# DEEPFAKE AUDIO DETECTION: A SUPERVISED MACHINE LEARNING APPROACH

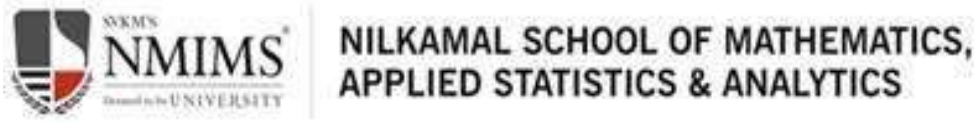
A PROJECT REPORT

SUBMITTED TO
SVKM'S NMIMS (DEEMED- TO- BE UNIVERSITY)

IN PARTIAL FULFILLMENT FOR THE DEGREE OF

**BACHELORS OF SCIENCE**
**IN**
**DATA SCIENCE**

BY
**Sean Minezes A015**
**Diya Rawat A018**
**Veer Shah A020**
**Udita Sinha A025**
**Jeel Paradava A034**




NMIMS NSOMASA
Ground Floor, SBMP Phase I,
Irla, N. R. G Marg, Opposite Cooper Hospital,
Vile-Parle (West), Mumbai – 400 056

APRIL 2024

**Mentored By: Professor Shraddha Sarode**

**CERTIFICATE**

This is to certify that work described in this thesis entitled "DEEPFAKE AUDIO DETECTION: A SUPERVISED MACHINE LEARNING APPROACH" has been carried out by Sean Minezes, Diya Rawat, Veer Shah, Udita Sinha, Jeel Paradava under my supervision. I certify that this is his/her bonafide work. The work described is original and has not been submitted for any degree to this or any other University.

Date:
Place: -

SUPERVISOR
(Dr. )

## ACKNOWLEDGEMENT

We appreciate the opportunity that was provided to us by our respected institution, NSOMASA, NMIMS, Mumbai, to conduct this work. It is with great gratitude that we thank our Professor Shraddha Sarode for all her support, encouragement and guidance to this point. Our creativity is ignited by her uncompromising support and direction to attain our targets. Knowledge, wide experience, and remarkable proficiency have helped us finish the project properly and correctly.

Thank you.

# Table of Contents

## INTRODUCTION

In today's digital landscape, deepfakes, often created with Generative Adversarial Networks (GANs), pose a significant threat to the authenticity of audio content. These synthetic media can manipulate videos and voices with startling realism, as evidenced by a recent Sensity study[1] that found a staggering 10x increase in deepfake detections across various industries. This rapid proliferation underscores the urgent need for effective methods to differentiate between genuine human recordings and AI-generated audio.

Further amplifying this urgency is the growing difficulty in discerning real from AI-generated voices. Research from the University of California, Berkeley [2] found that participants were fooled by deepfakes in over 50% of cases. The consequences of manipulated audio extend far beyond misinformation campaigns. Malicious actors can exploit AI-generated audio for a range of criminal activities, including phishing scams (Armorblox reports a 34% increase in deepfake phishing attempts in 2023 compared to the previous year), social engineering attacks, and even financial market manipulation.

This highlights the critical need for robust solutions to combat the growing threat of deepfakes. Our research project aims to develop a machine learning model capable of accurately identifying the source of audio recordings. By focusing on analyzing speech recordings and extracting features that differentiate between human speech patterns and those generated by AI, we hope to create a dependable tool for audio authentication. This initiative is driven by the belief that robust solutions are crucial to mitigate the risks associated with manipulated audio content and safeguard individuals and organizations in the digital realm.

## OBJECTIVE

To develop an advanced tool for authenticating audio material with precision and reliability. By employing supervised learning techniques, we aim to train our model to differentiate between human-generated and AI-generated audio sources effectively.

Extracting intricate features from the audio data and leveraging them to build a classification system that can reliably identify the origin of the audio.

Deliver a dependable instrument for audio authentication to safeguard against the proliferation of deepfakes and synthetic media, fostering trust and reliability in the digital content landscape.

