SCHULICH School of Engineering

UNIVERSITY OF CALGARY

Schulich School of Engineering / Electrical and Software

Speaker Recognition Using GMM

ENCM 509 - Fundamental Biometrics System Design
Winter 2024

Mushtaba Al Yasseen - 30094000
Abhay Khosla - 30085789
Parbir Lehal - 30096001

Group 20

Table of Contents

Introduction & Objectives

In the world of biometrics, a foundational concept of the Gaussian Mixture Model stands out which helps in deciphering human speech. Not only it's a statistical tool but in our project, it helps in analyzing the complex nature of the spoken language. For example, imagine yourself in a crowded room, your conversation through various voices. While all of this is happening our brain instinctively separates and recognizes individual speech patterns. Gaussian Mixture Model seeks to emulate through a sophisticated mathematical approach, breaking down speech into its elemental components to ensure the reliability of our analysis. The presence of speech recognition in our lives is undeniable. Whether it's virtual assistants like Siri or Alexa for the weather forecast, or providing directions, this technology transforms our spoken commands into actionable responses. At the core of this concept lies an intricate process: capturing audio, isolating its distinguishing features, similar to a fingerprint and interpreting each unique minutiae. Our project delves into a compelling investigation of AI-generated speech versus authentic speech patterns, leveraging the voices of two political figures, Justin Trudeau and Donald Trump. By employing a specialized probe dataset, we generated AI-driven speeches to rigorously evaluate our model's proficiency in differentiating the dataset. This project is an exploration of the boundary between artificial intelligence and the quintessential human attribute of speech. Our journey through the landscapes of these renowned personas will reveal not only the capabilities of current technology but also the evolving field of speech recognition.

Methodology

Description of Chosen Approach

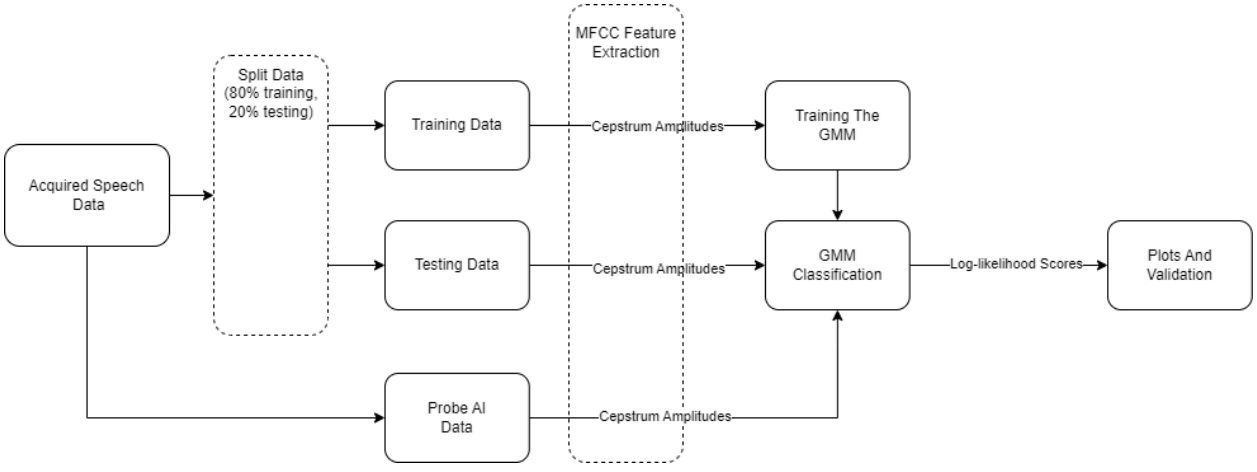

Figure 1 - Speaker Recognition Using GMM Project Block Diagram

As shown in *Figure 1*, we first start by splitting our data [1-16] into chunks based on the 80/20 rule, where 80% of the data is utilized for training, while the remaining 20% of data is reserved for testing. We have also highlighted the use of the probe AI data, which we will use for validation later on. Once we split our data into chunks, we then utilize a specific type of feature extraction known as Mel-frequency cepstral coefficients (MFCC) which is especially useful for the recognition of human speech as we will also get to later on. Furthermore, once we have extracted the features using MFCC, we will then utilize them to train the Gaussian Mixture Model and subsequently perform GMM classification in order to extract the log-likelihood scores. Finally, once the log-likelihood scores are extracted, we then have the opportunity to plot our probability density functions, confusion matrices, and the ROC/DET curves.

Key Phases in GMM-Based Speaker Recognition Process

1. **Data Collection:** This phase involves gathering raw speech data, which is the foundation for all analysis and model training in our AI speaker recognition project.
2. **Feature Extraction Using MFCC:** Mel-Frequency Cepstral Coefficients (MFCC) are computed to capture the unique aspects of speech, serving as the pivotal features for recognizing individual speakers.
3. **Performing Classification:** Once features are extracted, the Gaussian Mixture Model (GMM) classifier is trained to distinguish between speakers, leading to a model that can accurately classify speech samples.
4. **Plots and Validation:** The final step consists of visualizing the results through various plots such as Probability Density Function (PDF), Confusion Matrix, Receiver Operating Characteristic (ROC) and Detection Error Tradeoff (DET) curves. This will help us validate the model's performance and reliability.

Algorithm(s) & Implementation

MFCC & Gaussian Mixture Model

As previously explained, Mel-frequency cepstral coefficients (MFCC) is a specific type of speech feature extraction that is especially useful for the recognition of human speech [17]. Essentially, MFCCs make up an MFC in which the frequency bands are spaced equally on the mel scale [18]. Compared to frequency bands which are linearly spaced, MFCCs can estimate/mimic the human auditory system more strictly as humans don't recognize alterations in pitch linearly [18]. To calculate MFCCs, the discrete Fourier transform (DFT) of the audio signal is taken first [17]. Once a DFT is taken, a logarithm and Fourier inverse is then done, which is essentially taking a "spectrum of a spectrum", producing a "cepstrum" [17]. For our speech recognition project more specifically, we are using the *python_speech_features* library which includes a function for the computation of MFCCs (amplitudes of the cepstrum), finally storing them in a Numpy array [18, 19].

Log Likelihood Score & Probability Density Function

The log-likelihood score is a measure used in statistics to determine the goodness of fit of a statistical model to a set of data. It measures how well a model predicts the observed data. The higher the log-likelihood score, the better the model fits the data. The log-likelihood score is calculated as the natural logarithm of the likelihood function. The likelihood function is the probability of observing the data given a set of parameter values in the model. Taking the logarithm of the likelihood function is often

more convenient for mathematical and computational reasons. The probability density function (PDF) is a function that describes the likelihood of a random variable taking on a particular value. For continuous random variables, the PDF gives the probability that the variable falls within a particular range of values. The area under the PDF curve over a given range corresponds to the probability of the random variable falling within that range.

Integration of Confusion Matrix

The binary confusion matrix becomes a cornerstone in our speaker recognition project as it allows us to quantitatively assess the performance of our Gaussian Mixture Model-based classifier. The confusion matrix serves as a metric to evaluate the accuracy of the system. It is a sort of truth table that we learnt from our ENEL 353 Digital Circuits course, it categorizes each test sample into one of four categories: true positives, false positives, true negatives and false negatives. This methodological approach is critical when our model evaluates the artificial intelligence-generated speeches against the authentic vocal patterns. The matrix is a statistical mirror reflecting if the system gets tricked by the AI-generated speech, as well as its proficiency in validating genuine speech, thus ensuring that the model is reliable. Integrating the matrix into our project helps us understand the future work which can be done in this project, for example, it will allow us to fine-tune our model further to calibrate a threshold which decides the likelihood of a speech sample.

