جامعة الملك سعود
King Saud University

IT461
Practical Machine Learning

Audio DeepFake Detection

Supervised by:
Dr. Yousra Almathami

King Saud University

College of Computer and Information Sciences

Department of Information Technology

First Semester 1446H – Fall 2024

IT461 – Practical Machine Learning

Machine Learning Project

Audio DeepFake Detection Model

Prepared by:

Student Name:	Student ID:
Maryam Altuwaijri	443200235
Sarah Alhindi	443200630
Wassayef Alkherb	443200459
Shahad Bin Makhashen	443204385
Arwa Mesloub	443203895

Supervised by: Dr. Yousra Almathami

Table of Contents

List of Figures

List of Table

1. Introduction

In this technology-driven era where AI models are constantly improving, a new concept has been emerging; that is DeepFake. DeepFake can be defined as the capability to produce realistic images, audio, and videos using highly advanced computer graphics and AI algorithms where it becomes difficult to distinguish between the real and fake media [1]. As audio manipulation in deepfake is advancing rapidly, the need for robust detection models increased significantly, and as it can be used in many areas, like Fraud, fake news spread, and political manipulation, our solution aims to enhance trust and integrity in audio communications, providing the community with the assurance and awareness they need in this field.

This report presents the motivation behind our project, background research in audio deepfake field, and description of our chosen dataset.

2. Motivation

In recent years, the rapid advancement of AI resulted in the development of deep learning technologies, which led to the emergence of audio deepfake; these manipulated audio recordings that mimic real voices can be used to spread misinformation, commit fraud, or damage reputations, causing risks, distrust, and integrity issues pertaining the audio communications and shared information. As tools, including software and hardware for creating deepfake content become more accessible, the possibility for misuse increases. The core problem our model addresses is the detection of these manipulated audio files, distinguishing them from real audio.

The importance of tackling this issue includes protection against misinformation and fraud that can lead to severe consequences, both at the societal and individual levels, as deepfake can affect politics, media, and personal relationships. Misinformation propagated through deepfake audio can influence public opinion, sway elections, and disrupt social relationships. Our model seeks to empower people by providing a reliable and high accuracy means of detecting deepfake audio.

Our motivation rises from our desire to protect individuals from fake media and foster their trust in digital audio communications. In addition, the team wants to work on their ML models development skills, performance optimization, and experiment with a new dataset type -for the team-, i.e.: Audio dataset.

To address the issues above, we are focusing on developing an Audio DeepFake Detection model, aiming to protect victims, individuals, and organizations from consequences of these manipulations and impersonation by building up strong machine learning model. We will train the dataset using the DEEP VOICE dataset [2], this dataset is a collection of real human speeches from eight famous people whose speeches have been converted to other voices using some voice conversion method. We will do some data preprocessing to enhance its quality, extract meaningful features, and prepare it for further analysis. This processed data then feeds the core machine learning model, which at the backbone of algorithms and analytical capabilities, will finally achieve the goal of detecting with most precision whether the audio is fake or real (See Figure 1).

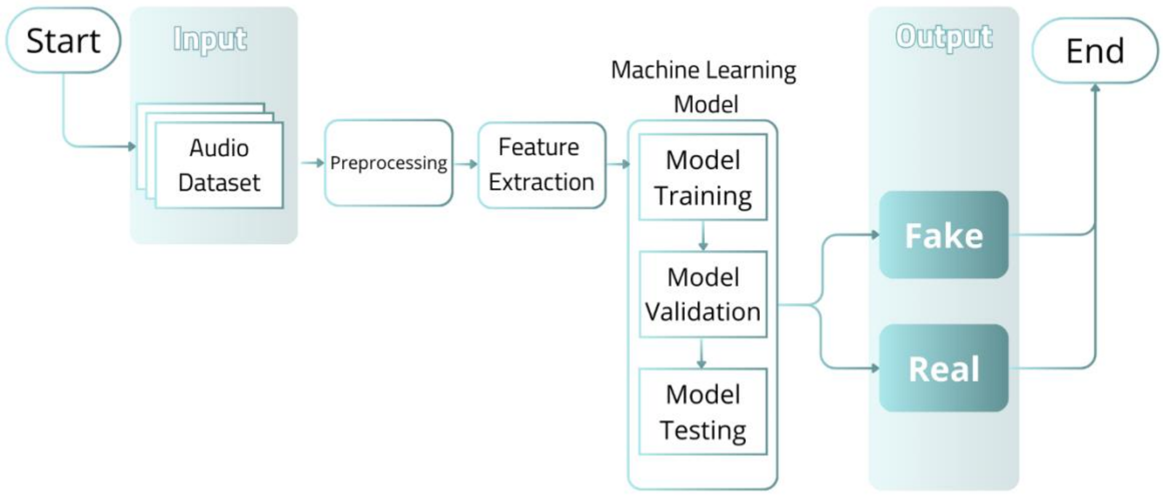

Figure 1: Workflow of deepfake audio detection.