## LITERATURE REVIEW

| Paper Name | Author Name | Year | Technique for problem addressed | Feature engineering/Feature selection | Evaluation/ Visualization parameter values of evaluation parameter | Dataset Name |
|---|---|---|---|---|---|---|
| The Effect of Deep Learning Methods on Deepfake Audio Detection for Digital Investigation | Mvelo Mcuba, Avinash Singh, Richard Adeyemi Ikuesan, Hein Venter | 2022 | Mel-spectrum, Chromagram, and spectrogram | Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) | Mean Accuracy, Precision, Recall | Baidu Silicon Valley AI Lab |
| Real-Time Detection of AI-Generated Speech for DeepFake Voice Conversion | Jordan J. Bird, Ahmad Lotfi | 2023 | Extreme Gradient Boosting (XGBoost), Random Forests, Quadratic and Linear Discriminant Analyses | Principal Component Analysis | Receiver Operating Characteristic Area Under the Curve (ROC AUC) ,F1 Score | DEEP-VOICE: DeepFake Voice Recognition (Kaggle) |
| Deepfake Audio Detection via MFCC Features Using Machine Learning | Ameer Hamza, Abdul Rehman Javed, Farkhund Iqbal, Natalia Kryvinska, Ahmad S. Almadhor, Zunera Jalil, Rouba Borghol | 2022 | Support Vector Machines (SVM), Gradient Boosting Algorithms | Discrete Cosine Transform (DCT) | Area Under the ROC Curve (AUC-ROC) ,F1 Score, Recall,Confusion Matrix | WaveFake: A data set to facilitate audio DeepFake detection, |

## DATA SOURCE

### LJ Speech: HUMAN AUDIO DATASET[3]

The LJ Speech Dataset is a boon for speech research. This free resource (2.6 GB) offers a meticulously curated collection of speech and text data. Over 13,100 short audio clips (1-10 seconds each) feature a single speaker reading non-fiction text, totaling 24 hours of high-fidelity audio. Corresponding text comes from public domain sources (1884-1964) with meticulously aligned transcripts (UTF-8) and a "normalized" version for clarity. Audio is stored in single-channel 16-bit PCM WAV format (22050 Hz) for accurate sound representation.

Most importantly, the dataset is entirely public domain (audio from LibriVox project), allowing copyright-free research use. This wealth of high-quality, aligned, and freely accessible speech and text data empowers researchers developing next-generation speech technologies.

### WaveFake: AI AUDIO DATASET [4]

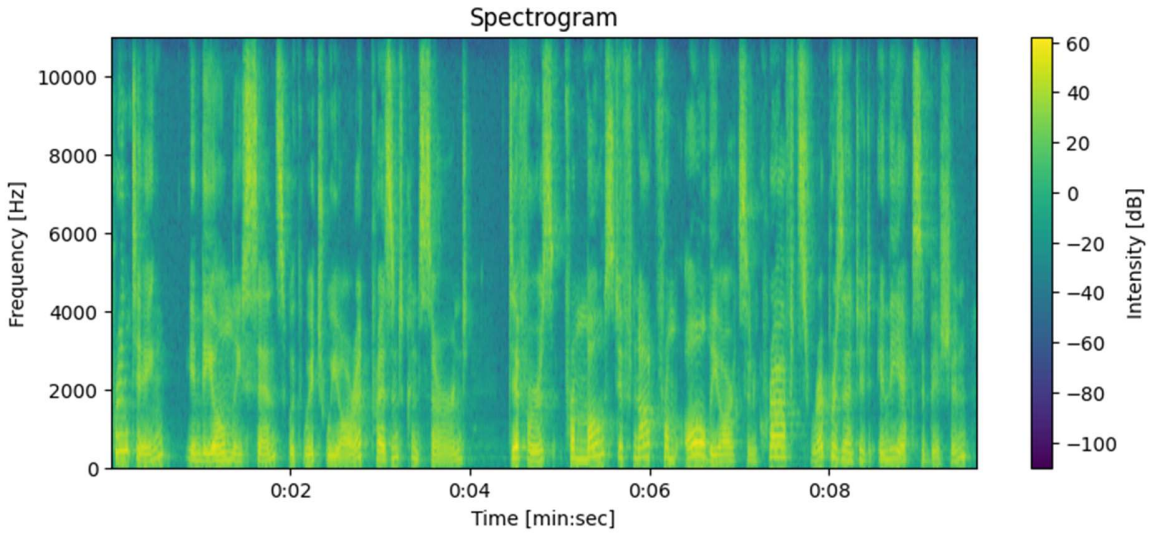
WaveFake tackles a growing challenge: audio DeepFakes used for malicious purposes. This dataset empowers researchers to combat them.