Results

Gaussian Mixture Model Probability Density Function

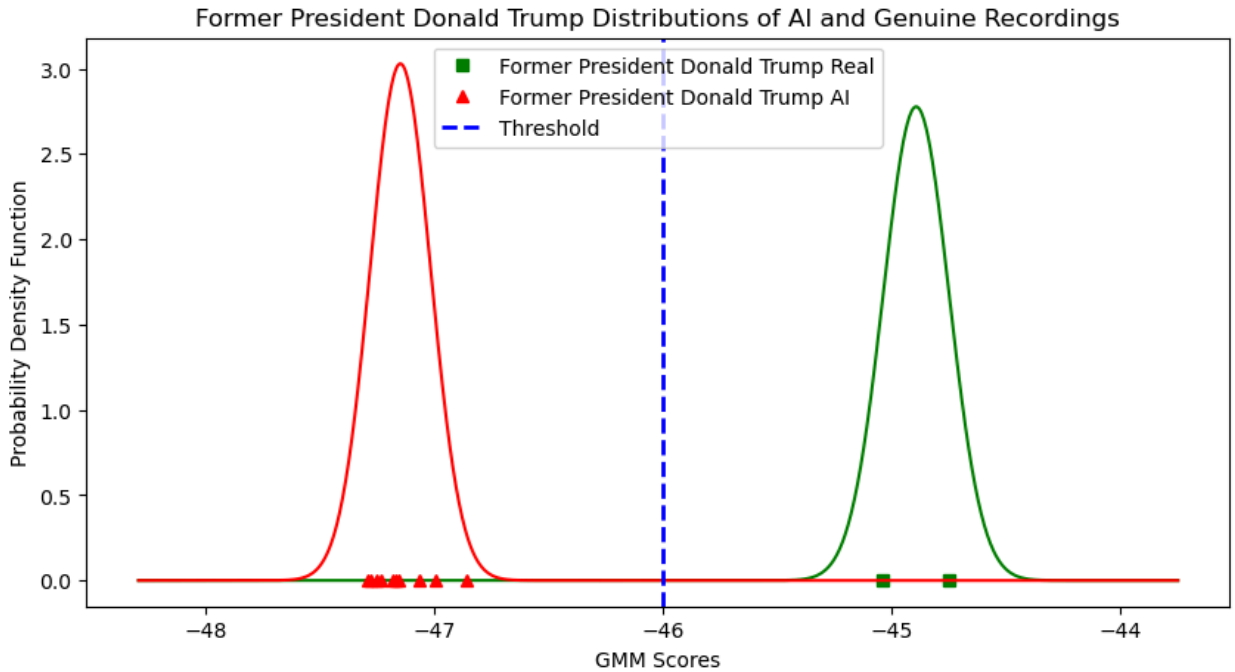

Figure 2 - Former President Donald Trump Normal Probability Density Function Plots

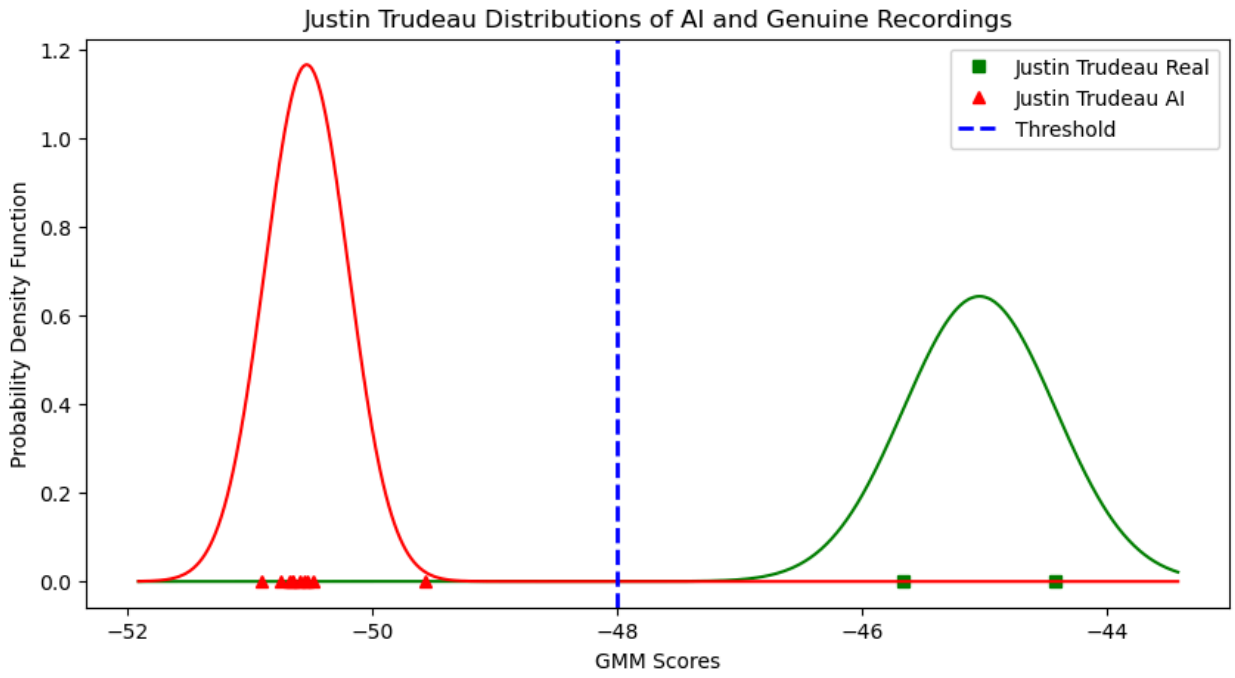

Figure 3 - Justin Trudeau Normal Probability Density Function Plots

After calculating our test scores and then visualizing the normalized probability density function plot, we can see a common trend between both our subjects. First and foremost, these graphs help us see where our threshold is, and in both graphs, we see a relatively similar threshold. Additionally, that helps us see the distinction between both AI and Genuine data, where the mean scores differ, indicating that there is a low close match between both speeches for both our subjects. One interesting takeaway from performing this project is the peak of our AI data curve seems to be larger than our genuine. This indicates to us that there is less score variation between the 10 sets of data, meaning that real speech will typically vary more from sample to sample than AI data.

Confusion Matrix Results

Prediction Outcomes	Prediction Outcome Values
True Positive Rate (TPR)	100.00 %
True Negative Rate (TNR)	100.00 %
False Positive Rate (FPR)	0.00 %
False Negative Rate (FNR)	0.00 %

Table 1 - Former President Donald Trump Classification Outcomes

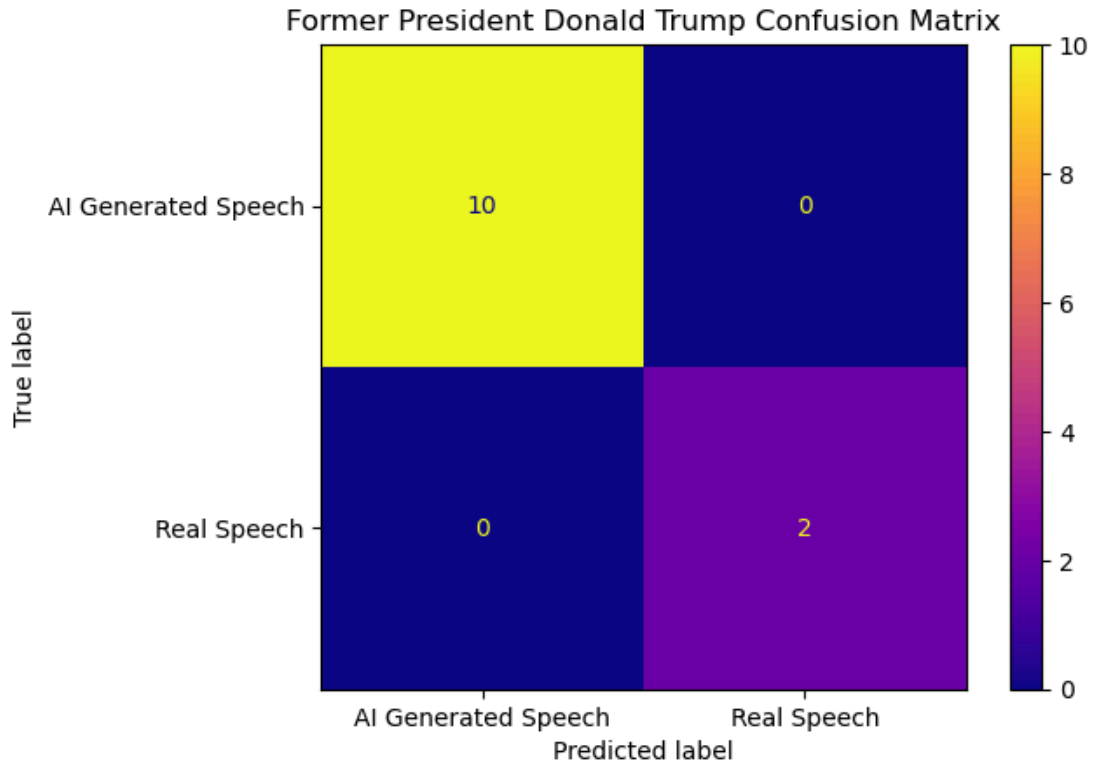

Figure 4 - Former President Donald Trump Confusion Matrix with 20 Gaussian Components

The confusion matrix for our analysis of AI-generated speech versus the real speech of Former President Donald Trump reveals a clear differentiation between the two classes. The speech generation tool [20] we utilized to generate the AI speech was notably distinct from authentic samples. As demonstrated from our results the true positive rate(TPR) and true negative rate(TNR) of 100%. Correspondingly 0% false positive rate(FPR) and false negative rate (FNR) underscore our model's ability in the performance in this classification task as it's evident that AI-generated speeches were vastly different from the authentic test samples. The flawless segregation achieved in our Donald Trump dataset highlights the pivotal success of our project.