A variety of performance metrics such as accuracy, precision, recall, and F1 score will be performed under a test protocol in order to measure the success of our model. These metrics give us clear indication about the performance in distinguishing between real and fake audio, making the proposed model reliable for practical applications.

3. Background

This section presents a foundational understanding of key concepts related to audio deepfake, explores various detection techniques, and summarizes several works related to audio deepfake detection, aiming to learn from their attempts and identify the gaps the field currently lacks.

• DeepFake Technology

Deepfake technology relies heavily on sophisticated algorithms to learn and make inferences from vast input data. In the area of audio, deep fake devices can reproduce a person's voice by analyzing pre-existing voiced samples to generate new speech that sounds like them. To do this effectively allows techniques such as GANs or LSTMs to simulate some characteristics of human voices, such as intonation patterns (how high or low notes are) and rhythm (melody's timing) [3].

• Voice Recognition and Manipulation

Voice recognition systems are supposed to recognize and process human sound. However, it has become increasingly difficult to discern between real speech and fake recordings due to the emergence of deep fake sound. Malicious purposes such as monetary trickery, identity appropriation, or disinformation campaigns can all be associated with deep audio fakes, which increases concerns over safety and faith in audio communications [4].

• Detection Techniques

Several biometric techniques were used to detect audio deepfakes. Spectral analysis examines audio signals to identify distinct voice patterns, revealing discrepancies that may indicate manipulation. While Deep-learning algorithms analyze individual voices, recognizing unique characteristics that are challenging to replicate in deepfakes. Detecting artifacts is also essential, as digital traces in deepfakes can result in unnatural breaks or unexpected background noises. Additionally, biometric systems can compare voices against known samples to enhance detection capabilities. These techniques primarily involve passive liveness testing, which requires no additional action from the user. To improve accuracy, active liveness testing can also be implemented, prompting the individual to pronounce a randomly generated phrase from a predefined list, ensuring the authenticity of the voice being analyzed [5].

3.1 Related Works

1- Efficient Deepfake Audio Detection Using Spectro-Temporal Analysis and Deep Learning [6].

This paper discusses the process of creating an Audio deepfake detection solution, using deep learning and Spectro-Temporal analysis.

The model development starts with using the ADD2022 dataset which is a dataset containing audio clips for the training, validation, and testing. The next step is the data preprocessing phase, where audio samples are resampled, normalized, and subjected to silence removal to ensure uniformity and enhance model input quality, feature extraction has been implemented using Spectro-Temporal analysis, which involves converting audio signals into spectrogram representations. This process uses the Short-Time Fourier Transform (STFT) to translate time- domain signals into the frequency domain, capturing the spectral content over time. The deep learning model architecture is hybrid, combining Convolutional Neural Networks (CNNs) for extracting spectral features and Recurrent Neural Networks (RNNs) or Long Short-Term Memory (LSTM) networks for analyzing temporal dynamics. The performance is then evaluated against key metrics such as accuracy, precision, F1, recall score, and the Equal Error Rate (EER) (see Figure2 for performance metrics of different models), with a designated portion of the dataset reserved for validation and testing to assess generalization performance. The training process utilizes a cross-entropy loss function for binary classification (genuine vs. deepfake), optimized using an Adam optimizer. Hyperparameters, including learning rate and batch size, are fine-tuned based on validation set performance. Regularization techniques, such as give up and weight decay, are implemented to avoid overfitting and improve model robustness. The paper findings exhibit the Enhanced Deepfake Audio Detection model's capability to distinguish deepfake audio.