WaveFake provides 117983 generated audio clips (175 hours total) showcasing the capabilities of various deep learning architectures used to create audio DeepFakes (MelGAN, Parallel WaveGAN, HiFi-GAN). This broad representation equips researchers with a holistic view of DeepFake generation techniques. Importantly, WaveFake focuses solely on manipulated audio samples (excluding clean speech data like LJ Speech) to provide researchers with tools for identifying and countering audio DeepFakes. By offering a deep dive into this realm, WaveFake safeguards speech technology integrity.

## DATA PREPARATION

Two sets of data are available: one containing recordings of human speech and the other featuring AI-generated audio samples, both formatted as WAV files. WAV is the standard format for voice classification. Quantification is performed using the Python library "librosa," which specializes in music and audio analysis. In this study, it aids in generating features. Two visualization tools, the oscillogram and the spectrogram, are utilized to analyze audio data. These visualizations are based on two key elements of sound: amplitude and frequency. A spectrogram, specifically, provides a visual representation of the frequency spectrum of a signal over time, illustrating the signal's intensity or "loudness" across different frequencies. In the spectrogram, frequency is represented vertically on the Y-axis in Hertz, time horizontally, and amplitude by brightness, with silence displayed as black and signal presence as varying brightness.

**Picture of spectrogram**




Spectral features like Mel-Frequency Cepstral Coefficients, Tonal Centroids features, Spectral Contrast, and Chromagram are extracted from the audio data. These features produce 2-dimensional arrays, each graphically represented. To simplify analysis, these arrays are converted into 1-dimensional arrays by calculating the mean across columns. Each graph yields a distinct set of features contributing to comprehensive audio analysis. The total dataset comprises 131,083 samples, each with 45 features.

## VARIABLES

### FEATURES:

**1. Mel-frequency Cepstral Coefficients (MFCCs):** (mfcc_0 to mfcc_19)

MFCCs are representation of the audio signal's frequency content, emphasizing aspects relevant to human hearing. These coefficients provide insights into various frequency bands, with the first coefficient typically reflecting overall loudness and subsequent coefficients detailing nuances across different frequency ranges.

**2. Tonal Centroid features:** tonnetz_0 to tonnetz_5

Tonal Centroid features focus on the harmonic content of the audio signal. They calculate the average pitch and its spread within the chromagram, aiding in tasks such as music transcription, chord recognition, and key detection.

**3. Spectral Contrast:** spectral_contrast_0 to spectral_contrast_6

Spectral Contrast measures the amplitude difference between peaks and valleys in the audio spectrum. This feature captures the distinctness of different frequency regions within the audio, providing information about patterns, musical characteristics, and attack/decay characteristics of instruments.

**4. Chromagram:** chromagram_0 to chromagram_11

Chromagram features represent the energy distribution of musical notes in the audio signal, offering insights into the presence and proportion of different musical notes. It provides a comprehensive view of the audio's tonal structure.

All of these combined give a total of 45 features.

### TARGET VARIABLE:

The distribution of the target variables in the dataset is 117,983 instances labeled as "AI" and 13,100 instances labeled as "Human".