Prediction Outcomes	Prediction Outcome Values
True Positive Rate (TPR)	100.00 %
True Negative Rate (TNR)	100.00 %
False Positive Rate (FPR)	0.00 %
False Negative Rate (FNR)	0.00 %

Table 2 - Justin Trudeau Classification Outcomes

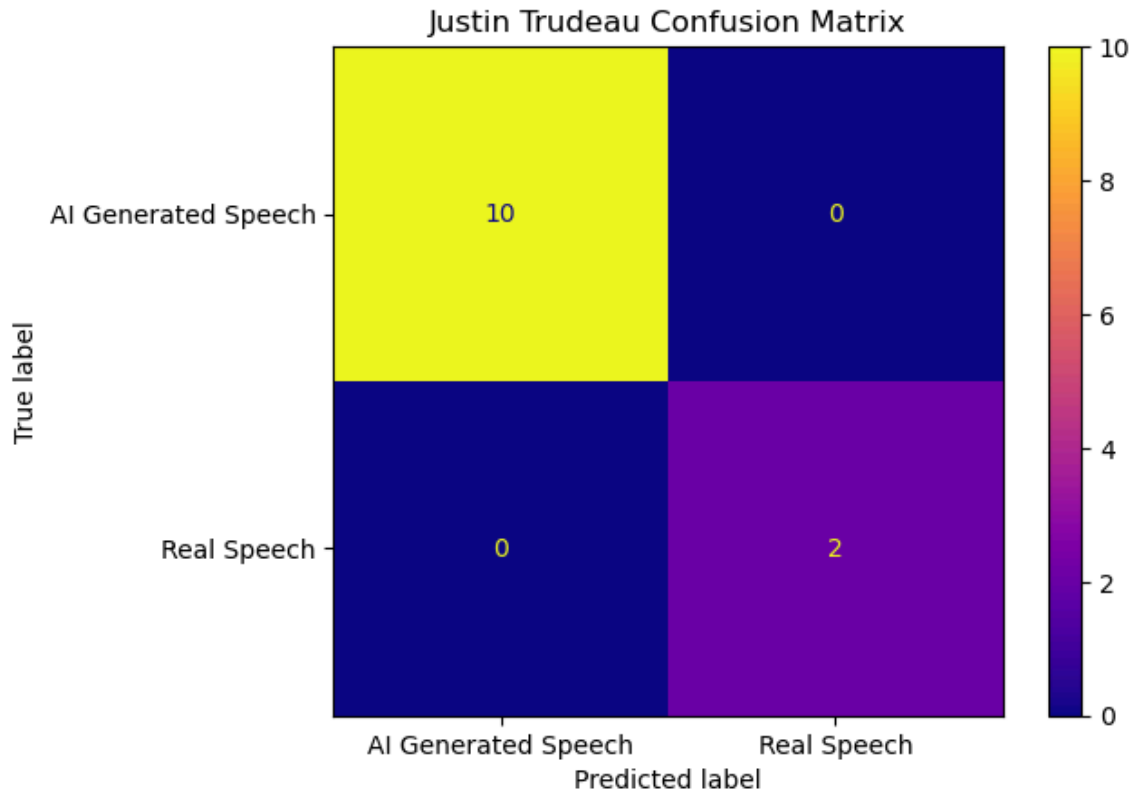

Figure 5 - Justin Trudeau Confusion Matrix with 20 Gaussian Components

The current Canadian Prime Justin Trudeau confusion matrix parallels the results obtained with Donald Trump's data, suggesting a consistent performance by our model across different speakers. Once again, we see the model achieving a TPR and TNR of 100%, with no instances of misclassification. The AI tool used to produce the AI-generated speeches for Trudeau failed to capture the essence of the real samples, evidenced by the model's classification. The consistency of the results between the confusion matrices of Donald Trump and Justin Trudeau indicates the reliability of our classification model while concurrently highlighting the speech tool's current limitations in replicating true human speech nuances. It encourages an ongoing effort to enhance the process to produce more lifelike AI speech, ultimately aiming to test the boundaries of our model's capabilities. The project's success so far is promising, marking a significant milestone in the field of biometric speech analysis and setting the stage for further advancements in AI speech recognition.

Receiver Operating Characteristic Curve

In the context of speaker recognition, the Receiver Operating Characteristic (ROC) curve serves as a graphical representation which encapsulates the performance of the Gaussian Mixture Model. As seen in our Week 1 Part 3 Lecture notes [21] the curve plots the True Acceptance Rate (TAR) or True Positive Rate (TPR) against the False Acceptance Rate (FAR), providing a visual contrast between genuine and imposter attempts at voice identification. The ROC curve is helpful in the evaluation and comparison of biometric systems, as it aids in the identification of an optimal decision threshold that balances the rate

of true acceptances against false acceptances. The ROC curve analysis can be found in the following study: "ROC curve for different biometric techniques" [22] (2014), available on ResearchGate, where the authors demonstrate the use of ROC curves to evaluate the performance of several biometric systems, ranging from GMM to hybrid methods.