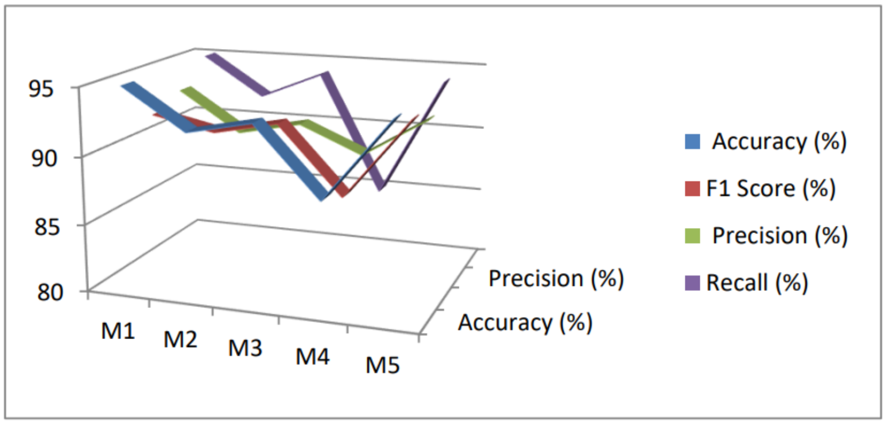

Figure 2: Performance of models on different parameters.

2- Training-Free Deepfake Voice Recognition by Leveraging Large-Scale Pre-Trained Models [7].

This paper addresses the increasing problem of identifying audio deep fakes, especially regarding the generalization of detection methods. The authors emphasize that traditional detectors of audio deep fakes face difficulties when dealing with out-of-distribution data, so it is vital to create techniques that work across different datasets.

The proposed solution reformulates the detection problem within a speaker verification framework. It uses a one-class approach, where only genuine voice samples of the claimed identity are used during evaluation instead of depending on training with fake audio samples. This innovative strategy ensures robust generalization by removing the necessity for training on specific cases of deep fake. The process begins with extracting audio features from large-scale pre-trained models such as Wav2Vec2-xlsr or BEATs, which enable efficient representation of these signals without requiring fine-tuning to specific datasets. During detection, the model compares the unknown audio against a reference set of real audio from the same identity. Authors use the cosine similarity measure to calculate how much test audio is similar to reference samples; the maximum similarity score is used as a decision criterion. This method exploits the intrinsic variation in the speaker's voice while ensuring high accuracy.

The Evaluation is done based on widely used datasets such as ASVSpoof2019 and InTheWild which detectors based on pre-trained models outperform the traditional supervised methods. Not only does this method provide a feasible substitute for older techniques, but it also exhibits substantial benefits concerning its capacity for generalization. Overall, these results support the efficiency of using large-scale pre-trained models in audio deepfake detection without recourse to synthetic training data, thus opening up several avenues for promising research in audio forensic science.

3- Detecting Deepfake Voice Using Explainable Deep Learning Techniques [8].

This paper tackles the challenge of detecting deepfake audio. The authors highlight the limitations of existing detection methods, particularly their lack of interpretability, which raises concerns about their reliability. These issues will be addressed by the authors using explainable AI techniques, usually applied in image classification, adapting them to the detection of audio deepfakes. They put forward a human-centered interpretability framework in order to make the decisions made by the model more understandable to non-experts.

The methodology will involve using simple CNNs and LSTMs, which are mainly involved in the analysis of audio features that were extracted through spectrograms. The study evaluates models using two datasets: ASVspoof 2019 with both authentic and synthesized audio, and LJSpeech for voice synthesis.

The results of the work are presented to show that by applying methods of XAI, valuable insights about model behavior can be obtained, presenting different attribution patterns which may provide a better understanding of deepfake detection mechanisms. Based on that, the paper contributes much to advocating for the approach of dual-modal interpretability, combining both visual and auditory analysis while bringing forward the need for transparency in AI systems in the effective combating of deepfake technologies.

4- Deepfake Audio Detection via MFCC Features Using Machine Learning [9].

This paper investigates the detection of deepfake audio using machine learning methods, primarily with Mel-frequency cepstral coefficients for the key feature extraction of the audio. Deepfake audio really sounds real and can be dangerous because this could be used for fraud or false information dissemination. The researchers used the Fake-or-Real dataset which has more than 195,000 audio samples and was categorized based on length and quality. They then ran various machine learning models including a Support Vector Machine, Random Forest, and Gradient Boosting to determine which one can classify the audio as fake with maximum accuracy. These results indicated that SVM worked best for smaller lengths of audioThat is important because it improves the ways to detect deepfake audio and can protect people from its possible harmful effects in society. In all, these would contribute to better security with more confidence in digital communications.

5- Investigation of Deepfake Voice Detection Using Speech Pause Patterns: Algorithm Development and Validation [10].

