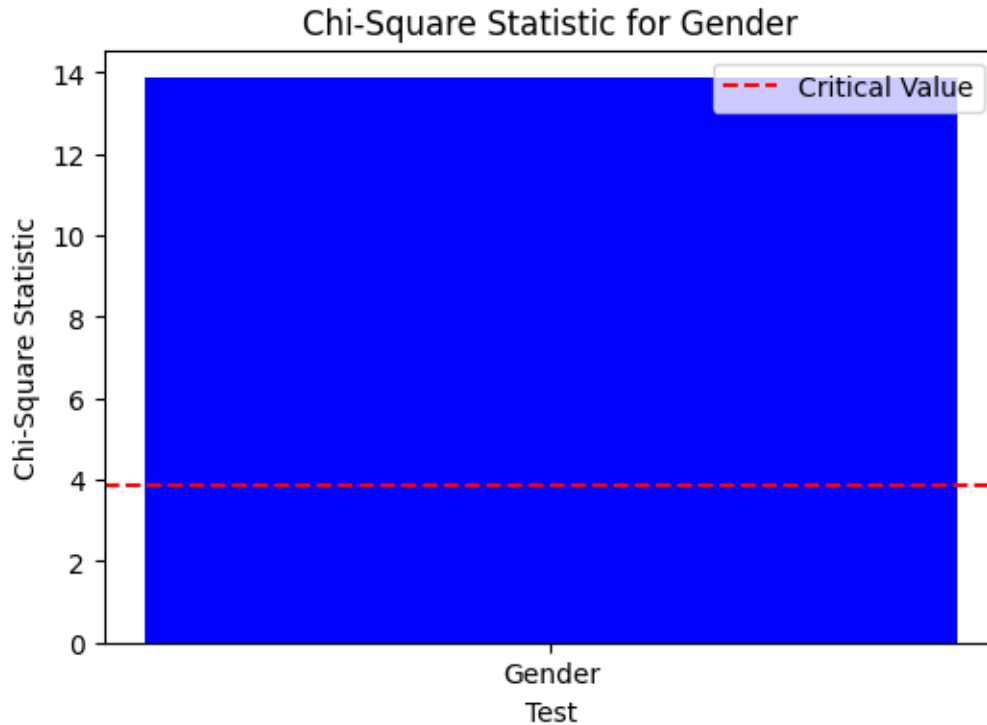


```

# Plotting the Chi-Square statistic for gender
plt.figure(figsize=(6, 4))
plt.bar(['Gender'], [chi2_gender], color='blue')
plt.axhline(y=3.84, color='red', linestyle='--', label='Critical Value')
plt.xlabel('Test')
plt.ylabel('Chi-Square Statistic')
plt.title('Chi-Square Statistic for Gender')
plt.legend()
plt.show()

```





Since the Chi-Square Statistic is 13.861744622839753 at 1 degree of freedom is less than the common alpha level of 0.05 at 3.84, we can reject the null hypothesis (H_0). This suggests that there is a statistically significant difference in the number of laughter events between women and men, with women having more laughter events.

```
[ ]: # Manual calculation of the Chi-Square statistic for role

# Expected counts assuming no role difference
expected_counts_role = [role_cross_tab['Total'].iloc[0] / total_pop * total,
                        role_cross_tab['Total'].iloc[1] / total_pop * total,
                        ]

# Create a dataframe with role counts and expected counts role
role_counts_df = pd.DataFrame({
    'Occurrences': role_counts,
    'Expected': expected_counts_role
})
print(role_counts_df)

# Chi-Square statistic calculation
chi2_role = chi_square_calc(role_counts, expected_counts_role)
chi2_dof_role = chi_square_dof(role_cross_tab)
```

```
# Display the Chi-Square statistic for gender
print(f"\nQ1:Chi-Square Statistic = {chi2_role}\nQ1:Degrees of Freedom =\n{chi2_dof_role}")
```

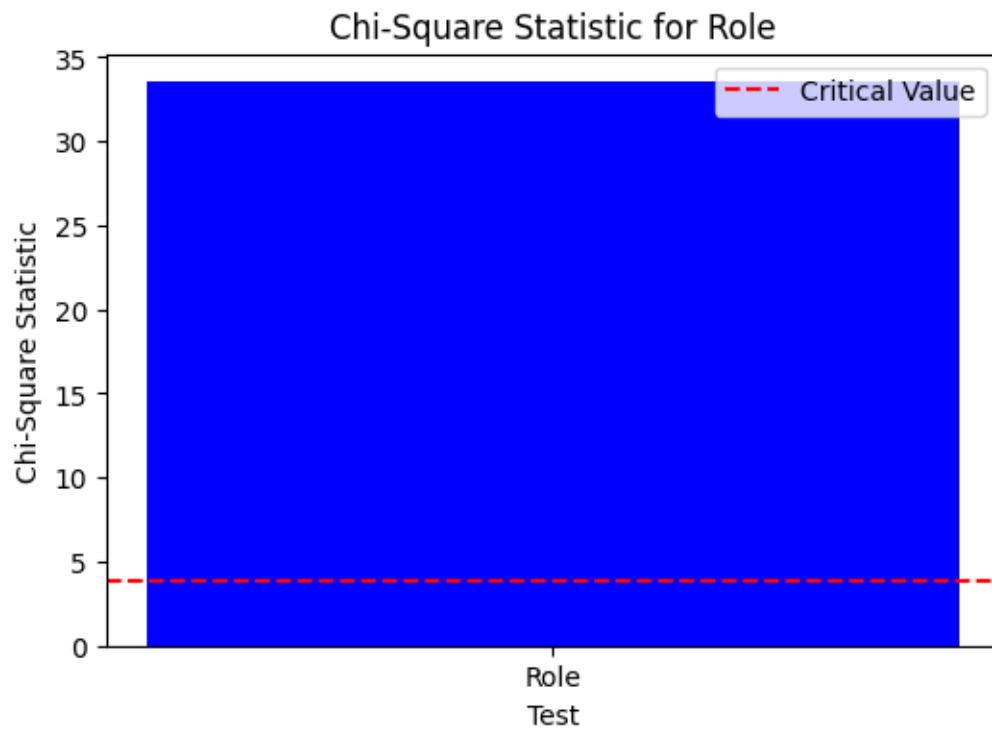
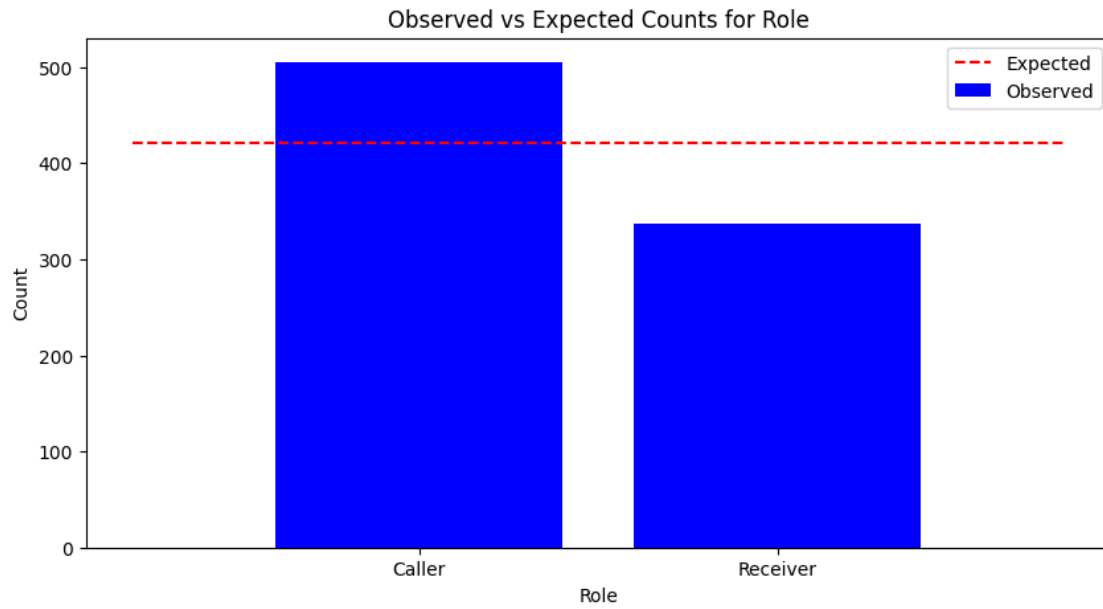
	Occurrences	Expected
Role		
Caller	505	421.0
Receiver	337	421.0

Q1:Chi-Square Statistic = 33.52019002375297

Q1:Degrees of Freedom = 1

```
[ ]: # Plotting the observed and expected counts for role
plt.figure(figsize=(10, 5))
observed_bars = plt.bar(role_counts_df.index, role_counts_df['Occurrences'],
    label='Observed', color='blue')
bar_width = observed_bars[0].get_width()
x_positions = [bar.get_x() + bar_width/2 for bar in observed_bars]
plt.hlines(y=role_counts_df['Expected'].iloc[0], xmin=x_positions[0],
    bar_width, xmax=x_positions[-1] + bar_width, color='red', label='Expected',
    linestyle='--')
plt.xlabel('Role')
plt.ylabel('Count')
plt.title('Observed vs Expected Counts for Role')
plt.legend()
plt.show()

# Plotting the Chi-Square statistic for role
plt.figure(figsize=(6, 4))
plt.bar(['Role'], [chi2_role], color='blue')
plt.axhline(y=3.84, color='red', linestyle='--', label='Critical Value')
plt.xlabel('Test')
plt.ylabel('Chi-Square Statistic')
plt.title('Chi-Square Statistic for Role')
plt.legend()
plt.show()
```



The Chi-Square Statistic is 33.52019002375297 at 1 degree of freedom, which is slightly less than 0.05 at 3.84, indicates that the difference in the number of laughter events between callers and receivers is statistically significant. We can reject the null hypothesis and conclude that callers

tend to have more laughter events than receivers.

1.2 Student's t-Test for Continuous Data (Questions 3 and 4)

3. Question 3: Duration of Laughter Events - Women vs. Men Are laughter events longer for women? H1: Laughter events are longer for women than for men. H0: There is no significant difference in the duration of laughter events between women and men.

- Sample Means and Variances: Calculated from the duration data for each gender.

4. Question 4: Duration of Laughter Events - Callers vs. Receivers Are laughter events longer for callers? H1: Laughter events are longer for callers than for receivers. H0: There is no significant difference in the duration of laughter events between callers and receivers.