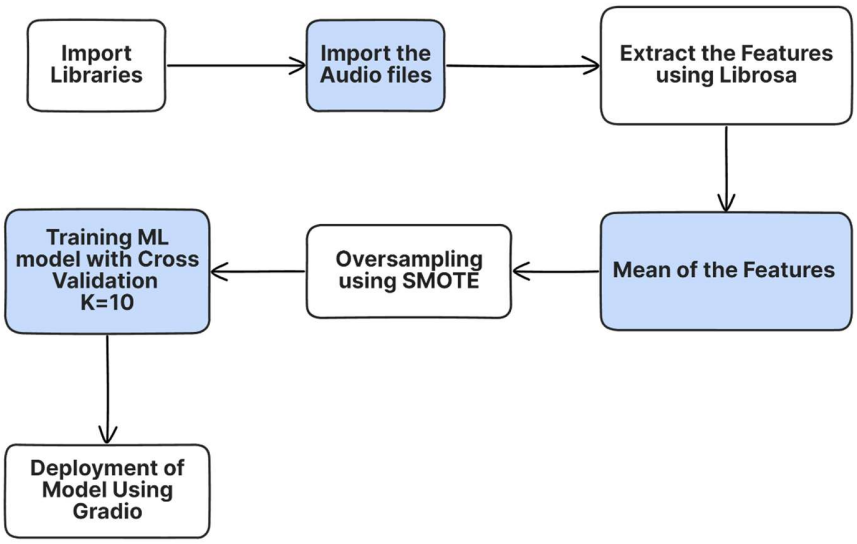
The dataset is seen to be imbalanced.

## METHODOLOGY

### LIBRARIES IMPORTED

1. **Numpy:** Fundamental package for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays
2. **Pandas:** Used for data manipulation and analysis. It consists of data structures and functions to perform efficient operations on data.
3. **Matplotlib:** Used for creating static, animated, and interactive visualizations in Python
4. **Seaborn:** It provides a high-level interface for drawing attractive and informative statistical graphics.
5. **Imbalanced-learn:** Helps in balancing the datasets which are highly skewed or biased towards some classes using resampling techniques.
6. **Scikit-learn:** Helps in data modeling and implementing machine learning models.
7. **Xgboost:** Implements machine learning algorithms under the Gradient Boosting framework

**The representation of code logic is below:**



**Step 1: Importing Librosa**

Librosa is a powerful Python library for working with audio data analysis. It allows us to load audio files and examine them over time, amplitude & its frequency content. It also helps us extract specific characteristics, like Mel-frequency cepstral coefficients (MFCCs), Spectral tonnetz, Chromagram etc. that can reveal unique information about the audio.

**Step 2: Extraction of Features**

| Sr no. | Features Extracted | No. of Features |
|---|---|---|
| 1 | MFCCs | 20 |
| 2 | Tonal Centroid | 6 |
| 3 | Spectral Contrast | 7 |
| 4 | Chromagram | 12 |
| | TOTAL | 45 |

**Step 3: Mean of Features**

After feature extraction, we obtained the features in a 2D matrix segmented into several frames. So we took the arithmetic mean with respect to total frames in order to condense into a 1D array representing the information provided in different time frames for each audio file.

**Step 4: DataFrame - Imbalanced Dataset**

We obtained the Dataframe [131083,47] with total feature columns 47 and rows 131083 i.e. Human and AI audio combined.

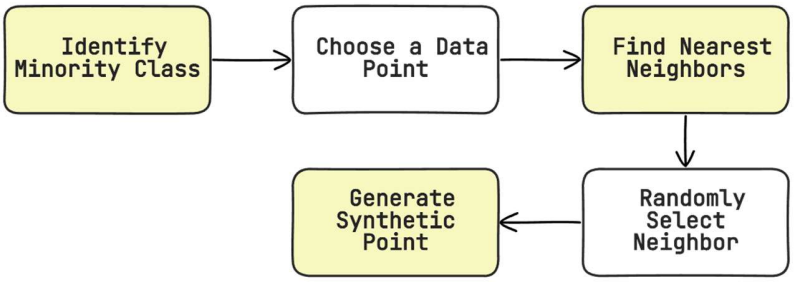
Plotting the distribution of the target variable Human & AI, we found that the data is unbalanced.