This study explores the creation of a model designed to detect deepfake audio by focusing on the distinct biological traits of human speech, which differ from machine-generated sounds. Audio samples were generated using Descript, ElevenLabs, and Podcastle. The study analyzed various pause patterns in these cloned samples and compared them to the original recordings. The underlying assumption was that differences in pause patterns could reveal whether a recording was real or fake, since machines do not need to pause for natural human functions like breathing or swallowing. The findings confirmed this theory, with unnatural pauses serving as a major indicator of fake audio. Five specific pause-related features were incorporated into a classification system, which demonstrated an accuracy of about 85%

when distinguishing between real and deepfake audio, using speech duration and pause characteristics as the key factors.

4. Dataset

The DEEP-VOICE dataset from Kaggle is a collection of real human speeches from eight famous people whose speeches have been converted to other voices using Retrieval-based Voice Conversion (RVC), each of these audio sources acts as the source speech, which is then converted into AI-generated speech. Table 1 (based on the data documentation [2]) lists all the sources of speech used. In total, 62 minutes and 22 seconds are collected from eight speakers, without letting any audio track be over ten minutes long. Some of the tracks are noisier than others; some are fairly clear. This may affect the quality of the conversion. In general, this dataset is pretty diverse and serves as a good starting point for training our model.

Table 1: Data collected for training, validation, and unseen testing for the experiments in this work.

Individual	Source	Length (MM:SS)
Joe Biden	Victory Speech	10:00
Ryan Gosling	Golden Globes Speech	1:33
Elon Musk	Commencement Speech	10:00
Barack Obama	Victory Speech	10:00
Margot Robbie	BAFTAs Speech	1:19
Linus Sebastian	Stepping Down Monologue	9:30
Taylor Swift	Women in Music Speech	10:00
Donald Trump	Victory Speech	10:00
Total		**62:22**

The names of the audio filenames indicate whose actual speech it is and to whose voice it has been converted. For example, "Obama-to-Biden" means the speech of Barack Obama has been converted to the voice of Joe Biden.

RVC is an interesting technology that's used to make voices sound different or to mimic someone else's voice in a conversation. Imagine having a chat with a friend, and you can make it sound like a famous movie star, your favorite cartoon or TV character, or even a historical figure [11] (See Figure 3).

Unlike traditional methods, RVC utilizes advanced algorithms to clone voices with remarkable accuracy. This technology has been pivotal in developing realistic AI cover voices and voice generators [12].

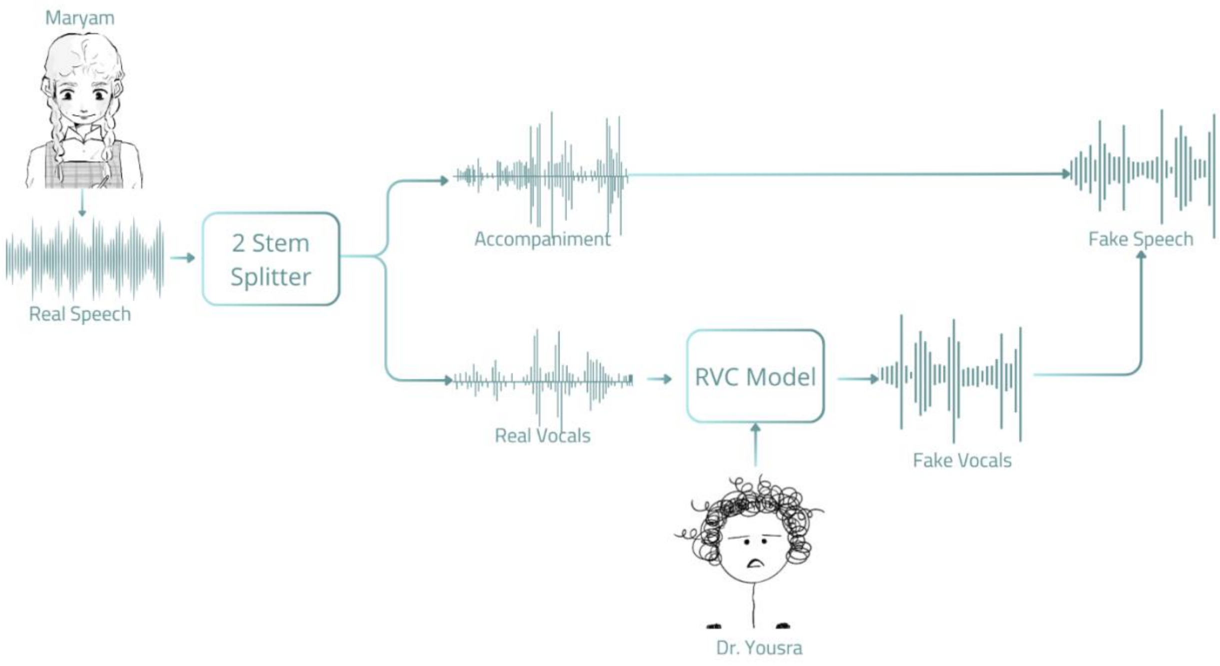

Figure 3: Overview of the Retrieval-based Voice Conversion process to generate DeepFake speech with Maryam's speech converted to Dr. Yousra.