- Sample Means and Variances: Calculated from the duration data for each role.

```
[ ]: # Separate data by gender and role for duration analysis
female_duration = data[data['Gender'] == 'Female']['Duration']
male_duration = data[data['Gender'] == 'Male']['Duration']
caller_duration = data[data['Role'] == 'Caller']['Duration']
receiver_duration = data[data['Role'] == 'Receiver']['Duration']

[ ]: # Function to calculate t-statistic for independent samples
def t_test_independent(sample1, sample2):
    mean1, mean2 = np.mean(sample1), np.mean(sample2)
    std1, std2 = np.std(sample1, ddof=1), np.std(sample2, ddof=1)
    n1, n2 = len(sample1), len(sample2)

    pooled_se = np.sqrt(std1**2/n1 + std2**2/n2)
    t_statistic = (mean1 - mean2) / pooled_se
    df = (std1**2/n1 + std2**2/n2)**2 / ((std1**2/n1)**2/(n1-1) + (std2**2/
    ↪n2)**2/(n2-1))
    return t_statistic, df

[ ]: # Calculate mean and standard deviation for each group and store them in ↵
    ↪variables
female_mean, female_std = np.mean(female_duration), np.std(female_duration, ↵
    ↪ddof=1)
male_mean, male_std = np.mean(male_duration), np.std(male_duration, ddof=1)
caller_mean, caller_std = np.mean(caller_duration), np.std(caller_duration, ↵
    ↪ddof=1)
receiver_mean, receiver_std = np.mean(receiver_duration), np.
    ↪std(receiver_duration, ddof=1)

# Create dataframes for male and female
gender_stats = pd.DataFrame({
    'Gender': ['Male', 'Female'],
    'Mean Duration': [male_mean, female_mean],
    'Standard Deviation': [male_std, female_std]
```

```

})

# Create dataframes for caller and receiver
role_stats = pd.DataFrame({
    'Role': ['Caller', 'Receiver'],
    'Mean Duration': [caller_mean, receiver_mean],
    'Standard Deviation': [caller_std, receiver_std]
})

# Display the dataframes
print(gender_stats.to_string(index=False))
print()
print(role_stats.to_string(index=False))

```

Gender	Mean Duration	Standard Deviation
Male	0.606231	0.404662
Female	0.709685	0.441615

Role	Mean Duration	Standard Deviation
Caller	0.745766	0.461145
Receiver	0.549401	0.346070

```

[ ]: # Calculating t-statistic and degrees of freedom for gender and role
t_statistic_gender, df_gender = t_test_independent(female_duration,
    ↪male_duration)
t_statistic_role, df_role = t_test_independent(caller_duration,
    ↪receiver_duration)

# Display results
print(f"T-Test for Gender Duration (t-statistic): {t_statistic_gender}, Degrees of
    ↪Freedom: {df_gender}")
print(f"T-Test for Role Duration (t-statistic): {t_statistic_role}, Degrees of
    ↪Freedom: {df_role}")

```

T-Test for Gender Duration (t-statistic): 3.5145791170880627, Degrees of Freedom: 780.7755366076725
T-Test for Role Duration (t-statistic): 7.046925950205163, Degrees of Freedom: 828.5128834922407

```

[ ]: import warnings
warnings.filterwarnings('ignore', category=FutureWarning)

# Plotting bar graphs for mean and standard deviation - Female vs Male
plt.figure(figsize=(10, 5))
sns.barplot(x=['Female', 'Male'], y=[female_mean, male_mean], yerr=[female_std,
    ↪male_std])
plt.title('Mean Duration with Standard Deviation - Female vs Male')

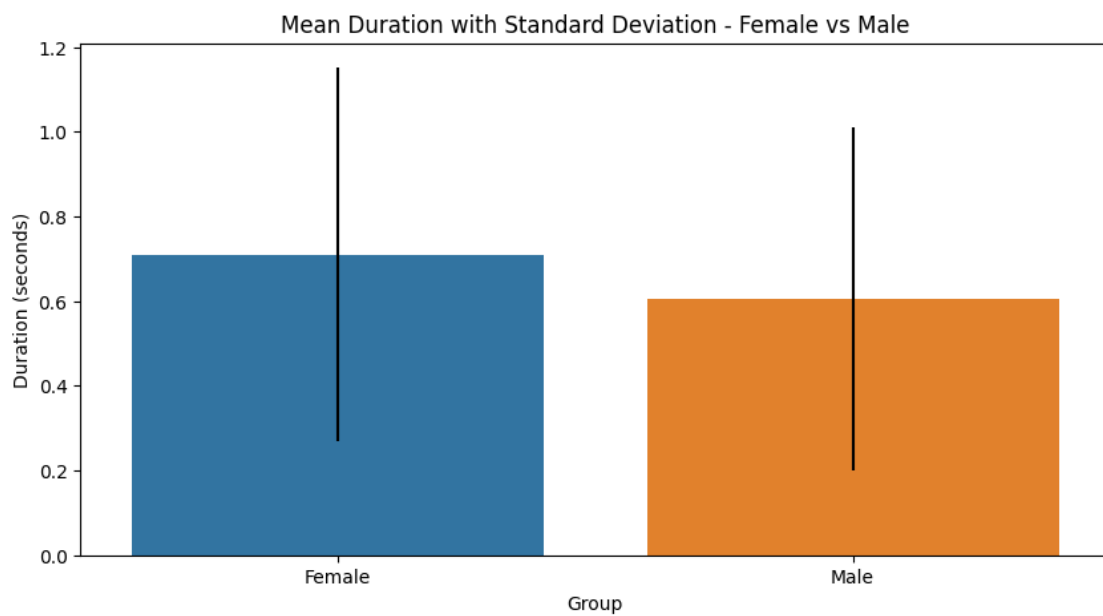
```

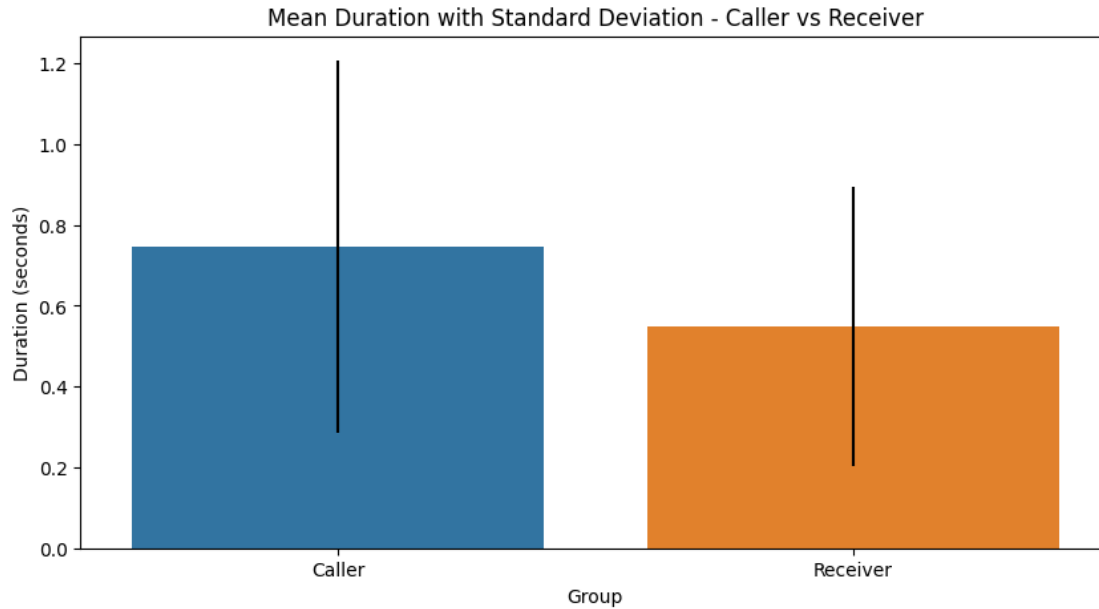
```

plt.xlabel('Group')
plt.ylabel('Duration (seconds)')
plt.show()

# Plotting bar graphs for mean and standard deviation - Caller vs Receiver
plt.figure(figsize=(10, 5))
sns.barplot(x=['Caller', 'Receiver'], y=[caller_mean, receiver_mean],
            yerr=[caller_std, receiver_std])
plt.title('Mean Duration with Standard Deviation - Caller vs Receiver')
plt.xlabel('Group')
plt.ylabel('Duration (seconds)')
plt.show()