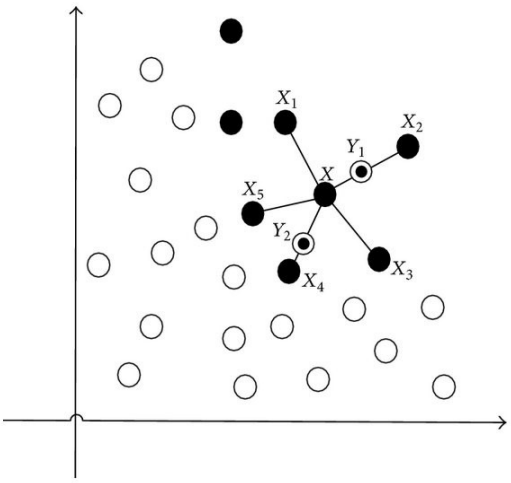
| | |
|---|---|
| **HUMAN** | 13100 |
| **AI** | 117983 |

**Step 5: Synthetic Minority Oversampling Technique (SMOTE)**

SMOTE is a specific oversampling technique effective in addressing class imbalance by synthesizing new instances for the minority class, thus improving the model's ability to learn from underrepresented classes without simply duplicating existing data.

The working of SMOTE is below:




1. **Identify the Minority Class:** SMOTE focuses on the minority class, the class with fewer data points compared to the majority class.
2. **Find Nearest Neighbors:** For each data point in the minority class, SMOTE identifies its k nearest neighbors, also from the minority class. This is typically done using a distance metric like Euclidean distance. The value of k is a parameter you can set (e.g., k=5).
3. **Synthetic Sample Creation:** SMOTE creates synthetic samples based on the minority class data points and their nearest neighbors.
4. **Repeat and Oversample:** Steps 2 and 3 are repeated for a predefined number of times for each data point in the minority class. This generates multiple synthetic samples for each minority class data point, effectively oversampling the minority class.

The following parameters were used:

```
from imblearn.over_sampling.SMOTE
smote = SMOTE(sampling_strategy='minority'', random_state=None,
k_neighbors=5, n_jobs=None)
```

1. **sampling_strategy='minority':** Targets the minority class for oversampling.

2. **random_state=None:** The randomness of the algorithm is not fixed, allowing for different results on each run.

3. **k_neighbors=5:** Uses 5 nearest neighbors to generate synthetic samples.

4. **n_jobs=None:** Uses one processor for the computation, not parallelizing the task.

Here the minority class is human with 131083 instances.

Hence, now the dataset is balanced - [235966,47] with total 235966 audio rows and 45 feature columns. Below is the balanced class:

| | |
|---|---|
| **HUMAN** | 117983 |
| **AI** | 117983 |

**Step 6: Training the Machine Learning Models**

I. **Naive Bayes Model: GaussianNB**

Based on Bayes' theorem, Gaussian Naive Bayes calculates the probability of a certain event occurring given prior knowledge.

```
class sklearn.naive_bayes.GaussianNB(*,priors=None,var_smoothing=1e-09)
```

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Assumes that continuous features within each class are normally distributed and mutually independent. It performs poorly when these assumptions are violated, such as in the presence of skewed or non-normally distributed features, and when features are not mutually independent.

II. **Logistic Regression Model**:

Logistic Regression model predicts probability of an instance belonging to a particular category using a linear relationship between the features and the log-odds of the target variable. Value is between 0 and 1

$$p = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\ldots+\beta_n x_n)}}$$

$$y = \ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

.

*class* sklearn.linear_model.**LogisticRegression**(*penalty='l2'*, *, *dual=False*, *tol=0.0001*, *C=1.0*, *fit_intercept=True*, *intercept_scaling=1*, *class_weight=None*, *random_state=None*, *n_jobs=None*, *l1_ratio=None*)

Assumes little to no multicollinearity among independent variables (features). Multicollinearity occurs when two or more features are highly correlated. It performs poorly when multicollinearity is present, as it relies on the assumption that independent variables are not highly correlated.