The actual speech is separated, using Spleeter's two-stem model in a U-Net encoder-decoder convolutional neural network [13]. Once the actual vocals and accompaniment tracks have been split, the vocals are then converted with a Retrieval-based Voice Conversion vocal model into another person's voice. Finally, the original accompaniment and the RVC vocals are mixed to create a fake speech track. The reason for splitting the tracks is so that the style of the deepfake voice is not transferred onto any background noise, like the cheers or murmur of an audience. In other words, the idea of this approach is to preserve ambient sounds while converting only the speaker's voice.

We chose this dataset because of its range of audio samples that are important and useful for training our models effectively. This dataset focuses on deepfake audio and is thus aligned with our problem. A lot of labelled data in it helps to differentiate between real and manipulated voices which is necessary in supervised learning approach. In summary, the DEEP-VOICE dataset provides us with required tools to make a better and more accurate as well as dependable system for detection of audio deep fakes in digital age whose need is increasing. By using this dataset, we aim at contributing towards ongoing efforts against manipulation of audios ensuring their originality.

4.1 Features description

Feature extraction was carried out by a previous work on the same dataset, and we will build our model on it. As this process requires significant GPU resources, we will make use of the features extracted in “Real-Time Detection of AI-Generated Speech for Deepfake Voice Conversion” study [14]. In this study, the voice conversion process, which utilized the CREPE algorithm, was executed on a GPU. Specifically, an Nvidia RTX 2080Ti with 4,352 CUDA cores was used for this purpose. For audio feature extraction, the Librosa library was employed by the original study.

Features are extracted for every 1-second of audio signal. In total, 26 features are extracted that describe various sound properties. Some of these features are chroma_stft, rms, spectral_centroid, and mfcc coefficients. The dataset also has one label column, where the files are classified, with possible values "FAKE" and “REAL”. The features are:

- **Chroma STFT:** It looks at the energy in different notes in an audio clip, it helps in identifying melodies and harmonies in music or speech.
- **RMS (Root Mean Square):** This measures how loud the sound is on average, it helps to understand the overall volume or energy in a sound.
- **Spectral Centroid:** It tells us where the "center" of the sound frequencies is, meaning whether the sound is more high-pitched or low-pitched, helps differentiate between sharp and soft sounds.
- **Spectral Bandwidth:** This measures how wide the range of frequencies in the sound is, it helps differentiate between simple sounds (like a whistle) and complex sounds (like crowd noise).
- **Rolloff:** This measures the frequency below which most of the energy in the sound is found, it helps to separate smooth, harmonic sounds (like a guitar) from more abrupt, noisy sounds (like drums).
- **Zero-Crossing Rate:** This checks how often the sound wave crosses the zero point (goes from positive to negative), It’s useful for identifying sharp, noisy sounds like claps or clicks.
- **MFCC (Mel-Frequency Cepstral Coefficients):** These are numbers that summarize the shape of the sound wave in a way that’s similar to how humans hear, it’s one of the best tools for recognizing different types of sounds, whether it’s someone speaking or an instrument playing.

4.2 Summary Statistics

The dataset consists of a total of 64 files, with 56 FAKE audio files and 8 REAL audio files that are all in WAV format. The sizes of the files range from 13.3 MB to 109 MB. For the most part, they are almost 100 MB, such as a 15.7 MB file (1 minute 33 seconds), another one at 13.3 MB (1 minute 19 seconds), while others are bigger still, like 95 MB (9 minutes) 100 MB (10 minutes), and one at 109 MB (11 minutes). The duration for audio samples is between 1 minute, 19, and 11 minutes, averaging around 10 minutes per sample. The bit rate is 1411 kbps for these audio files, divided into two classes : “FAKE” or “REAL.”

Table 2: Summary Statistics.