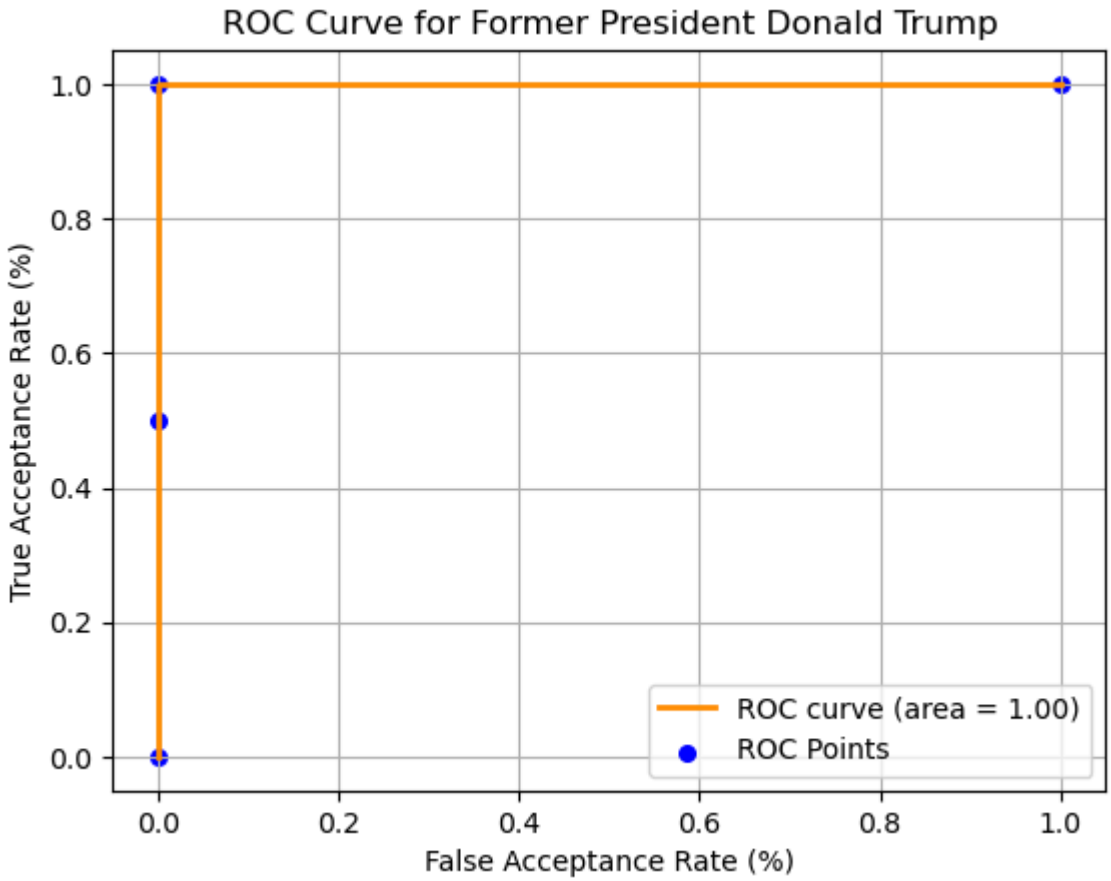

Figure 6 - ROC Curve for Former President Donald Trump

The ROC curve above highlights the model's performance with the area under the curve of 1.00 indicating an outstanding accuracy in differentiating between the authentic and AI-generated speeches of Former President Donald Trump. This is backed up by the rates showcased below in the table with their corresponding thresholds. The model demonstrates complete accuracy signifying an exceptional binary classification with no middle ground.

Thresholds	False Acceptance Rate (FAR)	True Acceptance Rate (TAR)
-43.78	0.00 %	0.00 %

-44.79	0.00 %	50.00 %
-45.01	0.00 %	100.00 %
-47.26	100.00 %	100.00 %

Table 3 - Former President Donald Trump

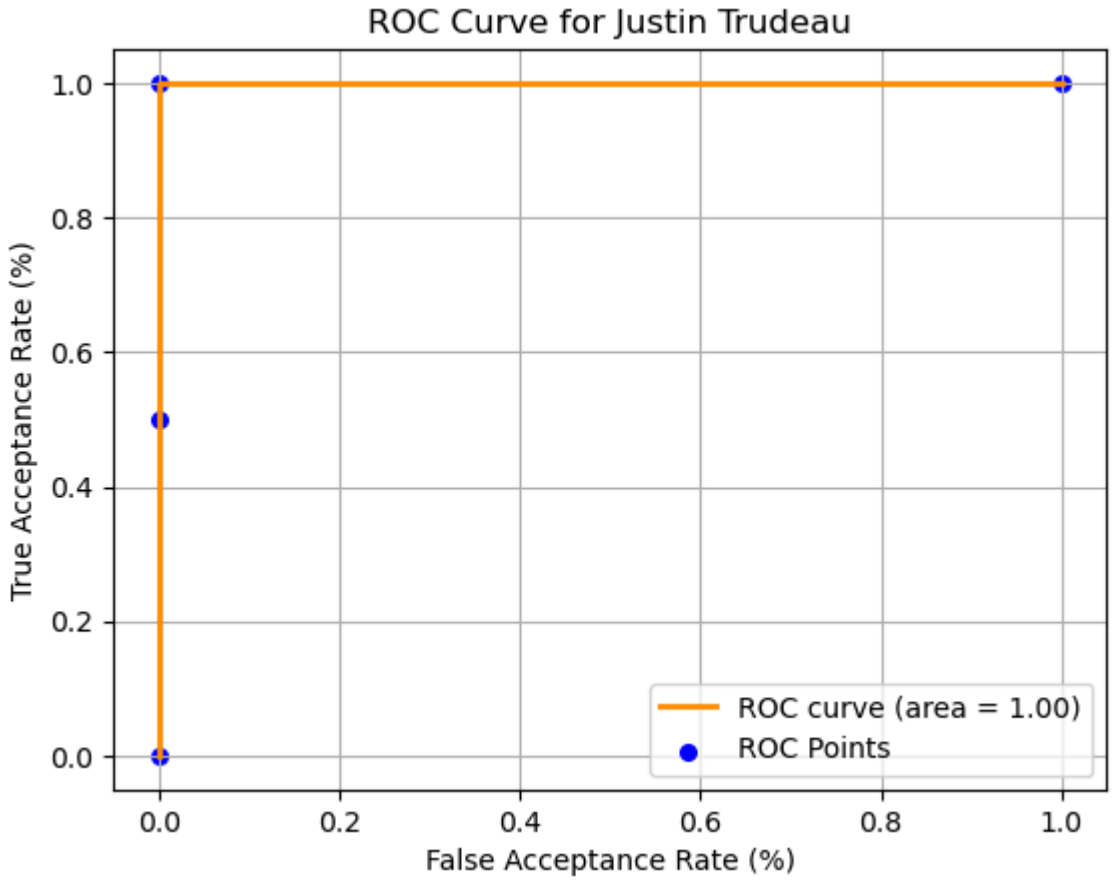

Figure 7 - ROC Curve for Justin Trudeau

The ROC curve for Justin Trudeau also attains an area under the curve of 1.00 mirroring the proficiency as seen with the model handling Donald Trump's speeches. This remarkable accuracy, although indicating a potentially accurate model, must be cautiously interpreted considering the limited test samples used, which may not fully represent the variability of natural speech. The model, composed of 20 Gaussian components, seems to have perfectly modelled the features within the scope of the available data.

Thresholds	False Acceptance Rate (FAR)	True Acceptance Rate (TAR)
-43.49	0.00 %	0.00 %

-44.50	0.00 %	50.00 %
-45.63	0.00 %	100.00 %
-50.92	100.00 %	100.00 %

Table 4 - Justin Trudeau

Detection Error Trade-Off Curve

In speaker recognition systems, the Detection Error Tradeoff (DET) curve is a helpful analytical tool often used alongside the Receiver Operating Characteristic Curve (ROC) curve. A DET curve, as illustrated in our lecture slides in Week 1 [21] plots the False Acceptance Rate (FAR) against the False Rejection Rate (FRR) on scales. This graph helps us to determine the tradeoffs between type I and type II errors more distinctly. The DET curve is pivotal in pinpointing the decision threshold where the balance between false rejections and false acceptances is optimal.

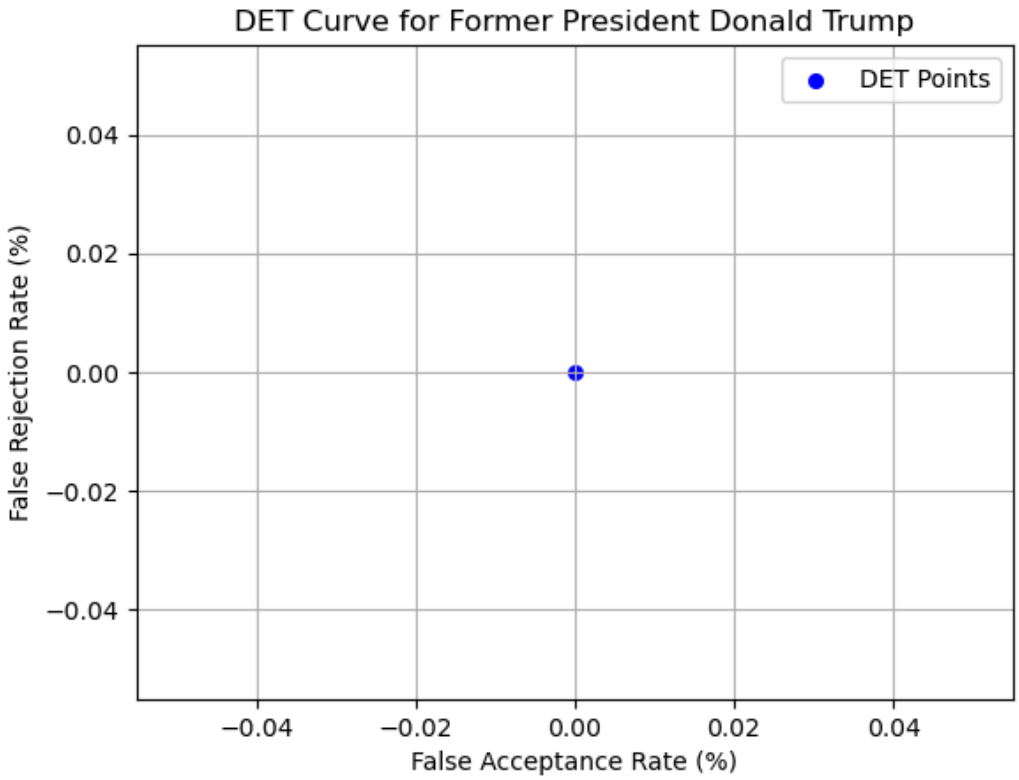

Figure 8 - DET Curve for Former President Donald Trump

The DET curve for Former President Donald Trump presents a point at the origin suggesting our model endures excellent performance with zero False Acceptance Rate (FAR) and False Rejection Rate (FRR). This indicates a perfect separation between the genuine and imposter voice samples. In biometric systems usually, this is hard to achieve it also underscores the model's precision as the small sample size, this flawless classification is most likely due to our AI dataset having an absence of variability and noise

which typically challenge such models and the singular threshold value reflect a model that did not encounter any edge cases that would be challenging.