III. **Decision Tree Model**:

This model makes decisions by asking a series of questions about the input features and predicts the target variable based on the answers, making them intuitive and easy to interpret. Split the variable with the least gini index.

$$\text{Gini}(p) = 1 - \sum_{i=1}^{C} p_i^2$$

$$\mathbf{gini\_coef(p) = 1 - p^2 - (1-p)^2 = p * (1-p) + (1-p) * p}$$

*class* sklearn.tree.**DecisionTreeClassifier**(*, *criterion='gini'*, *splitter='best'*, *class_weight=None*)

IV. **KNN**:

KNN predicts the class of a new instance based on the majority class among its k nearest neighbors in the feature space i.e. measures Distance between K nearest Neighbors

$$\text{Distance}(\mathbf{x}_i, \mathbf{x}_{\text{new}}) = \sqrt{\sum_{j=1}^{n} (x_{ij} - x_{\text{new},j})^2}$$

*class* sklearn.neighbors.**KNeighborsClassifier**(*n_neighbors=5*, *, *weights='uniform'*, *n_jobs=None*)

The performance can degrade in high-dimensional spaces due to the "curse of dimensionality," where distance-based algorithms struggle with increased dimensionality. Here with 45 dimensions in the dataset, KNN may not perform well due to this curse of dimensionality.

V. **Ensembling Models - Random Forest & XG Boost**

*class* sklearn.ensemble.**RandomForestClassifier**(*n_estimators=100**, *criterion='gini'*)

Are effective on high-dimensional data, including datasets with many features.

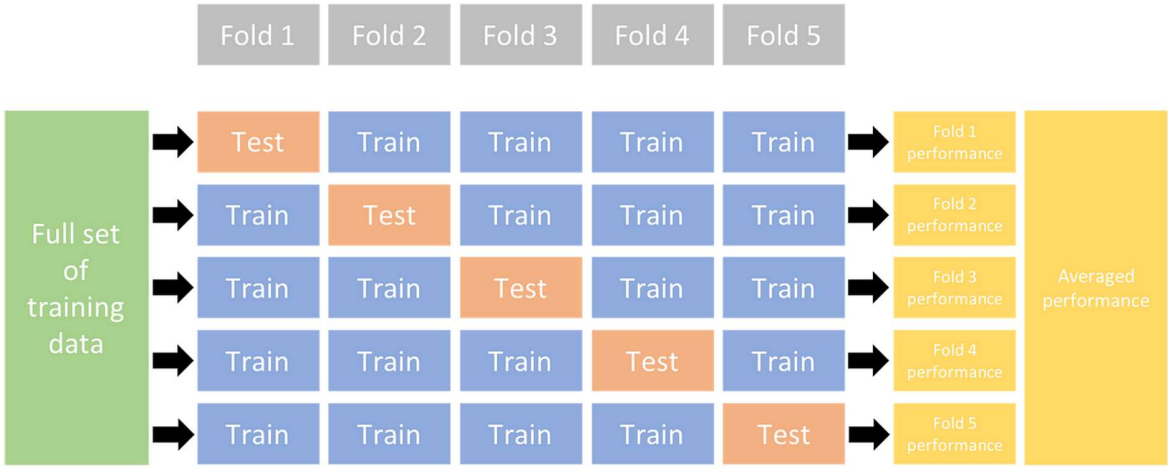
They are robust to multicollinearity, as they can handle correlated features well. It can capture nonlinear relationships between features and the target variable, unlike linear models. They are generally robust to noisy data, with Random Forest being particularly effective due to its ensemble averaging effect.It can handle outliers better than some other algorithms like linear regression, although extremely influential outliers can still affect performance to some extent.

**Step 7: K-Fold Cross-Validation**

Cross-validation is a technique for evaluating a machine learning model and testing its performance. K-Fold gives a more stable and trustworthy result since training and testing is performed on several different parts of the dataset. We can make the overall score even more robust if we increase the number of folds to test the model on many different sub-datasets.

sklearn.model_selection.**cross_validate**(*estimator*, *X*, *y=None*, *,*scoring=None*, *cv=None*, *n_jobs=None*, *params=None*,)

The working of cross validation is below:




The algorithm of the k-Fold technique:

1. Pick a number of folds – k. Usually, k is 5 or 10 but you can choose any number which is less than the dataset's length.
2. Split the dataset into k equal (if possible) parts (they are called folds)
3. Choose k – 1 folds as the training set. The remaining fold will be the test set
4. Train the model on the training set. On each iteration of cross-validation, you must train a new model independently of the model trained on the previous iteration
5. Validate on the test set
6. Save the result of the validation
7. Repeat steps 3 – 6 k times. Each time use the remaining fold as the test set.
8. To get the final score average the results that we got on step 6.