Attribute:	Details
Total Number of Audio Files	64 (56 FAKE audios and 8 Real audios)
Audio Format	WAV
File Size Range	13.3 MB to 109 MB
Common File Size	Most files close to 100 MB
Specific Audio File Sizes And length	15.7 MB (1 minute 33 seconds), 13.3 MB (1 minute 19 seconds) 95 MB (9 minutes), 100 MB (10 minutes), 109 MB (11 minutes)
Duration Range	1:19 minutes – 10 minutes
Average Duration	10 minutes
Bit Rate	1411 kbps
Classification Labels	2 class labels (potential values: "FAKE", "REAL")

The visual comparison of the wave patterns in the graph -see Figure 4- indicates apparent differences between FAKE and REAL audio data, as the FAKE data displays irregular and noisy patterns. In contrast, REAL data shows smooth and leveled patterns since it is more consistent and has stable waveforms. Therefore, this graph gives a visual hint that there may be quality disparity and authenticity dissimilarity within both groupings of audio recordings.

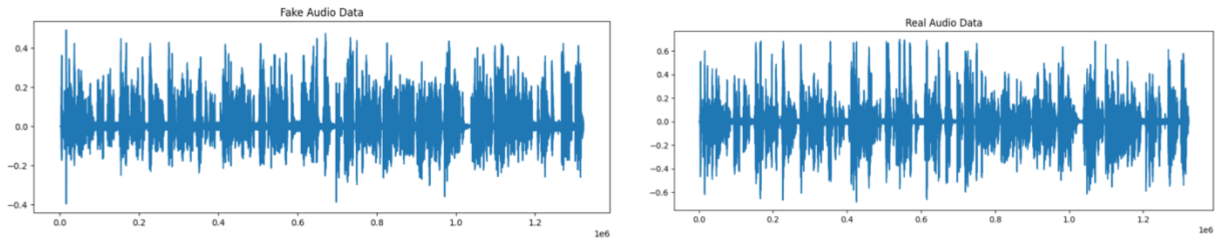

Figure 4: representative example of a dataset showing Fake, and Real audio data

4.3 <u>Data preprocessing</u>

Our audio dataset required processing and preparation to ensure our models perform well on it. This included:

1. Noise Cancellation: We used “noisereduce” library to eliminate unwanted background noise and improve audio clarity.

2. Feature Extraction: We derived the main and most important features from the audio files we have, to capture temporal and spectral characteristics, which can help in classification.

3. Resampling: We addressed the class imbalance in our data using the SMOTE (Synthetic Minority Over-sampling Technique) method.

For each model, additional preprocessing steps were tailored according to each model’s requirements and pre-defined functions:

- Scaling and Encoding: Scaling was used to standardize the range of features, ensuring they contribute equally to models. And encoding was used to convert categorical labels into numerical form.

- Feature Importance Analysis: Conducted to identify the most impactful features using techniques like Recursive Feature Elimination (RFE).

- Feature Selection: Based on the feature importance analysis, a subset of most important features was chosen to enhance model efficiency and accuracy.

These preprocessing steps were critical for ensuring the models were trained on high-quality, balanced, and meaningful data, leading to robust performance. We will detail more on these steps in the “Experiments” section of the report.

5. Methods

As indicated in the introduction, deepfakes in audio are often used and considered a significant threat. So, the challenge is finding the best approach to detect them. This section will mention the different models we used to classify deepfake audio, which were selected based on their applicability to our problem.

Before starting the model, we must prepare the data using feature engineering and dimensional reduction. As we know, feature engineering is a pivotal step in preparing audio data for modeling, so we need to begin with feature extraction, where we derived a range of different measures since every characteristic played a distinct role in defining the sound, encompassing frequency distribution, tonal content, and signal energy. Moreover, noise cancellation techniques must be applied to improve the audio signals, as this step reduces background noise, which could enhance the input data from the model to have proper pattern learning. Additionally, audio signals must be resampled to ensure uniformity for input data across both labels, as consistent sampling rates are critical in effectively training the model. Furthermore, dimensionality reduction techniques, especially Principal Component Analysis (PCA), were utilized to improve model performance and decrease computational complexity since they detect the most influential features while removing irrelevant features from training, thus enhancing model interpretability and helping prevent overfitting for model efficiency [15].

All these techniques help us build and train our models effectively. As for the models, we suggested using Support vector machines (SVMs), Extreme Gradient Boosting (XGBoost), and Convolutional Neural Networks (CNNs) algorithms.