```





```
[ ]: from scipy.stats import norm # Import normal distribution function only to
      graph my results

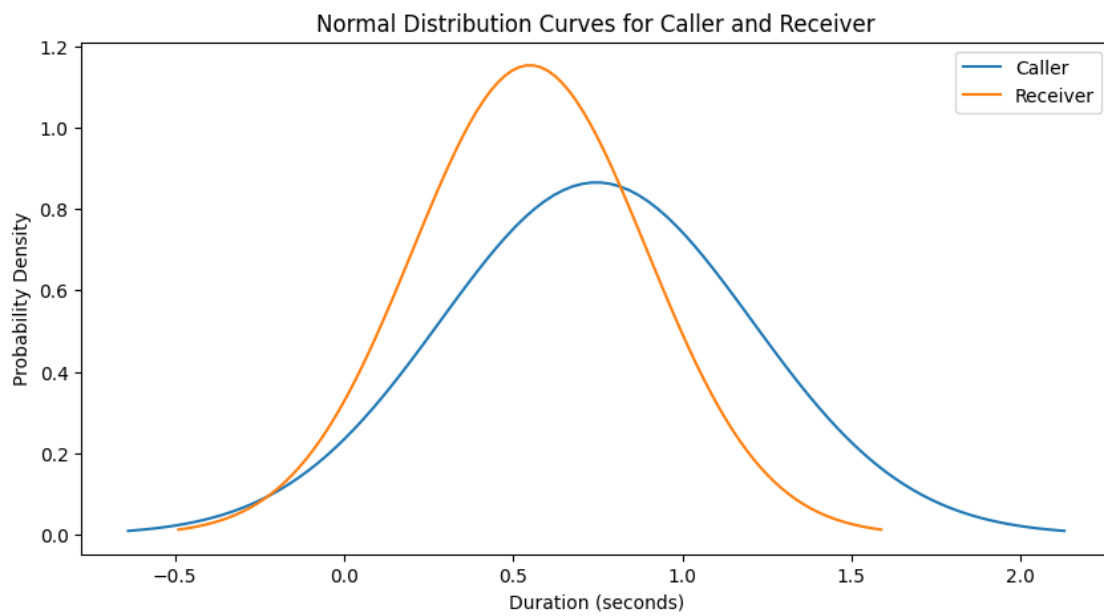
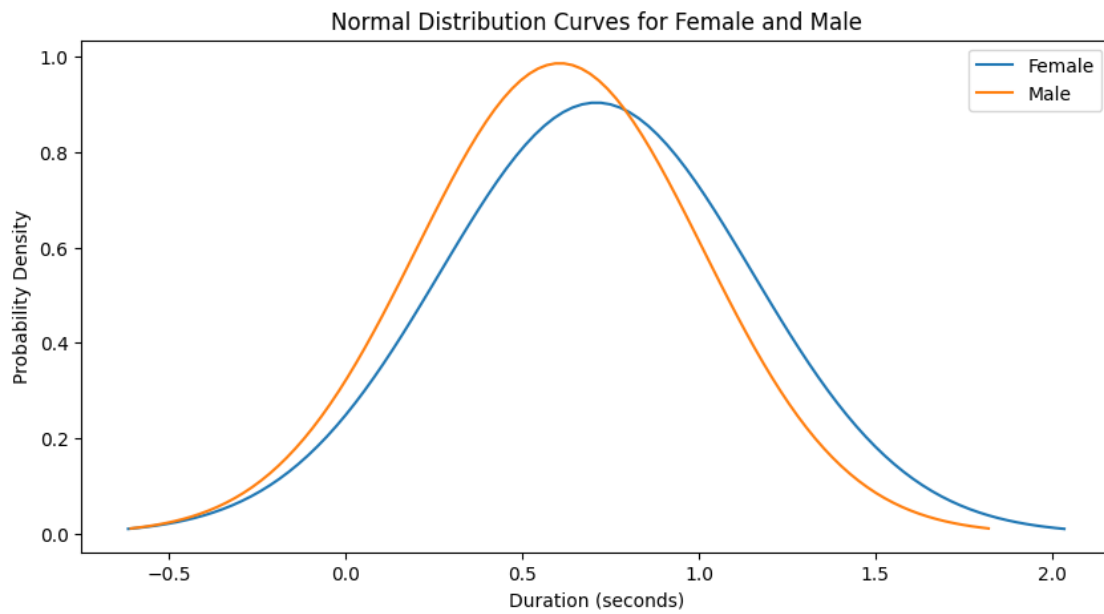
# Function to plot normal distribution
def plot_normal_distribution(mean, std, label):
    # Generate a range of values
    x = np.linspace(mean - 3*std, mean + 3*std, 100)
    # Calculate the normal distribution values
    y = norm.pdf(x, mean, std)
    plt.plot(x, y, label=label)

# Plotting normal distribution curves for female and male
plt.figure(figsize=(10, 5))
plot_normal_distribution(female_mean, female_std, 'Female')
plot_normal_distribution(male_mean, male_std, 'Male')
plt.title('Normal Distribution Curves for Female and Male')
plt.xlabel('Duration (seconds)')
plt.ylabel('Probability Density')
plt.legend()
plt.show()

# Plotting normal distribution curves for caller and receiver
plt.figure(figsize=(10, 5))
plot_normal_distribution(caller_mean, caller_std, 'Caller')
plot_normal_distribution(receiver_mean, receiver_std, 'Receiver')
plt.title('Normal Distribution Curves for Caller and Receiver')
plt.xlabel('Duration (seconds)')
```



```
plt.ylabel('Probability Density')
plt.legend()
plt.show()
```



2 Part 2

```
[ ]: # Load data
train_data = pd.read_csv("csv/training-part-2.csv")
test_data = pd.read_csv("csv/test-part-2.csv")

[ ]: # Separating features and labels in both training and test datasets
X_train = train_data.drop('Class', axis=1)      # Features for training
y_train = train_data['Class']                  # Labels for training
X_test = test_data.drop('Class', axis=1)        # Features for testing
y_test = test_data['Class']                    # Labels for testing

def calculate_parameters(features, labels):
    ''' Calculates mean and variance for each class in the dataset'''
    classes = labels.unique()
    parameters = {}
    for cls in classes:
        class_features = features[labels == cls]
        parameters[cls] = {
            'mean': class_features.mean(),
            'variance': class_features.var()
        }
    return parameters

[ ]: def gaussian_pdf(x, mean, variance):
    """ Gaussian Probability Density Function """
    exponent = np.exp(-(x - mean) ** 2 / (2 * variance))
    return (1 / np.sqrt(2 * np.pi * variance)) * exponent

def classify(features, parameters):
    """ Classify each instance in features """
    classes = list(parameters.keys())
    likelihoods = {cls: 0 for cls in classes}
    for cls in classes:
        likelihoods[cls] = np.prod(gaussian_pdf(features,
                                                    parameters[cls]['mean'],
                                                    parameters[cls]['variance']))
    return max(likelihoods, key=likelihoods.get)

[ ]: def calculate_error_rate(predictions, actual_labels):
    """ Calculate error rate for predictions """
    incorrect_predictions = (predictions != actual_labels).sum()
    total_predictions = len(actual_labels)
    error_rate = incorrect_predictions / total_predictions
    return error_rate * 100 # To get the percentage

def predict_row(row, parameters):
```

```

""" Classify a single row of features """
return classify(row, parameters)

```

```

[ ]: # Calculating mean and variance for each class in the training set
parameters = calculate_parameters(X_train, y_train)

# Predicting class for each instance in the test set
predictions = []
for index, row in X_test.iterrows():
    predicted_class = predict_row(row, parameters)
    predictions.append(predicted_class)

# Convert predictions to a Pandas Series for easy comparison
predictions = pd.Series(predictions, index=X_test.index)

# Calculate error rate and display predictions vs actual labels
error_rate = calculate_error_rate(predictions, y_test)
comparison_df = pd.DataFrame({'Predictions': predictions, 'Actual': y_test})

# Highlight when a prediction doesn't match an actual
comparison_df['Match'] = comparison_df['Predictions'] == comparison_df['Actual']
comparison_df['Match'] = comparison_df['Match'].apply(lambda x: 'Yes' if x else
    ↪ 'No')

print(f"Error Rate: {error_rate}%")
print(comparison_df.to_string(index=False))

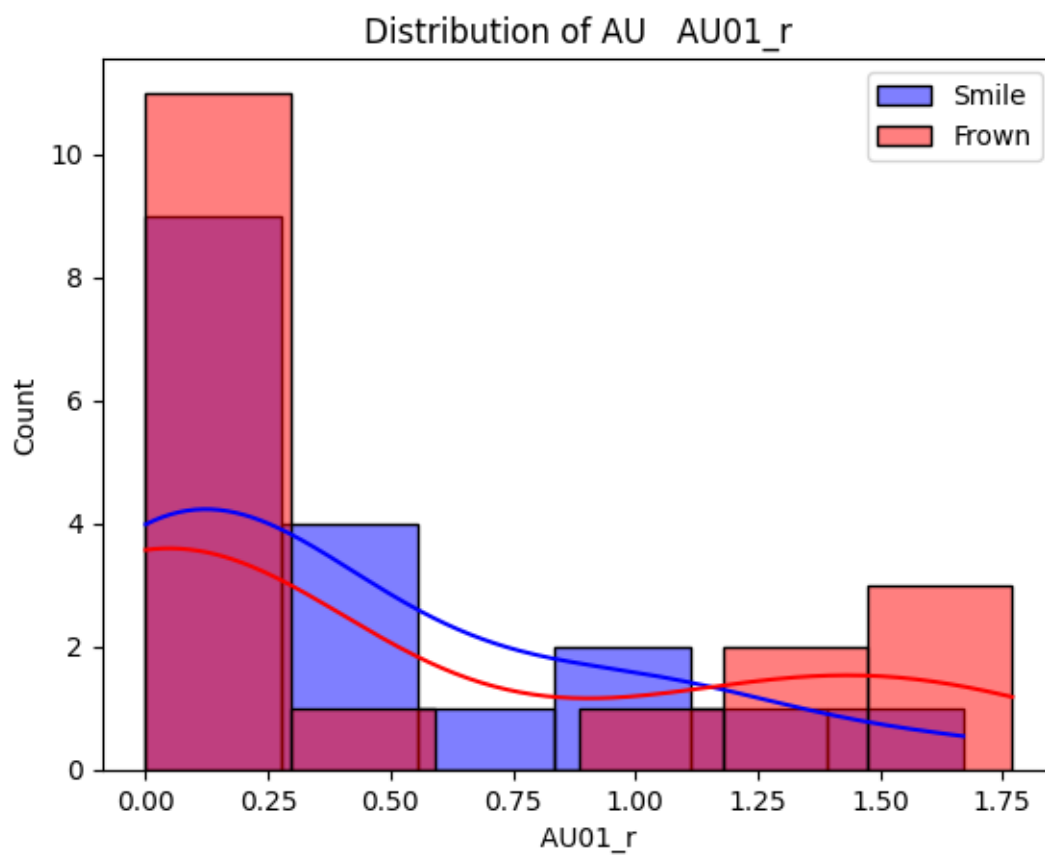
```