Thresholds	False Acceptance Rate (FAR)	True Acceptance Rate (TAR)
-45.07	0.00 %	0.00 %

Table 5 - Former President Donald Trump

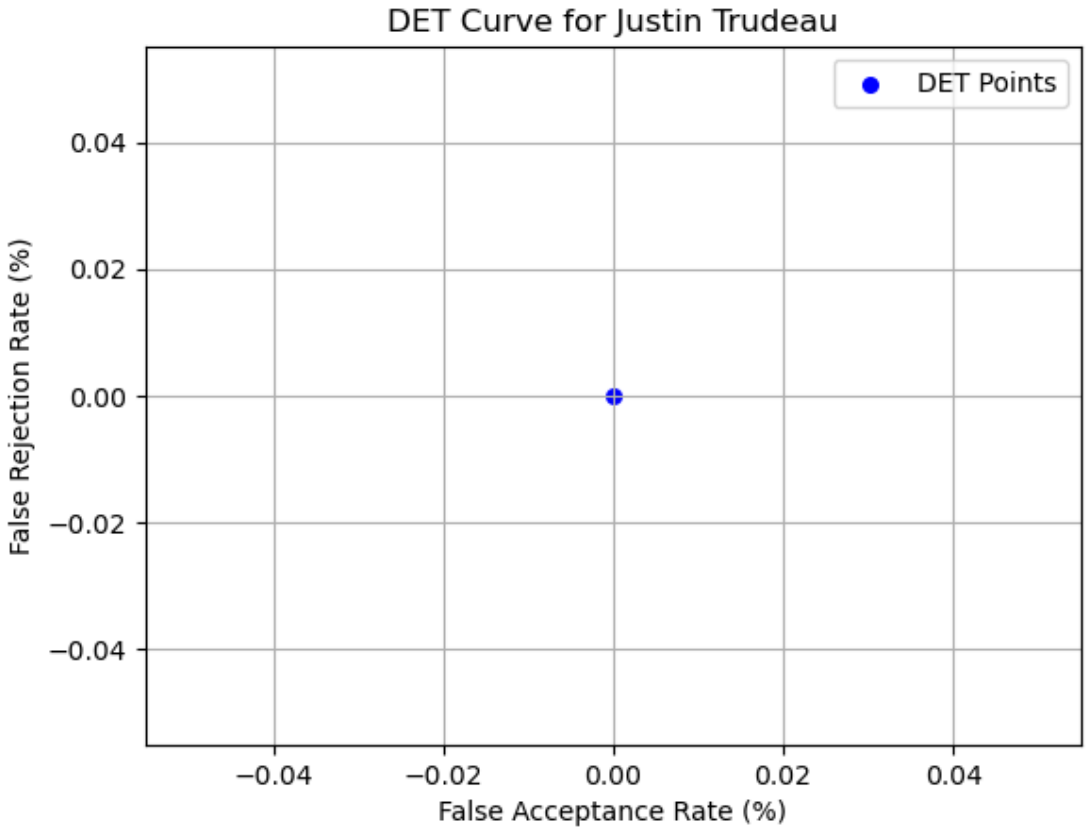

Figure 9 - DET Curve for Justin Trudeau

Similarly, the DET curve for Justin Trudeau highlights one single point at the origin, that we can conclude that the model is a perfect classifier and its performance distinguishes between true and false speech samples. This result is indicative of a model that operates with remarkable accuracy within the limits of the current dataset. However, the reality of a single threshold yielding perfect classification does raise questions regarding our model's generalizability to larger and potentially more diverse sets of voice data.

Thresholds	False Acceptance Rate (FAR)	True Acceptance Rate (TAR)
-45.66	0.00 %	0.00 %

Table 6 - Former President Donald Trump

Conclusion

In conclusion, our project has delved into the fascinating realm of AI-generated speech versus authentic speech patterns, using the voices of Justin Trudeau and Donald Trump as our subjects. Through the application of the Gaussian Mixture Model (GMM) and other statistical tools like the log-likelihood score, probability density function, and confusion matrix, we have rigorously evaluated our model's proficiency in differentiating between AI-generated and genuine speech.

Our results have shown a clear distinction between the two classes, with a 100% true positive rate (TPR) and true negative rate (TNR) for both subjects, indicating that our model successfully differentiated between the AI-generated and authentic speeches. The ROC curves further reinforce the model's accuracy, with an area under the curve of 1.00 for both Trump and Trudeau, signifying outstanding performance in binary classification.

However, while our model has shown promising results, it's essential to note that the AI-generated speech was not as close to the real speech as we had hoped. This highlights the ongoing challenge in AI speech generation and the need for further advancements to achieve more lifelike results. Nonetheless, our project marks a significant milestone in the field of biometric speech analysis, showcasing the capabilities of current technology and paving the way for future advancements in AI speech recognition.

References

[1] "Trump White House Archived," YouTube, [Online]. Available: https://www.youtube.com/watch?v=hDNiNdsPHNA&ab_channel=TrumpWhiteHouseArchived.

[2] "Trump White House Archived," YouTube, [Online]. Available: https://www.youtube.com/watch?v=4aQjA-oZW_o&ab_channel=TrumpWhiteHouseArchived.

[3] "Trump White House Archived," YouTube, [Online]. Available: https://www.youtube.com/watch?v=TYwbGPxQLrs.

[4] "Trump White House Archived," YouTube, [Online]. Available: https://www.youtube.com/watch?v=dKppODKE-LU.

[5] "Trump White House Archived," YouTube, [Online]. Available: https://www.youtube.com/watch?v=6h5_d3DUdR4.

[6] "Trump White House Archived," YouTube, [Online]. Available: https://www.youtube.com/watch?v=vjE1ZGtAGck.

[7] "Trump White House Archived," YouTube, [Online]. Available: https://www.youtube.com/watch?v=5LJhaat0iTg&ab_channel=TrumpWhiteHouseArchived.

[8] "Trump White House Archived," YouTube, [Online]. Available: https://www.youtube.com/watch?v=vb9qXvGAQTA&ab_channel=TrumpWhiteHouseArchived.

[9] "Trump White House Archived," YouTube, [Online]. Available: https://www.youtube.com/watch?v=QCTicBFE5t0&ab_channel=TrumpWhiteHouseArchived.

[10] "Justin Trudeau - Prime Minister of Canada," YouTube, [Online]. Available: https://www.youtube.com/watch?v=5HEMKhi0JaM&ab_channel=JustinTrudeau%E2%80%93PrimeMinisterofCanada.

[11] "Justin Trudeau - Prime Minister of Canada," YouTube, [Online]. Available: https://www.youtube.com/watch?v=InR69mIZMzA&ab_channel=JustinTrudeau%E2%80%93PrimeMinisterofCanada.

[12] "Justin Trudeau - Prime Minister of Canada," YouTube, [Online]. Available: https://www.youtube.com/watch?v=AYDuYpAFvro&ab_channel=JustinTrudeau%E2%80%93PrimeMinisterofCanada.

[13] "Justin Trudeau - Prime Minister of Canada," YouTube, [Online]. Available: https://www.youtube.com/watch?v=MxLx12kWl7c&ab_channel=JustinTrudeau%E2%80%93PrimeMinisterofCanada.

[14] "Justin Trudeau - Prime Minister of Canada," YouTube, [Online]. Available: https://www.youtube.com/watch?v=HyNnXnmTaKM&ab_channel=JustinTrudeau%E2%80%93PrimeMinisterofCanada.