Here, we did 10 Fold cross validation.

**Step 8: Evaluation Metrics**

**Confusion Matrix**

Visually summarizes the model's performance on a classification task. It shows the number of correctly and incorrectly classified data points for each class. Provides a clear picture of where the model is making mistakes.

We take an example here:

| | | **Actual** | |
|---|---|---|---|
| | | **Positive** | **Negative** |
| **Predicted** | **Positive** | **540 (True Positive)** | **150 (False positive)** |
| | **Negative** | **110 (False Negative)** | **200 (True Negative)** |

Rows represent actual classes, columns represent predicted classes. Values in the table represent counts of data points in each category - True Positives, True Negatives, False Positives, False Negatives

**1. Accuracy:**

The proportion of correctly classified data points. It represents the overall effectiveness of the model in making accurate predictions.

$$Accuracy = \frac{Number\ of correct\ predictions}{Total\ number\ of prediction}$$

$$\frac{540 + 200}{540 + 110 + 150 + 200} = 0.74$$

An accuracy of 0.74 in a confusion matrix indicates that 74% of the predictions made by the model are correct.

**2. Precision:**

The proportion of predicted positive labels that are actually correct (positive predictive value).

$$\text{Precision} = \frac{True\ Positives}{True\ Positives\ +\ False\ Positives}$$

$$= \frac{540}{540 + 150} = 0.7826$$

A precision of 0.7826 means that out of all the positive predictions made by the model, approximately 78.26% were correct. This indicates a relatively good ability of the model to avoid false positives.

**3. Recall:**

The proportion of actual positive labels that are correctly predicted (sensitivity).

$$Recall = \frac{True\ Positives}{True\ Positives\ +\ False\ Negatives}$$

$$\frac{540}{540 + 110} = 0.8307$$

A recall value of 0.8307 means that the model correctly identified approximately 83.07% of all actual positive instances in the dataset. This indicates a good ability of the model to capture the positive instances (true positives) in the dataset.

**4. F1-Score:**

Harmonic mean of precision and recall, addressing the limitations of each metric individually.

$$F1 - Score = 2 \cdot \frac{precision \cdot recall}{precision\ + recall}$$

$$= 2 \cdot \frac{0.7826 \cdot 0.8307}{0.7826 + 0.8307} = 0.8059$$

An F1 score of 0.8059 represents the harmonic mean of precision and recall, balancing both metrics. This score indicates a reasonably good overall performance of the model in terms of both precision (correctness of positive predictions) and recall (ability to capture positive instances). A higher F1 score closer to 1.0 generally signifies better overall performance in binary classification tasks.

## RESULTS & DISCUSSION

### MODEL COMPARISON TABLE

| Technique | Models | Accuracy | Precision | Recall | F1 score |
|---|---|---|---|---|---|
| SMOTE | Random Forest | 0.98 | 0.97 | 1 | 0.98 |
| | Decision Tree | 0.96 | 0.97 | 1 | 0.96 |
| | Xgboost | 0.92 | 0.87 | 1 | 0.93 |
| | Naive Bayes | 0.79 | 0.72 | 0.93 | 0.81 |
| | KNN | 0.74 | 0.66 | 0.97 | 0.79 |
| | Logistic Regression | 0.66 | 0.63 | 0.79 | 0.70 |

### BEST MODEL - RANDOM FOREST

| Best Model | Accuracy | Precision | Recall | F1 score |
|---|---|---|---|---|
| Random Forest | 0.98 | 0.97 | 1 | 0.98 |

### INTERPRETATION

With a high accuracy of 98%, the Random Forest model correctly predicts the class labels for the majority of instances in the dataset. A precision of 0.97 indicates that when the model makes a positive prediction, it is highly likely to be correct i.e. it is correct about 97% of the time. The perfect recall score of 1.00 means that the model correctly identifies all positive instances in the dataset.With an F1 score of 0.98, the Random Forest model achieves a high balance between precision and recall, indicating robust performance across both metrics and making it well-suited for classification. Hence, the Random Forest model demonstrates exceptional performance across all evaluation metrics, achieving high accuracy, precision, recall, and F1 score.