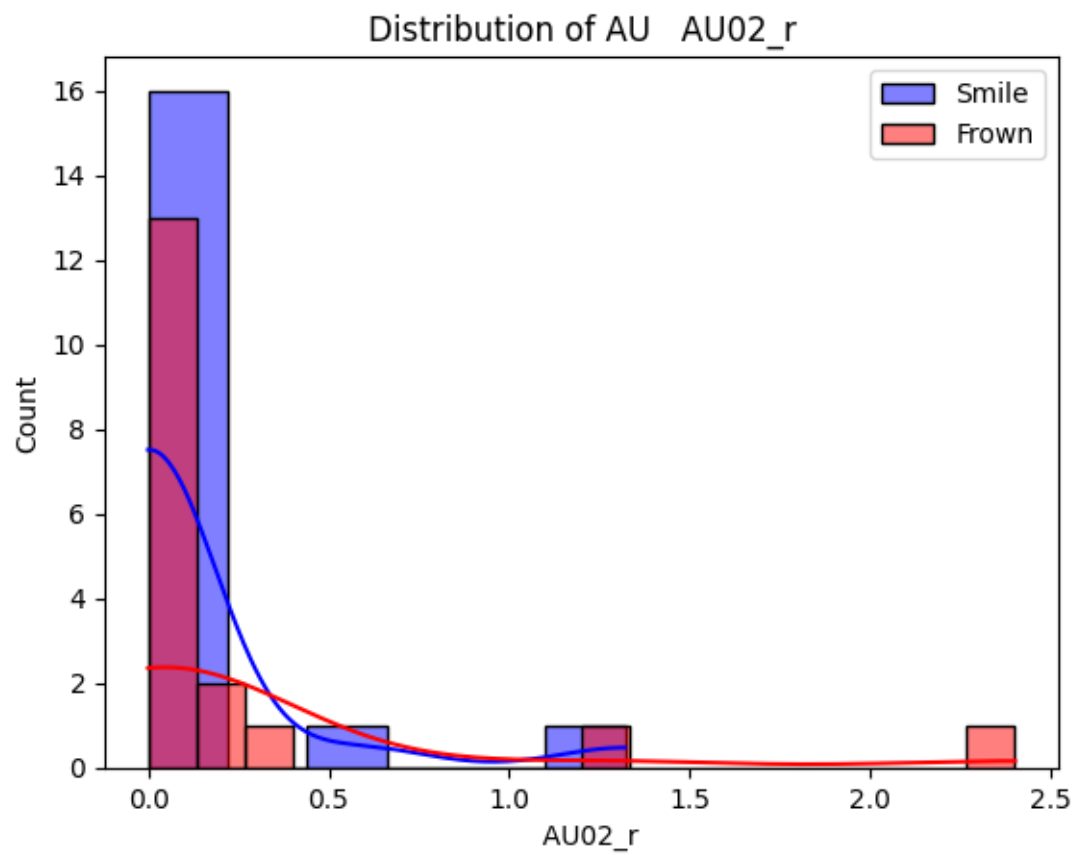
Error Rate: 12.5%

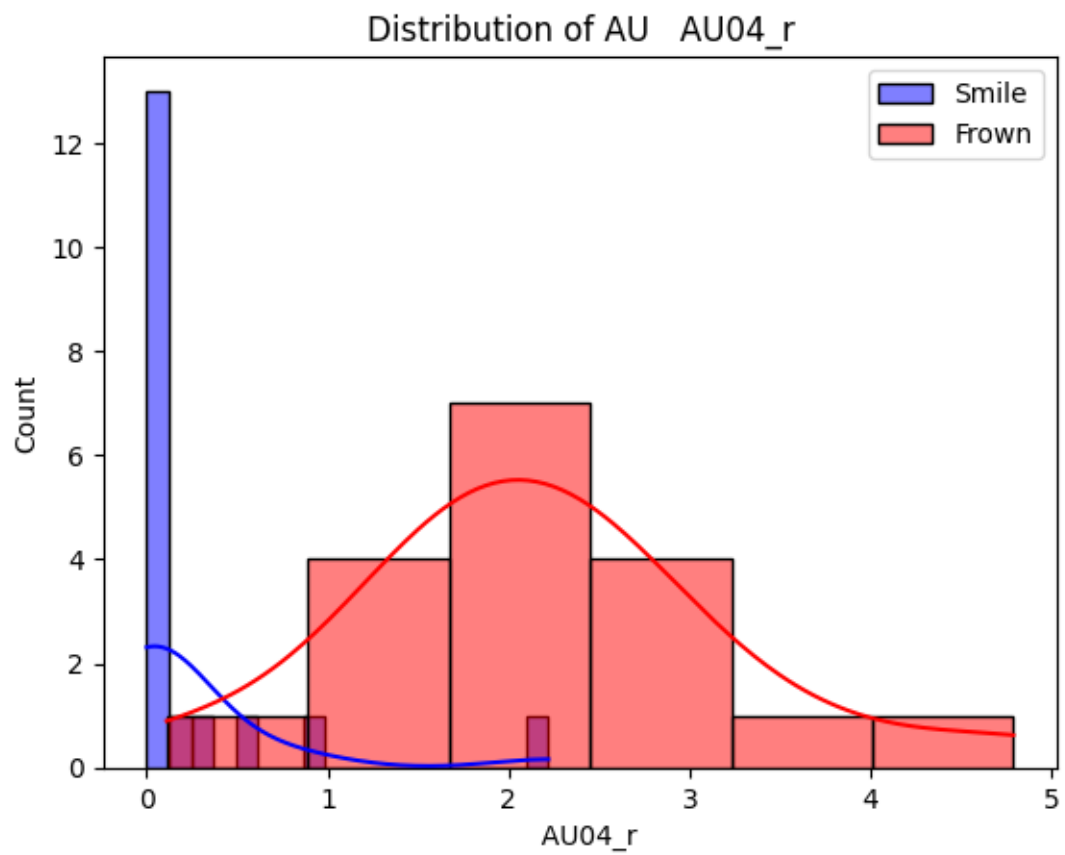
Predictions	Actual	Match
frown	frown	Yes
frown	frown	Yes
frown	frown	Yes
frown	frown	Yes
smile	frown	No
frown	frown	Yes
frown	frown	Yes
frown	frown	Yes
smile	smile	Yes
smile	smile	Yes
smile	smile	Yes
smile	smile	Yes
smile	smile	Yes
smile	smile	Yes
frown	smile	No
smile	smile	Yes

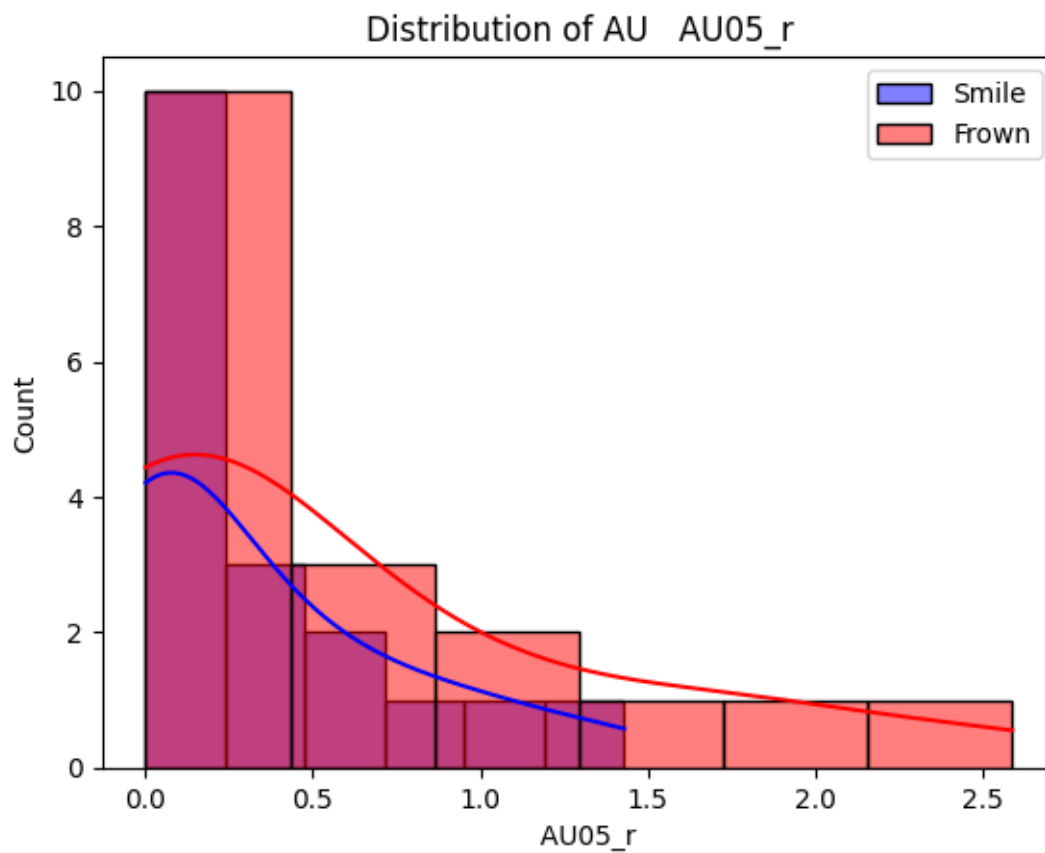
```
[ ]: import warnings
warnings.filterwarnings('ignore', category=FutureWarning)

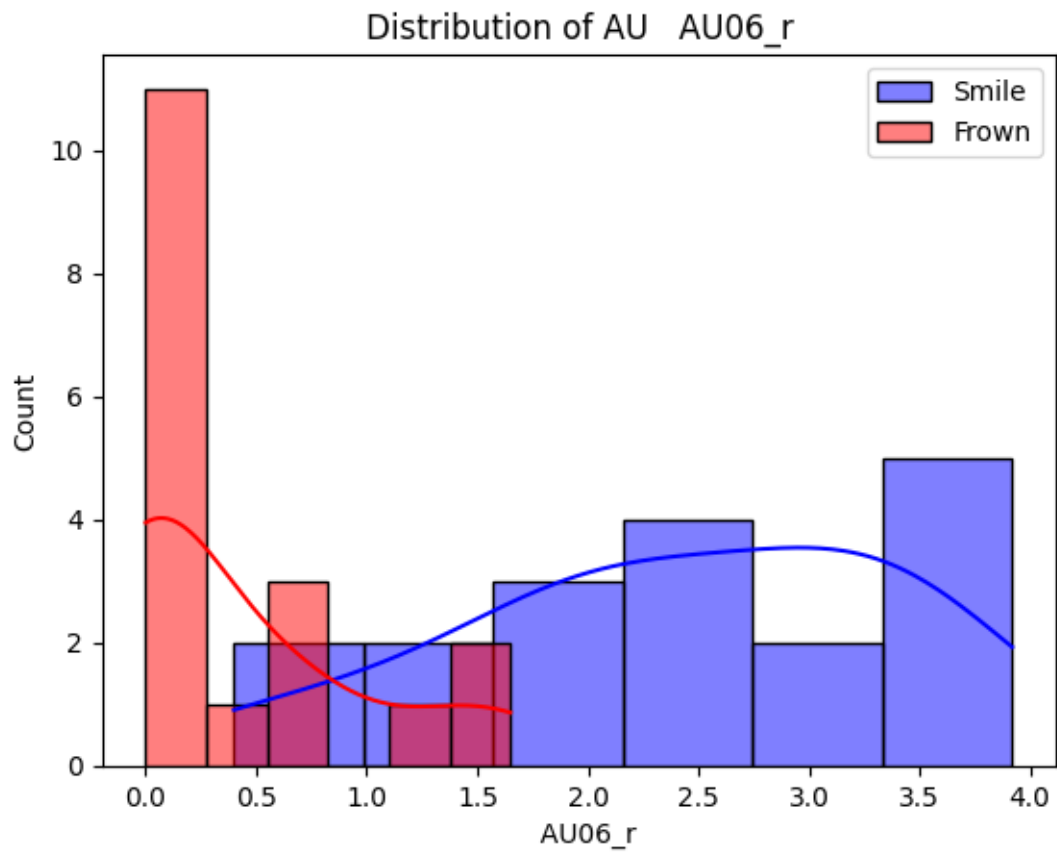
# Histograms/Density Plots
for au in X_train.columns:
    sns.histplot(X_train[y_train == 'smile'][au], color='blue', label='Smile',
    ↪kde=True)
    sns.histplot(X_train[y_train == 'frown'][au], color='red', label='Frown',
    ↪kde=True)
    plt.title(f"Distribution of AU {au}")
    plt.legend()
    plt.show()
```