- **SVM**

We start with a base and a simple model to examine the model's ability to define the patterns in audio. Even though it’s a base classifier, SVM is a robust classification algorithm for our audio dataset since it can deal with high-dimensional spaces. Through kernel functions, it is approachable to both linear and non-linear classification problems, which will help us deal with audio. Furthermore, SVM is well known for its excellent generalization, which is critical to separating subtle differences between real and fake audio [16]. Lastly, SVM is known to perform well with small datasets, which is our case.

- **XGBoost**

For a more complicated model, we turned to XGBoost, which incorporates fast and efficient gradient-boosting decision trees and is built to enhance machine learning model performance and computational speed. This is necessary for our problem, as we extract many features that take a long computational time. We have chosen XGBoost for its robustness against overfitting and properties that are important for noisy and inconsistent audio data [17].

- **CNNs**

Finally, we employed Convolutional Neural Networks (CNNs), a well-known algorithm for handling audio data. It can automatically learn hierarchical features from raw audio transformed into spectrograms. CNN also capture patterns that may not be distinguishable by other models [18]. This is crucial for differentiating between real and fake audio, where minor differences can significantly impact decisions.

In conclusion, combining SVM, XGBoost, and CNN with feature engineering is expected to provide a good performance for Audio DeepFake detection, where each model was selected to address specific issues related to our problem.

6. Experiments

In this section, we will explain the steps taken to train and evaluate the model for audio deepfake detection.

In this project, we aimed to accurately detect deepfake audio by developing a reliable model. Starting by choosing appropriate models, collecting and processing the dataset, feature extraction and selection, hyperparameter tuning, and concluding with assessing model performance. During these steps, we evaluated three different models: Support Vector Machine (SVM), XGBoost, and Convolutional Neural Network (CNN). Each one was selected for their capabilities to achieve the best performance.

After choosing the dataset, we focused on data preparation and feature engineering. Audio data naturally contains noise and variations that can obscure important patterns needed for the model. To address this, we import **noisereduce** library [19] to eliminate background noise, generating cleaner and clearer data for feature extraction. Next, we used **librosa** [20] to extract the most contributed features, as detailed in the methods section.

Features Selection was the next step to improve the model's performance. By utilizing **RFE** [21] with an SVM model, we selected the top 15 features with the greatest predictability. This action decreased dimensionality and avoided overfitting, all while preserving the most important features of the data, as detailed in the Methods section. Moreover, we standardized the features by removing the mean and scaling to unit variance using the **StandardScale** library [22]. This step was essential for SVM, XGBoost, and CNN models, as they are affected by the input data's scale.

One significant problem we faced in the dataset was the imbalance between the quantities of the classes "REAL" and "FAKE" audio with 8 REAL audio and 56 FAKE audio, the class labels are not equally represented with a much larger number of "FAKE" files compared to "REAL". This imbalance caused a bias with "FAKE" audios. To address this issue, we resampled the dataset by importing **Synthetic Minority Oversampling Technique (SMOTE)** [23]**,** a method from **imblearn** library [24], SMOTE works as a combination of over-sampling the minority class and under-sampling the majority class to achieve better classifier performance. Therefore, SMOTE will generate fake instances for the minority class ("REAL") by creating additional data points. This ensured that the dataset was evenly balanced before separated into training, validation, and testing sets which may lead to unbiased classification. SMOTE improved the model's ability to identify patterns by balancing the representation of both classes, resulting in reduced bias and enhanced classification performance.

Model Development

As mentioned in the method section, we will experiment with three complex models to deal with our data.

The selection of these sophisticated models can help uncover hidden, complex patterns and interactions within the dataset, improving accuracy and recall. Before discussing each model individually, we applied a consistent methodology for splitting the dataset into training, testing, and validation sets for SVM and XGBoost models. This approach allows for a fair comparison between models. We utilized 5-fold cross-validation, which divides the data into five subsets: four for training and one for validation, this technique is repeated in each iteration. Cross-validation was selected to reduce overfitting and enhance generalization, ensuring that our performance metrics accurately reflect the models' ability to perform well on unseen data. The choice of five-fold is based on the fact that 5-fold cross-validation is a widely accepted standard in machine learning, providing a balance between computational efficiency and accuracy in validation. For the CNN model, it was complex to use 5-fold cross-validation. Therefore, we decided on a 70-30 train-test split, which reduces training time while ensuring thorough data coverage. Hyperparameter tuning varied for each model, leading to a comprehensive evaluation of their performance.