[15] "Justin Trudeau - Prime Minister of Canada," YouTube, [Online]. Available: https://www.youtube.com/watch?v=KDPe9MHy-5c&ab_channel=JustinTrudeau%E2%80%93PrimeMinisterofCanada.

[16] "Justin Trudeau - Prime Minister of Canada," YouTube, [Online]. Available: https://www.youtube.com/watch?v=JAi4o4SGqng&ab_channel=JustinTrudeau%E2%80%93PrimeMinisterofCanada.

[17] S. Z. Mirski and A. Basu, "Head and neck cancer: Role of robotics," Otolaryngol. Clin. North Am., vol. 56, no. 3, pp. 515-525, Jun. 2023. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/B9780323917766000166.

[18] "Mel-frequency cepstrum," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Mel-frequency_cepstrum#:~:text=In%20sound%20processing%2C%20the%20mel,collectively%20make%20up%20an%20MFC.

[19] "Python Speech Features," Read the Docs, [Online]. Available: https://python-speech-features.readthedocs.io/en/latest/.

[20] “Voice Cloning – Clone Your Own Voice." [Online]. Available: https://play.ht/studio/voice-cloning.

[21] ESE Department, Schulich School of Engineering, “LecWeek1-Part3-ENCM509”, 2024. Available: On D2L Slides

[22] "ROC curve for different biometric techniques," ResearchGate, [Online]. Available: https://www.researchgate.net/figure/ROC-curve-for-different-biometric-techniques_fig4_258652120#:~:text=A%20ROC%20curve%20is%20a,or%20fingerprint%2Donly%20based%20methods.

Appendix

Appendix 1: Installation and Execution Guide

Environment Setup

1. **Open Anaconda Prompt:** Navigate to the start menu and find the Anaconda3 folder. Click on "Anaconda Powershell Prompt" to open it.
2. **Create Conda Environment:** To create a new environment specifically for the Speaker Recognition project, run the following command:

```
conda create -n encm509SpeakerRecongition python=3.12.2 numpy matplotlib scipy
```

3. **Activate Environment:** Before installing additional packages, activate the environment:
   `conda activate encm509SpeakerRecongition`
4. **Install Required Libraries:** With the environment activated, install the following libraries using the conda package manager:
   `conda install -c conda-forge python_speech_features sklearn`
   The command above installs the python_speech_features library for extracting Mel Frequency Cepstral Coefficients (MFCC), and sklearn for the Gaussian Mixture Model (GMM) and confusion matrix display functionality.
5. **Install Jupyter Notebook:** If you don't have Jupyter Notebook installed in this environment, install it using:
   `conda install -c conda-forge jupyterlab`
6. **Launch Jupyter Notebook:** Run the following command to start Jupyter Notebook:
   `jupyter notebook`

Running the Notebook

1. **Access the Project Folder:** Using the Jupyter Notebook interface in your web browser, navigate to the folder containing the `SpeakerRecognitionFinalProject.ipynb` notebook file. Your folder structure should look similar to the provided screenshot, with separate folders for AI and real speech data for each individual (TrudeauAI, TrudeauReal, TrumpAI, TrumpReal) all of the data is provided as a ZIP hence make sure it is unzipped into a folder before running the notebook.

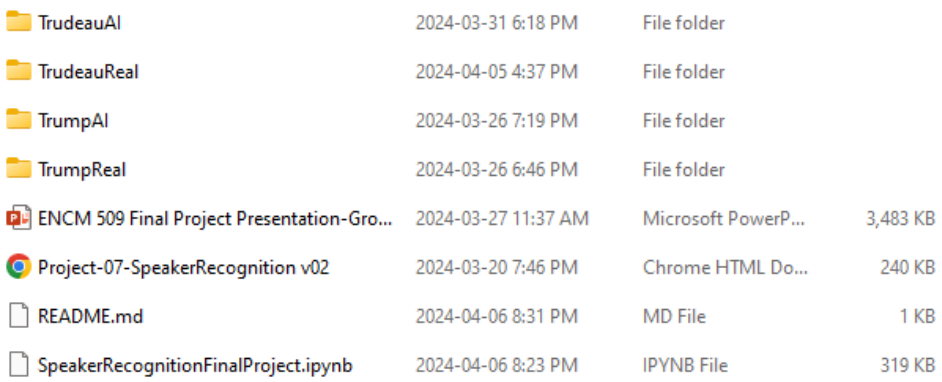

Figure A1.1 - Screenshot of Project Folder Structure on Windows

2. **Open the Notebook:** Click on the `SpeakerRecognitionFinalProject.ipynb` file to open the notebook in a new tab.
3. **Run the Notebook:** Execute each cell in the notebook by clicking on the cell by using the "Run" button in the Jupyter interface. Ensure all cells are run in sequence from top to bottom.
4. **Review the Results:** After running the notebook, the outputs will be displayed below each cell, including the plots for the probability distribution functions and the confusion matrices.

Appendix 2: Code

`SpeakerRecongnitionFinalProject.ipynb`

Note: The file will also be attached alongside the report

```
# Packages required to run the application
import numpy as np
from python_speech_features import mfcc
from sklearn.mixture import GaussianMixture
from scipy.io import wavfile
import matplotlib.pyplot as plt
from scipy.stats import norm
from sklearn.metrics import ConfusionMatrixDisplay, roc_curve, auc
```

Loading Donald Trump Data

```
NUMOFTRAININGSAMPLES = 8  # Number of samples to test is a constant

# Directory containing the testing/training data files for Donald Trump
TRUMPREALDATA = "./TrumpReal/"
donaldTrumpTrainingPaths = [f"{TRUMPREALDATA}{i}.wav" for i in range(1,
NUMOFTRAININGSAMPLES + 1)] # Training data
```

```
donaldTrumpTestingPaths = [f"{TRUMPREALDATA}{i}.wav" for i in
range(NUMOFTRAININGSAMPLES + 1, 11)] # Testing data
donaldTrumpAIPaths = [f"./TrumpAI/{i}.wav" for i in range(1, 11)] # Probe dataset
```

Loading Justin Trudeau Data

```
# Directory containing the testing/training data files for Justin Trudeau
TRUDEAUREALDATA = "./TrudeauReal/"
justinTrudeauTrainingPaths = [f"{TRUDEAUREALDATA}{i}.wav" for i in range(1,
NUMOFTRAININGSAMPLES + 1)] # Training data
justinTrudeauTestingPaths = [f"{TRUDEAUREALDATA}{i}.wav" for i in
range(NUMOFTRAININGSAMPLES + 1, 11)] # Testing data
justinTrudeauAIPaths = [f"./TrumpAI/{i}.wav" for i in range(1, 11)] # Probe
dataset
```

Helper Function 1 - Extracting Features From Audio Files (MFCC)

```
def extractMFCCFeatures(filePaths):
    """Extracts MFCC features from each .wav file specified in the file paths.

    Parameters
    ----------
    filePaths : list of str
        Paths to the .wav files.

    Returns
    -------
    numpy.ndarray
        A numpy array of MFCC features from all files.
    """
    features = []
    for filePath in filePaths:
        # -------------- Start of Referenced Code from ENCM 509
Project-07-SpeakerRecongition v02 on D2L---------------
        sample_rate, audio_data = wavfile.read(filePath) # Read the .wav file
        # -------------- End  of Referenced Code from ENCM 509
Project-07-SpeakerRecongition v02 on D2L---------------
        mfcc_data = mfcc(audio_data, samplerate=sample_rate) # Compute the MFCC
feautres
        features.append(mfcc_data)
```

```
    return np.concatenate(features, axis=0) # Concatenate the features from all
files into a single numpy array
```