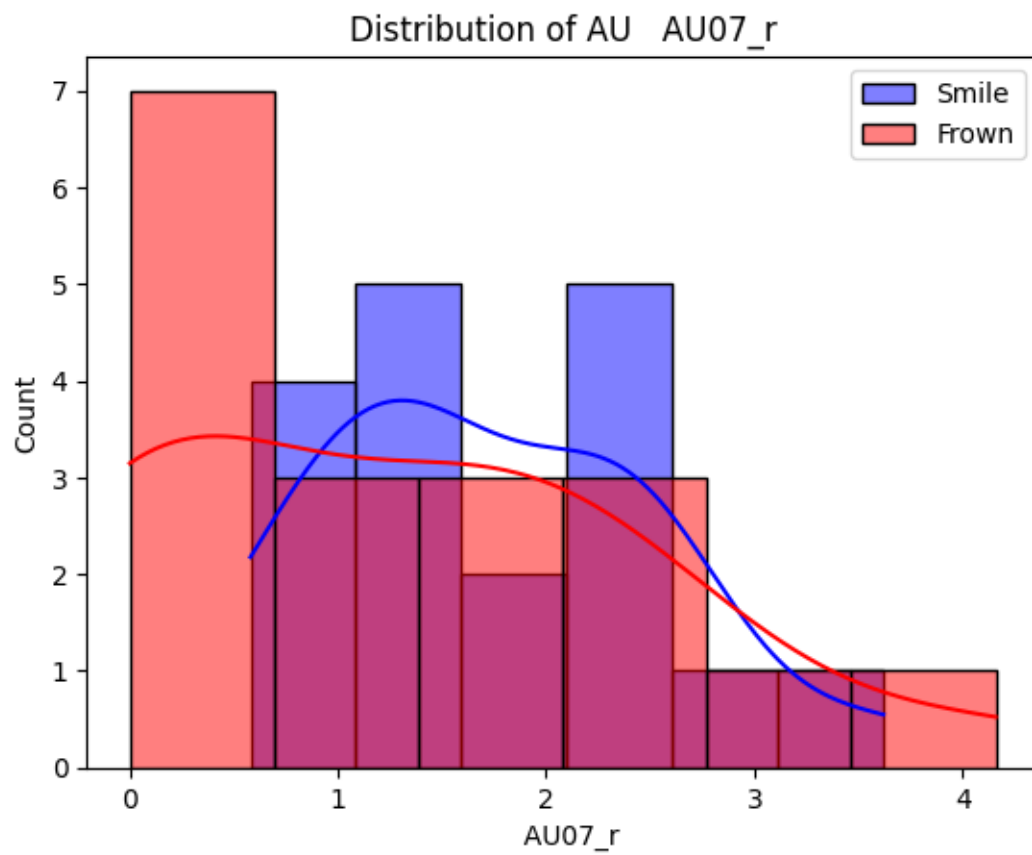


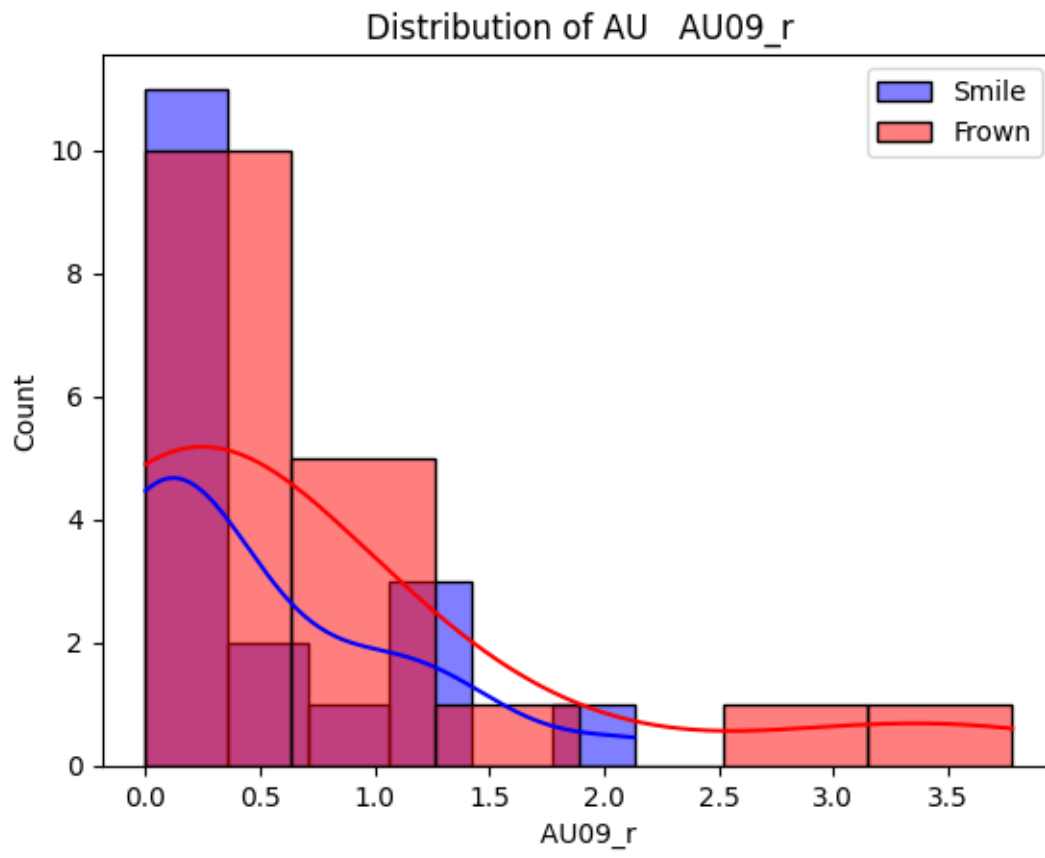


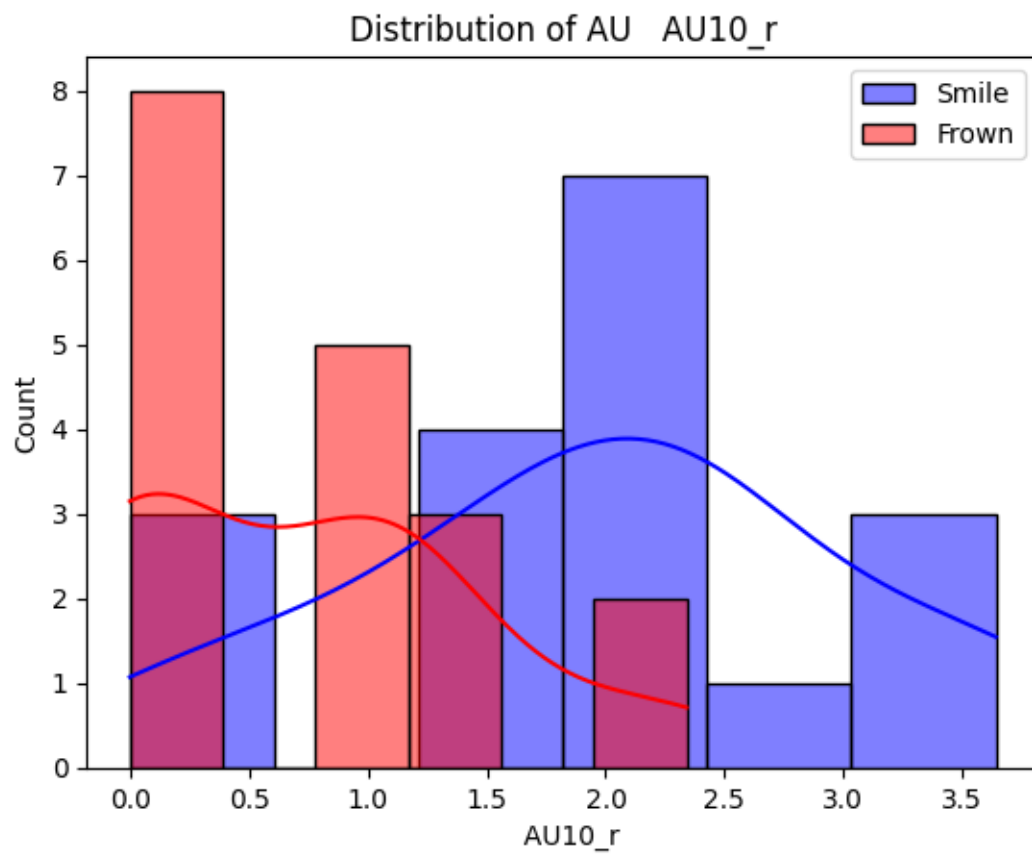


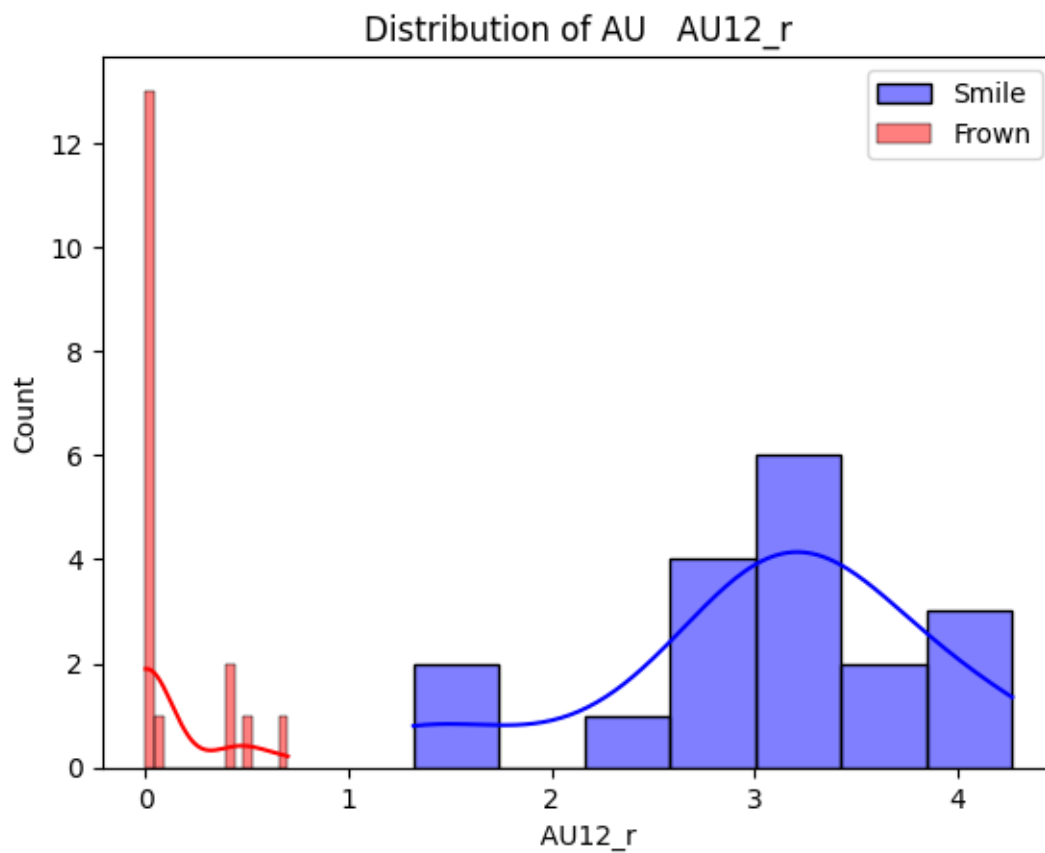


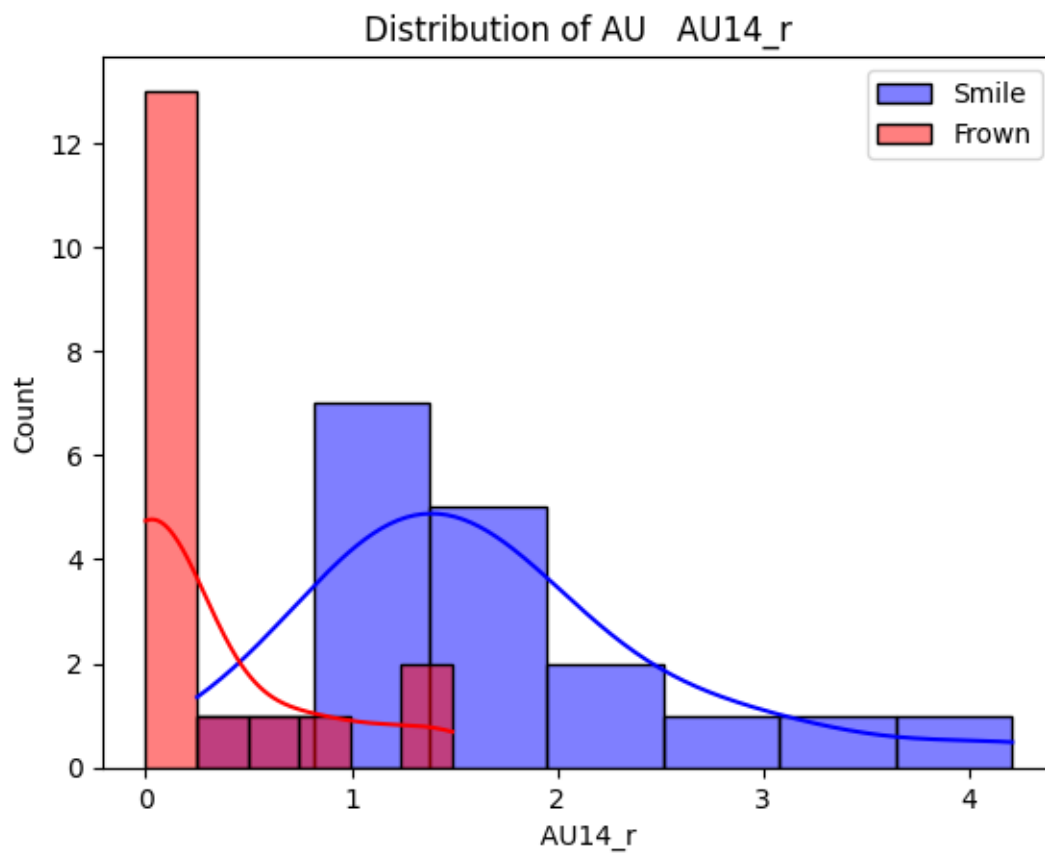


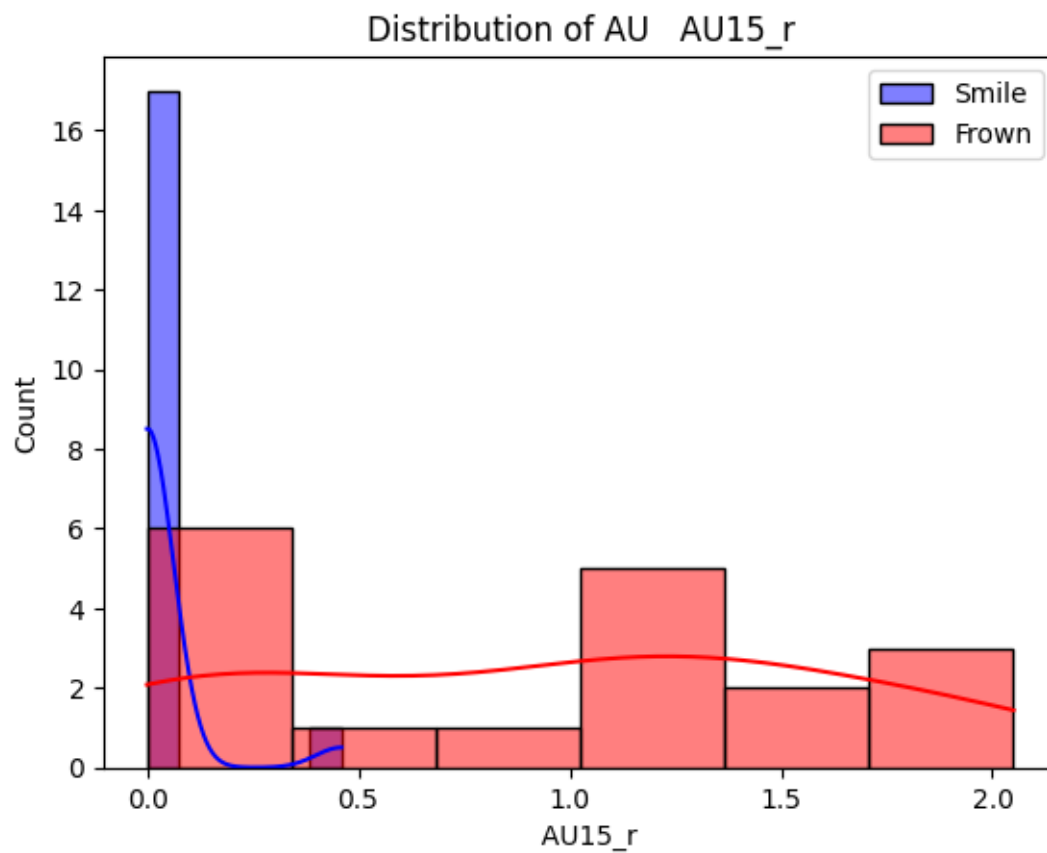


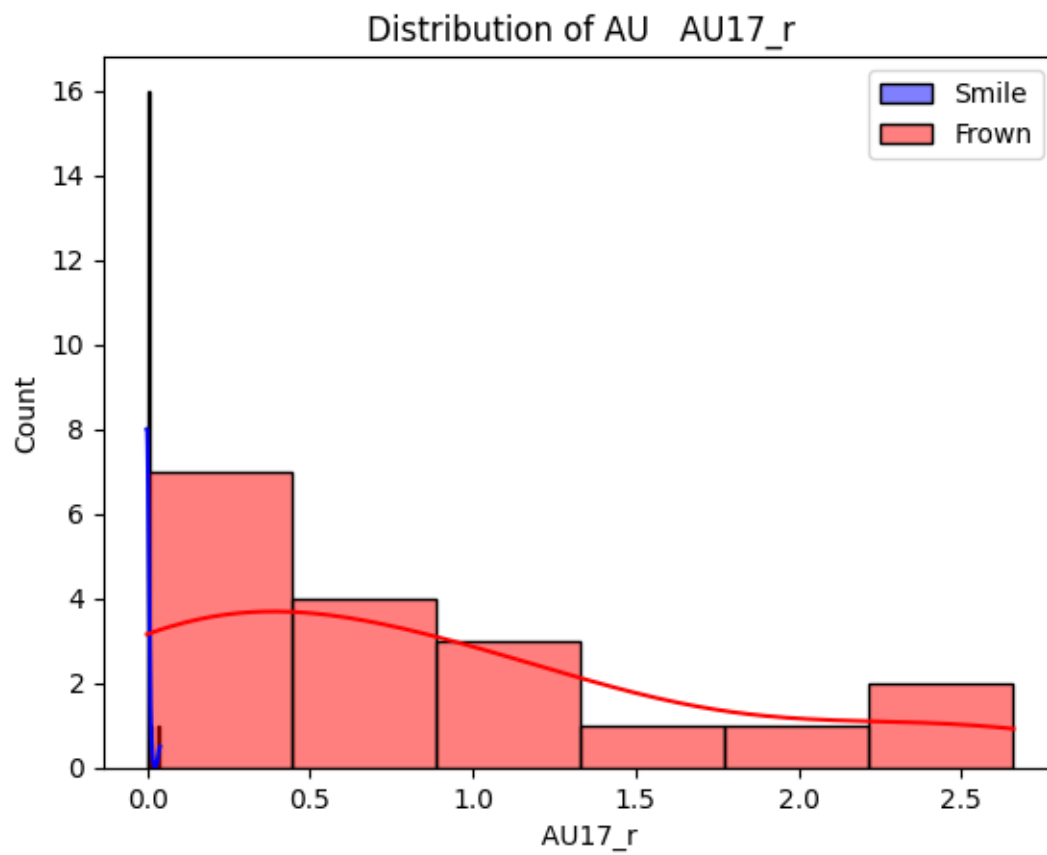


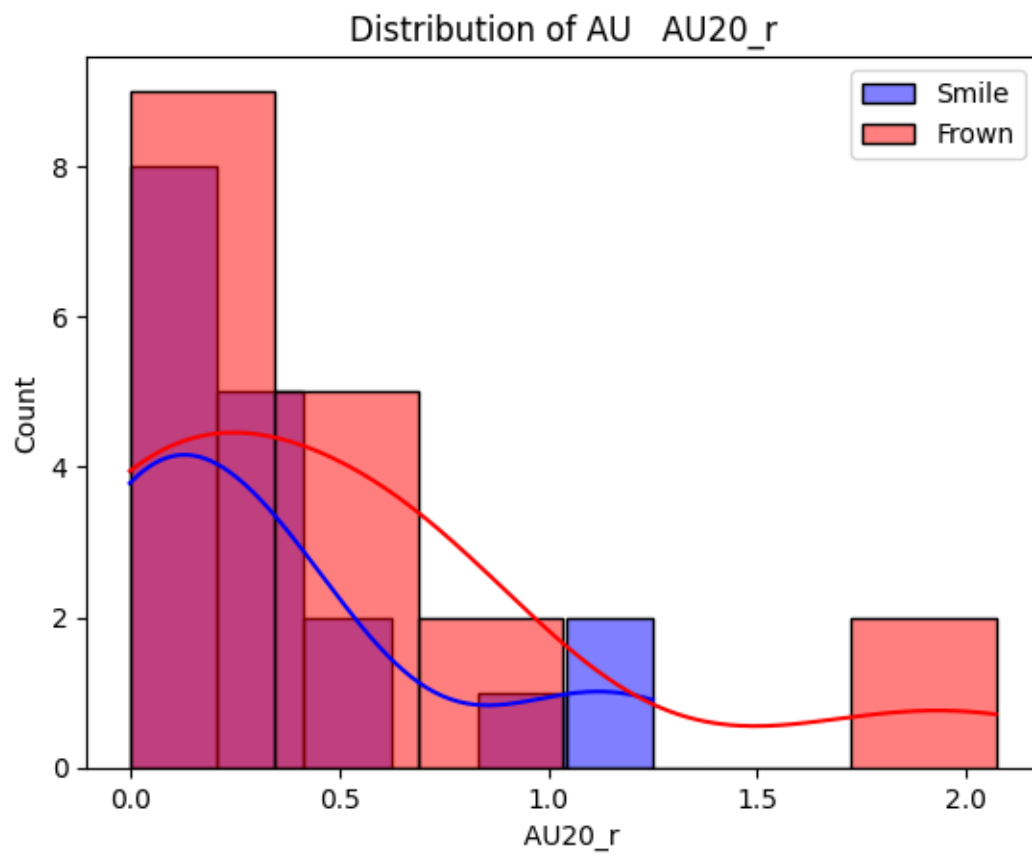