## CONCLUSION

Some of the growing security implications of generative AI have been addressed in this study, especially those concerning spoofing with artificial intelligence-generated human speech. Given the rapid progress in system quality, it is critical that systems provide real-time transparency about the veracity of human voices over the phone or in conferences.

Speech produced by artificial intelligence (AI) has the potential to be misused, such as in social engineering techniques where impersonation is used. This work makes three contributions: first, it creates an original dataset for audio classification that can be used in other studies; second, it thoroughly analyzes the statistical significance of audio features extracted from artificial intelligence-generated speech as well as real speech; and third, it develops machine learning models that can predict speech legitimacy.

## LIMITATIONS & FUTURE SCOPE

- Expanding feature extraction for deeper audio comprehension.
- Utilizing deep learning methods such as CNNs, RNNs, and other neural networks for efficient model development.
- Increasing the availability of human-labeled data.
- Acquiring up-to-date AI-generated data for real-time audio classification and detection, paving the way for future advancements.

## APPLICATIONS

Real-time AI audio detection acts as a security guard for our digital interactions. It can:

- Fake Voice Recognition: Stop identity theft by spotting fake voices during calls with banks and other sensitive services.
- Secure voice authentication system: like phone unlocking by identifying real users from AI mimics.
- Social Media DeepFake Detection: It will help social media platforms weed out false, misleading deepfake content before it spreads.
- Detects AI generated calls & Voice Impersonation: Protect public from voice impersonation  preventing scams and manipulation.
- Overall, it fosters a more secure and trustworthy online environment for everyone.

## REFERENCES


[1] S. Team, “Reports - Sensity AI,” *Sensity*, Dec. 05, 2023. https://sensity.ai/reports/

[2] Dessa, “Detecting audio deepfakes with AI - dessa news - medium,” *Medium*, Dec. 12,2021.[Online].Available:https://medium.com/dessa-news/detecting-audio-deepfakes-f2edfd8e2b35

[3] The LJ Speech Dataset (keithito.com)

[4] WaveFake: A data set to facilitate audio DeepFake detection (zenodo.org)

[5] Feature extraction — librosa 0.10.1 documentation

[6] E. Daehnhardt, “Audio Signal Processing with Python’s Librosa,” *Elena’s AI and Python Coding Blog, Living With AI daehnhardt.com*, Mar. 05, 2023. https://daehnhardt.com/blog/2023/03/05/python-audio-signal-processing-with-librosa/

[7] https://wwwaltexsoft.com/blog/audio-analysis/

[8] R.Wells,“Upsampling and Downsampling Imbalanced Data in Python,”*wellsr.com*. https://wellsr.com/python/upsampling-and-downsampling-imbalanced-data-in-python

[9] Mcubaa, M., Singha, A., Ikuesanb, R. A., & Ventera, H. (2023). The Effect of Deep Learning Methods on Deepfake Audio Detection for Digital Investigation. In M. Mcubaa, A. Singha, R. A. Ikuesanb, & H. Ventera (Eds.), Procedia Computer Science (Vol. 219, pp. 211–219). Elsevier B.V.

[10] Bird, J.J., Lotfi, A. (Nottingham Trent University, UK). "REAL-TIME DETECTION OF AI-GENERATED SPEECH FOR DEEPFAKE VOICE CONVERSION." In Proceedings of the International Conference on Artificial Intelligence (ICAI), 1st edition. Nottingham, UK: Nottingham University Press, 2023

[11] A. Hamza et al., "Deepfake Audio Detection via MFCC Features Using Machine Learning," in IEEE Access, vol. 10, pp. 134018-134028, 2022, doi: 10.1109/ACCESS.2022.3231480.