Helper Function 2 - Calculating GMM Score for Each Audio File

```
def calculateGMMScores(filePaths, gmmModel):
    """Calculates GMM scores for a given set of file paths using the specified
GMM model.

    Parameters
    ----------
    filePaths : list of str
         Paths to the .wav files.
    gmmModel : GaussianMixture
        Trained GMM model.

    Returns
    -------
    list
        A list of GMM scores.
    """
    scores = []
    for filePath in filePaths:
        # -------------- Start of Referenced Code from ENCM 509
Project-07-SpeakerRecongition v02 on D2L---------------
        sample_rate, audio_data = wavfile.read(filePath) # Read the .wav file
        # -------------- End of Referenced Code from ENCM 509
Project-07-SpeakerRecongition v02 on D2L---------------
        mfcc_data = mfcc(audio_data, sample_rate) # Compute the MFCC feautres
        # -------------- Start of Referenced Code from ENCM 509 Lab3-SigVerif
GMM.ipynb on D2L---------------
        score = gmmModel.score(mfcc_data) # compute the per-sample average
log-likelihood of the given data.
        scores.append(score) # Concatenate the scores from all files into a
single list
         # -------------- End of Referenced Code from ENCM 509 Lab3-SigVerif
GMM.ipynb on D2L---------------
    return scores
```

Helper Function 3 - Plotting Probability Density Function of GMM Real Speech Data vs. AI Generated Speech

```
def plotProbabiltyDensityFunction(testScore, aiScore, thresholdValue, label1, label2, plotTitle):
    """Plots the probability distributions of real and AI-generated speech scores.

    Parameters
    ----------
    testScore : list
        GMM scores for the real speech samples.
    aiScore : list
         GMM scores for the AI-generated speech samples.
    thresholdValue : int
       The threshold value to determine the decision boundary.
    label1 : str
        The first label for the plot.
    label2 : str
        The second label for the plot.
    plotTitle : str
        The title for the plot.
    """
    # Determine the range for x values based on the lowest and highest scores from both test and AI scores
    minScore = min(min(testScore), min(aiScore))
    maxScore = max(max(testScore), max(aiScore))
    x = np.linspace(minScore - 1, maxScore + 1, 1000)

    # Calculate probability distributions
    geniuineDist = norm.pdf(x, loc=np.mean(testScore), scale=np.std(testScore)) # Calculate the probability distribution (PDF) for the test scores using a normal distribution
    aiDist = norm.pdf(x, loc=np.mean(aiScore), scale=np.std(aiScore)) # Calculate the PDF for the AI scores using a normal distribution

    plt.figure(figsize=(10, 5))  # Begin plotting the figures with a specified figure size
    plt.plot(x, geniuineDist, 'g') # Plot the PDF for the genuine (test) scores
```

```
    plt.plot(testScore, np.zeros(len(testScore)), 'gs', lw=2, label=label1) # Overlay the test scores on the plot with green squares
    plt.plot(x, aiDist, 'r') # Plot the PDF for the AI scores
    plt.plot(aiScore, np.zeros(len(aiScore)), 'r^', lw=2, label=label2) # Overlay the AI scores on the plot with red triangles
    plt.axvline(thresholdValue, color='b', linestyle='dashed', linewidth=2, label='Threshold') # Draw a vertical line for the threshold value
    plt.xlabel("GMM Scores") # Set the x-axis label
    plt.ylabel("Probability Density Function") # Set the y-axis label
    plt.legend() # Display the legend to describe the elements of the plot
    plt.title(plotTitle) # Set the title of the plot
    plt.show() # Show the plot on the screen
```

Helper Function 4 - Creating Confusion Matrix for Classification

```
def createConfusionMatrix(testScore, aiScore, thresholdValue, confusionMatrixTitle):
    """
    Calculates the confusion matrix and related rates and plots the confusion matrix.

    Parameters
    ----------
    testScore : list
        GMM scores for the real speech samples.
    aiScore : list
        GMM scores for the AI-generated speech samples.
    thresholdValue : int
        The threshold value to determine the decision boundary.
    trueLabel : list of str
        The labels for the true classes.
    predictedLabel : list of str
        The labels for the predicted classes.
    confusionMatrixTitle : str
        The title for the confusion matrix plot.
    """
    # Initialize confusion matrix values
    TP, FP, TN, FN = 0, 0, 0, 0
```

```
    # Calculate True Positives, False Negatives, True Negatives, and False
Positives
    for score in testScore:
        if score < thresholdValue: # Check if below threshold value then increase
the score for FN
            FN += 1
        else:
            TN += 1
    for score in aiScore:
        if score < thresholdValue: # Check if below threshold value then increase
the score for TP
            TP += 1
        else:
            FP += 1

    # -------------- Start of Referenced Code from ENCM 509
Lab06-FaceRec2024.ipynb on D2L---------------
    # True Positive Rate (TPR) also known as Sensitivity or Recall
    TPR = TP / (TP + FN)
    # True Negative Rate (TNR) also known as Specificity
    TNR = TN / (TN + FP)
    # False Positive Rate (FPR)
    FPR = FP / (FP + TN)
    # False Negative Rate (FNR)
    FNR = FN / (TP + FN)

    print("\tTPR: %.2f%%, TNR: %.2f%%, FPR: %.2f%%, FNR: %.2f%%" % (TPR*100,
TNR*100, FPR*100, FNR*100))
    # -------------- End  of Referenced Code from ENCM 509
Lab06-FaceRec2024.ipynb on D2L---------------
    cm = np.array([[TP, FP], [FN, TN]])
    labels = np.array(["AI Generated Speech", "Real Speech"]) # Set the labels
for the matrix
    ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=labels).plot(cmap="plasma") # Create the matrix with passing the
values for the TP,FP, FN, TN to the matrix
    plt.title(confusionMatrixTitle) # Set the title of the confusion matrix
    plt.show() # Show the confusion matrix on the screen
```

Helper Function 5 - Plotting ROC Curve

```
def plotROCCurve(trueLabels, predictionScores, subjectName):
    """
    Plots the ROC curve for the given true labels and prediction scores.

    Parameters
    ----------
    trueLabels : list[int]
        The true binary labels for the data (1 for positive class, 0 for negative
class).
    predictionScores : list
        The prediction scores outputted by the model, indicating the likelihood
of the positive class.
    subjectName : str
        The name of the subject for which the ROC curve is being plotted.
    """

    # -------------- Start of Referenced Code from
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
on Scikit Learn ROC Curve---------------
    # Compute ROC curve and ROC area for each class
    far, tar, thresholds = roc_curve(trueLabels, predictionScores)
    rocAuc = auc(far, tar)
    # -------------- End of Referenced Code from
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
on Scikit Learn ROC Curve---------------
    plt.figure()
    lw = 2
    plt.plot(far, tar, color='darkorange', lw=lw, label=f'ROC curve (area =
{rocAuc:.2f})') # Plot the ROC curve
    plt.scatter(far, tar, color='blue', s=30, label='ROC Points') # Plot the
individual points on the ROC curve
    plt.xlim([0.0, 1.0]) # Set the x-axis limit
    plt.ylim([0.0, 1.05]) # Set the y-axis limit
    plt.xlabel('False Acceptance Rate (%)') # Set the labels for the x axis
    plt.ylabel('True Acceptance Rate (%)') # Set the labels for the y axis
    plt.grid(True) # Add grid lines to plot for better readability
    plt.title(f'ROC Curve for {subjectName}') # Set the title for the curve
    plt.legend() # Display the legend to describe the elements of the plot
```

```
    plt.show()
    threshold_array = np.column_stack((thresholds, far, tar)) # Define the
threshold array with the far and tar

    # Print the threshold array in a formatted way
    print("Thresholds | False Match Rate (FAR) | True Match Rate (TAR)")
    for row in threshold_array:
        print(f"{row[0]:.2f} | {row[1]*100:.2f}% | {row[2]*100:.2f}%")
```

Helper Function 6 - Plotting DET Curve

```
def plotDETCurve(trueLabels, predictionScores, subjectName):
    """
    Plots the DET curve for the given true labels and prediction scores.

    Parameters
    ----------
    trueLabels : list[int]
        The true binary labels for the data (1 for positive class, 0 for negative
class).
    predictionScores : list
        The prediction scores outputted by the model, indicating the likelihood
of the positive class.
    subjectName : str
        The name of the subject for which the DET curve is being plotted.
    """

    # -------------- Start of Referenced Code from
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.det_curve.html
on Scikit Learn DET Curve---------------
    # Compute DET curve and DET area for each class
    far, frr, thresholds = det_curve(trueLabels, predictionScores)
    # -------------- End of Referenced Code from
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.det_curve.html
on Scikit Learn DET Curve---------------
    plt.figure()
    plt.plot(far, frr, color='darkorange', lw=2) # Plot the DET curve
    plt.scatter(far, frr, color='blue', s=30, label='DET Points') # Plot the
individual points on the DET curve
    plt.xlabel('False Acceptance Rate (%)') # Set the labels for the x axis
```

```
    plt.ylabel('False Rejection Rate (%)') # Set the labels for the y axis
    plt.grid(True) # Add grid lines to plot for better readability
    plt.title(f'DET Curve for {subjectName}') # Set the title for the curve
    plt.legend() # Display the legend to describe the elements of the plot
    plt.show()
    threshold_array = np.column_stack((thresholds, far, frr)) # Define the
threshold array with the far and frr

    # Print the threshold array in a formatted way
    print("Thresholds | False Acceptance Rate (FAR) | False Rejection Rate
(FRR)")
    for row in threshold_array:
        print(f"{row[0]:.2f} | {row[1]*100:.2f}% | {row[2]*100:.2f}%")
```