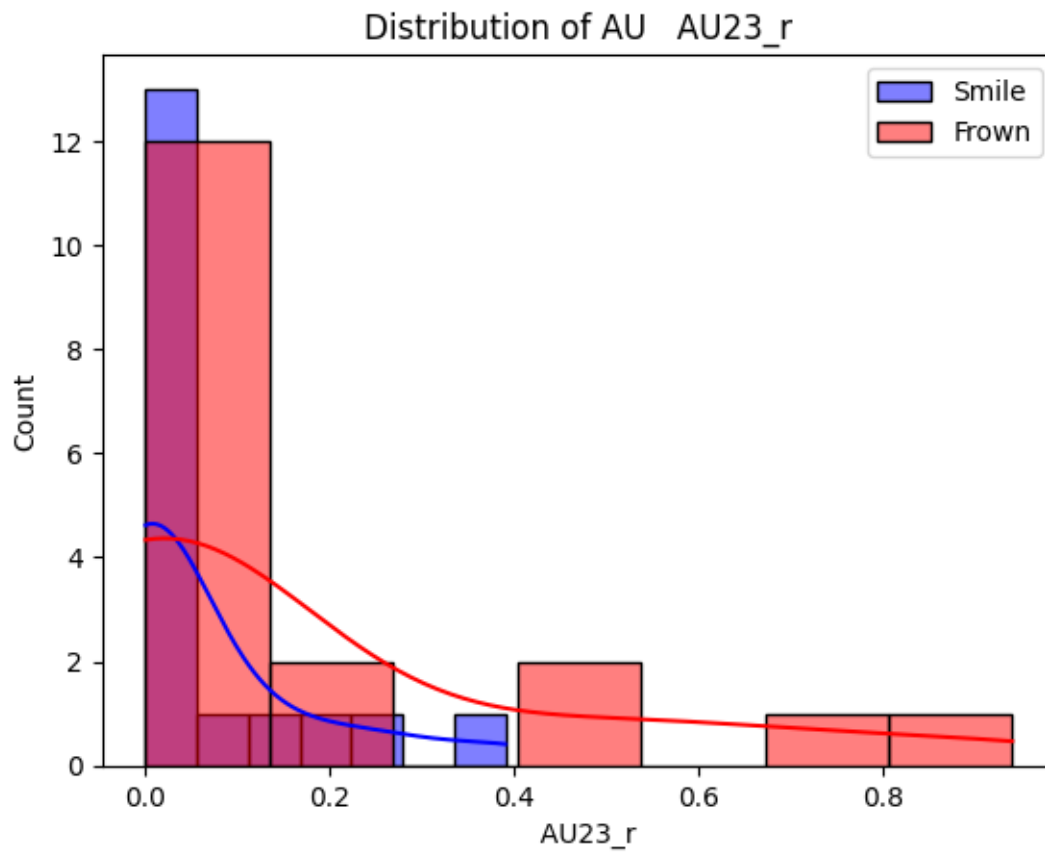


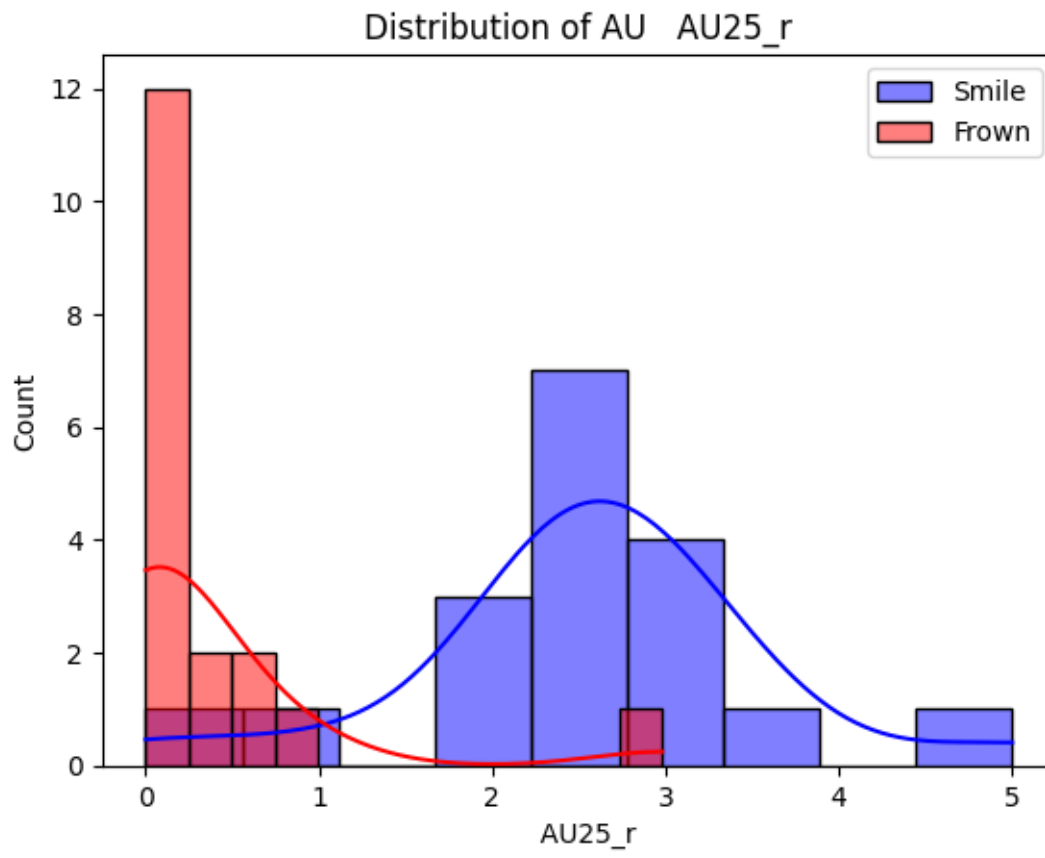


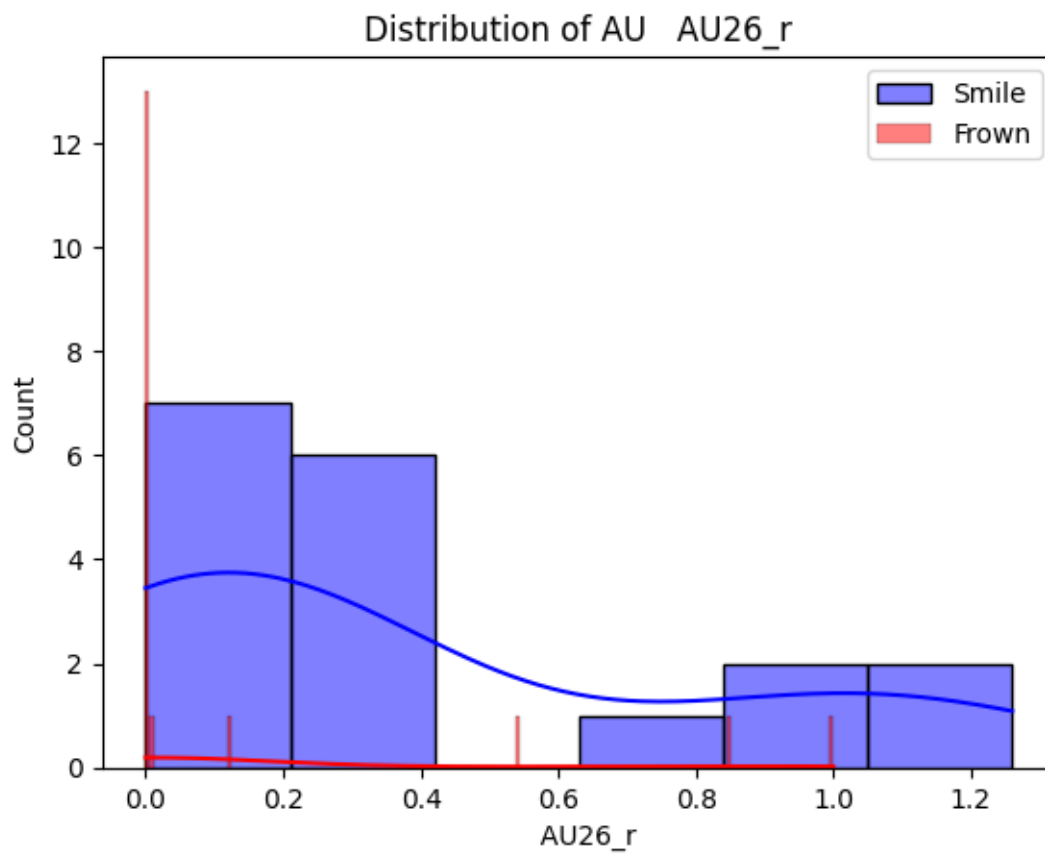


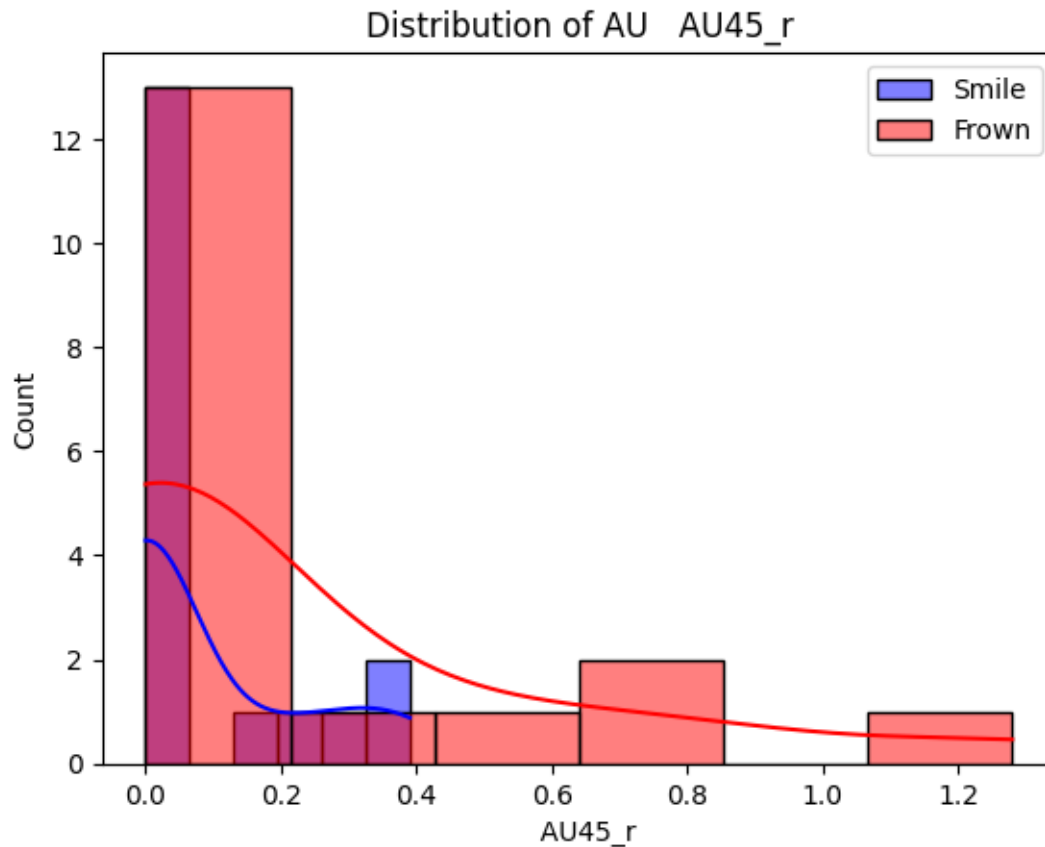








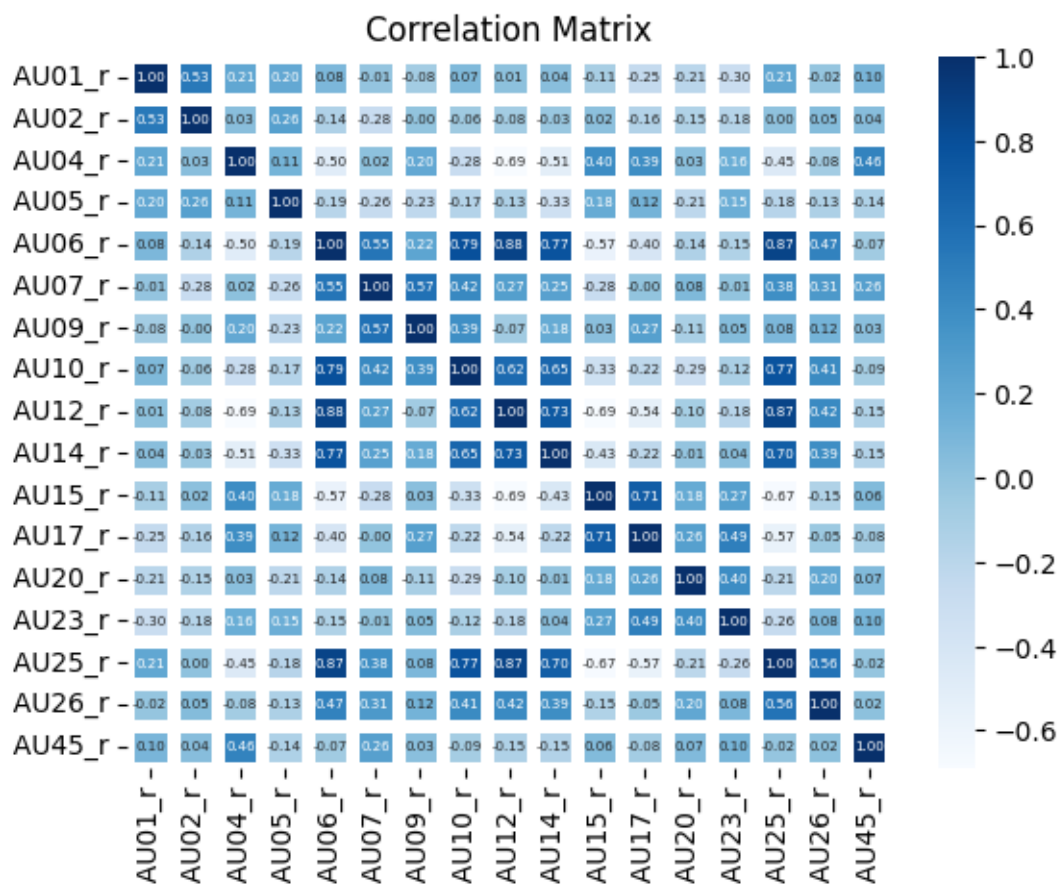


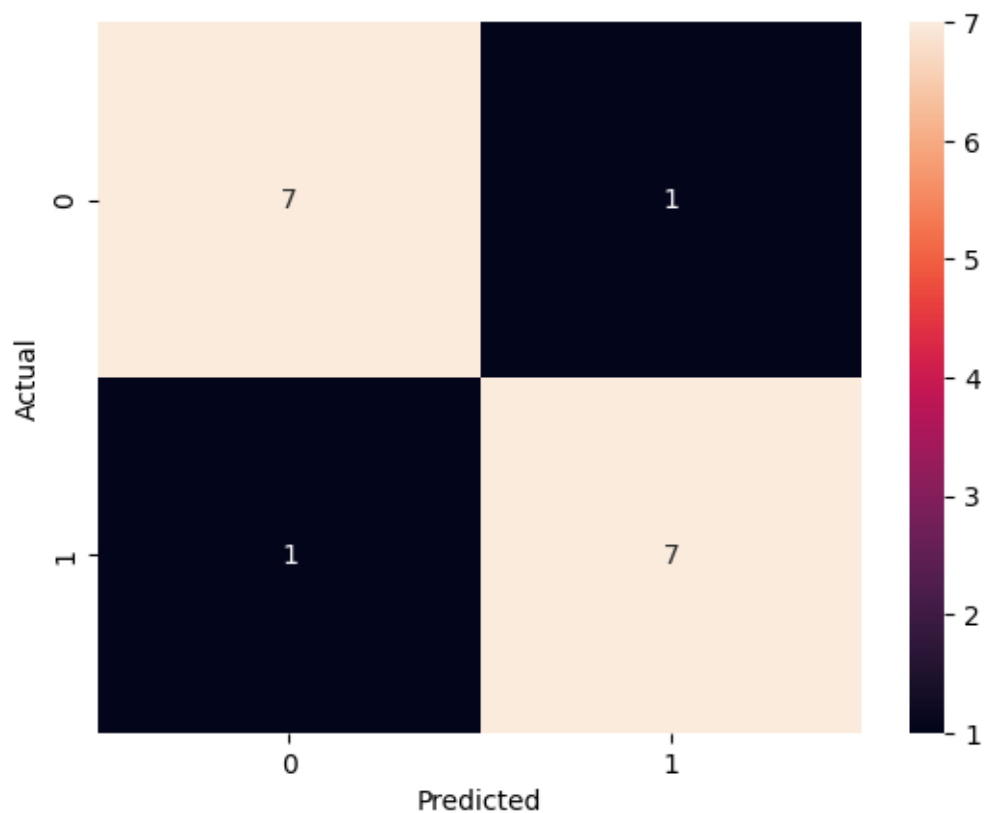


```
[ ]: from sklearn.metrics import confusion_matrix

# Correlation Matrix
corr_matrix = X_train.corr()
sns.heatmap(corr_matrix, annot=True, cmap='Blues', linewidths=5.0,
            annot_kws={"size": 5}, fmt=".2f")
plt.title("Correlation Matrix")
plt.show()

# Confusion Matrix
cm = confusion_matrix(y_test, predictions)
sns.heatmap(cm, annot=True, fmt='d')
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```





```
[ ]: from sklearn.metrics import classification_report
print(classification_report(y_test, predictions))
```

	precision	recall	f1-score	support
frown	0.88	0.88	0.88	8
smile	0.88	0.88	0.88	8
accuracy			0.88	16
macro avg	0.88	0.88	0.88	16
weighted avg	0.88	0.88	0.88	16