Donald Trump Training Data GMM Model Creation

```
# create a GMM with n components
# -------------- Start of Referenced Code from ENCM 509
Project-07-SpeakerRecongition v02 on D2L---------------
n = 20
donaldTrumpGMM = GaussianMixture(n_components=n) # Initializing the Gaussian
mixture model object
# -------------- End of Referenced Code from ENCM 509
Project-07-SpeakerRecongition v02 on D2L---------------
donaldTrumpMFCCFeatures = extractMFCCFeatures(donaldTrumpTrainingPaths) #
Function call to extract the MFCC features for the training data
print(donaldTrumpMFCCFeatures[0]) # To see the MFCC fetures array
# -------------- Start of Referenced Code from ENCM 509
Project-07-SpeakerRecongition v02 on D2L---------------
donaldTrumpGMM.fit(donaldTrumpMFCCFeatures) # Fitting the Gaussian Mixture Model
to MFCC features
# -------------- End of Referenced Code from ENCM 509
Project-07-SpeakerRecongition v02 on D2L---------------
```

Donald Trump AI GMM Scoring

```
donaldTrumpAIScores = calculateGMMScores(donaldTrumpAIPaths, donaldTrumpGMM) #
Calculating the GMM scores for the AI probe dataset and passing the trained model
print(donaldTrumpAIScores) # Printing the AI generated speech scores
```

Donald Trump Real Speech GMM Scoring

```
donaldTrumpTestScores = calculateGMMScores(donaldTrumpTestingPaths,
donaldTrumpGMM) # Calculating the GMM scores for the testing dataset and passing
the trained model
print(donaldTrumpTestScores) # Printing the testing real speech scores
```

Donald Trump Probability Density Function

```
# Calling the helper function to plot the probability density funciton based on
the test scores and the AI scores
DONALDTRUMPTHRESHOLDVALUE = -46
plotProbabiltyDensityFunction(donaldTrumpTestScores, donaldTrumpAIScores,
DONALDTRUMPTHRESHOLDVALUE, 'Former President Donald Trump Real', 'Former
President Donald Trump AI', 'Former President Donald Trump Distributions of AI
and Genuine Recordings')
```

Donald Trump Confusion Matrix

```
createConfusionMatrix(donaldTrumpTestScores, donaldTrumpAIScores,
DONALDTRUMPTHRESHOLDVALUE,'Former President Donald Trump Confusion Matrix') #
Using the helper function to generate the confusion matrix for Donald Trump
```

Donald Trump ROC Curve

```
donaldTrumpTrueLabels = [1] * len(donaldTrumpTestScores) + [0] *
len(donaldTrumpAIScores) # Create a list of true labels for Donald Trump's speech
samples where '1' represents the genuine class (real speech)
donaldTrumpCombinedScores = donaldTrumpTestScores + donaldTrumpAIScores # Combine
the list of scores from the real speech samples with the scores from the
AI-generated speech samples
plotROCCurve(donaldTrumpTrueLabels, donaldTrumpCombinedScores, 'Former President
Donald Trump') # Plot the ROC curve using the combined scores and the
corresponding true labels for the subject "Donald Trump"
```

Donald Trump DET Curve

```
donaldTrumpTrueLabels = [1] * len(donaldTrumpTestScores) + [0] *
len(donaldTrumpAIScores) # Create a list of true labels for Donald Trump's speech
samples where '1' represents the genuine class (real speech)
donaldTrumpCombinedScores = donaldTrumpTestScores + donaldTrumpAIScores # Combine
the list of scores from the real speech samples with the scores from the
AI-generated speech samples
```

```
plotDETCurve(donaldTrumpTrueLabels, donaldTrumpCombinedScores, 'Former President Donald Trump') # Plot the DET curve using the combined scores and the corresponding true labels for the subject "Donald Trump"
```

Justin Trudeau Training Data GMM Model Creation

```
# create a GMM with n components
n = 20
# -------------- Start of Referenced Code from ENCM 509 Project-07-SpeakerRecongition v02 on D2L---------------
justinTrudeauGMM = GaussianMixture(n_components=n) # Initializing the Gaussian mixture model object
# -------------- End of Referenced Code from ENCM 509 Project-07-SpeakerRecongition v02 on D2L---------------
justinTrudeauMFCCFeatures = extractMFCCFeatures(justinTrudeauTrainingPaths) # Function call to extract the MFCC features for the training data
print(justinTrudeauMFCCFeatures[0]) # To see the MFCC fetures array
# -------------- Start of Referenced Code from ENCM 509 Project-07-SpeakerRecongition v02 on D2L---------------
justinTrudeauGMM.fit(justinTrudeauMFCCFeatures) # Fitting the Gaussian Mixture Model to MFCC features
# -------------- End of Referenced Code from ENCM 509 Project-07-SpeakerRecongition v02 on D2L---------------
```

Justin Trudeau AI GMM Scoring

```
justinTrudeauAIScores = calculateGMMScores(justinTrudeauAIPaths, justinTrudeauGMM) # Calculating the GMM scores for the AI probe dataset and passing the trained model
print(justinTrudeauAIScores) # Printing the AI generated speech scores
```

Justin Trudeau Real Speech GMM Scoring

```
justinTrudeauTestScores = calculateGMMScores(justinTrudeauTestingPaths, justinTrudeauGMM) # Calculating the GMM scores for the testing dataset and passing the trained model
print(justinTrudeauTestScores) # Printing the testing real speech scores
```

Justin Trudeau Probability Density Function

```
# Calling the helper function to plot the probability density funciton based on the test scores and the AI scores
JUSTINTRUDEAUTHRESHOLDVALUE = -48
```

```
plotProbabiltyDensityFunction(justinTrudeauTestScores, justinTrudeauAIScores, JUSTINTRUDEAUTHRESHOLDVALUE, 'Justin Trudeau Real', 'Justin Trudeau AI', 'Justin Trudeau Distributions of AI and Genuine Recordings')
```

Justin Trudeau Confusion Matrix

```
createConfusionMatrix(justinTrudeauTestScores, justinTrudeauAIScores, JUSTINTRUDEAUTHRESHOLDVALUE,'Justin Trudeau Confusion Matrix') # Using the helper function to generate the confusion matrix for Justin Trudeau
```

Justin Trudeau ROC Curve

```
justinTrudeauTrueLabels = [1] * len(justinTrudeauTestScores) + [0] * len(justinTrudeauAIScores) # Create a list of true labels for Justin Trudeau's speech samples where '1' represents the genuine class (real speech)
justinTrudeauCombinedScores = justinTrudeauTestScores + justinTrudeauAIScores # Combine the list of scores from the real speech samples with the scores from the AI-generated speech samples
plotROCCurve(justinTrudeauTrueLabels, justinTrudeauCombinedScores, 'Justin Trudeau') # Plot the ROC curve using the combined scores and the corresponding true labels for the subject "Justin Trudeau"
```

Justin Trudeau DET Curve

```
justinTrudeauTrueLabels = [1] * len(justinTrudeauTestScores) + [0] * len(justinTrudeauAIScores) # Create a list of true labels for Justin Trudeau's speech samples where '1' represents the genuine class (real speech)
justinTrudeauCombinedScores = justinTrudeauTestScores + justinTrudeauAIScores # Combine the list of scores from the real speech samples with the scores from the AI-generated speech samples
plotDETCurve(justinTrudeauTrueLabels, justinTrudeauCombinedScores, 'Justin Trudeau') # Plot the DET curve using the combined scores and the corresponding true labels for the subject "Justin Trudeau"
```