- **SVM Model**

The initial model utilized was SVM, which stands for Support Vector Machine. SVM was the starting point because it uses a hyperplane to separate data into two categories, which is straightforward and effective for simpler patterns. However, audio data often contains complex, non-linear relationships between features, which limits the SVM. Moreover, to ensure optimal performance, we imported **GridSearchCV** [25] for hyperparameter tuning, allowing us to systematically evaluate all the hyperparameter values to maximize the model performance. SVM parameters include the weights and biases that define the decision boundary and optimize the model.

The SVM model's hyperparameters are the kernel's type and parameters and the regularization (C). Using grid search, we explored combinations of kernel: [linear, rbf, poly] representing the common kernel types, C: [0.1, 1, 10, 100] values up to 100 to avoid restriction and aiming for a higher margin, gamma: [scale, auto], adaptive settings for the influence of training examples, degree: [3, 4, 5], to prevents excessive computational complexity.

The optimal combination of kernel = rbf, C = 0.1, and gamma = 'scale' (with no value for the degree, as it is used only in the poly kernel) yielded the highest accuracy on the validation set during the 5-fold cross-validation process. Therefore, we fitted the SVM model using these best hyperparameters and got the promising results, but they were insufficient for dealing with the complexity of deepfake audio. This limitation highlighted the need to explore more advanced models, which led us to XGBoost.

- **XGBoost Model**

XGBoost is a strong ensemble algorithm, particularly a boosting one. It has the capability to manage complex datasets and enhance predictions using **gradient boosting**. XGBoost was chosen due to its ability to build decision trees sequentially, enabling it to effectively capture complex patterns compared to SVM. Additionally, the feature importance function helped us identify key audio characteristics, such as MFCC and spectral features, allowing the model to focus on the most relevant data. Starting by feature selection to choose the highest 15 features, resulting several MFCC features, Chroma-based features, and Crossing rate, that have high contribution of the correct classification task.

Similar to SVM, the parameters of XGBoost include the learned weights for each feature and the bias term that adjusts the output, which together help achieve accurate predictions. Furthermore, to achieve the best performance, it was crucial to finely adjust important hyperparameters using **GridSearchCV** [25] to systematically explore all options for each hyperparameter, including learning rate: [0.01, 0.05, 0.1] to test the trade-off between learning speed and model performance, maximum depth: [3, 5, 7] to control tree complexity, minimum child weight: [1, 5, 10] with lower values to allow more leaves and higher values helping to prevent overfitting, subsample: [0.6, 0.8, 1.0] that test the amount of samples utilized for each tree; lower values reduce overfitting through randomness, column sample by tree: [0.6, 0.8, 1.0] to introduce diversity, number of estimators: [50, 100, 150] with more estimators potentially improving accuracy, L1 regularization: [0.1, 0.5, 1] and L2 regularization: [1, 2, 5] to assess their impact on generalization and model complexity. Best hyperparameters was learning rate of 0.1, max_depth of 3 to preventing overfitting and capturing enough patterns min_child_weight of 1 which allow more leaves, subsample of 0.6 to strike a balance between randomness and model reliability, colsample_bytree of 0.8 to prevent the overfitting, number of boosting rounds, defined by n_estimators with 150 iterations were sufficient for convergence, reg_alpha of 0.1 and reg_lambda of 2 which introduces moderate sparsity and stronger penalty to enhance generalization. All these values were selected for the final model to optimize performance. However, it performed well but overall worse than SVM, meaning it struggled to manage the dataset's complexity. This can be interpreted due to SVM's ability to achieve better with smaller datasets, which is our case, or because our data still needs a more complex model to capture the relationships and patterns in it. To test these hypotheses, we will build the next final model, which is CNN.

- **CNN Model**

Finally, we employed CNN (Convolutional Neural Network) that is well-suited for analyzing complex data patterns, especially in audio. CNN uses convolutional layers, a pooling layer user to extract features that capture information for both the frequency and time aspects in the audio data, and a fully connected layer for classification. Our model consists of two convolutional and max-pooling layers, a dense layer, and an output layer. We converted the features into a 4D format to make sure they work with this CNN structure. Moreover, CNN processes raw input features hierarchically; it learns and detects patterns from low features to high abstract representations. Thus, we don't need to apply a feature selection.

The parameters of a CNN include the learned weights and biases associated with each convolutional and fully connected layer, and to enhance the model, we utilized **Keras Tuner RandomSearch** [26] for hyperparameter tuning instead of GridSearch because of the complexity of CNN. Exploring different **numbers of filters** in each convolutional layer, which impacts the model's feature extraction capability, between [32, 64, 96, 128], dropout rates: [0.2, 0.3, 0.4, 0.5] were tested to prevent overfitting by randomly dropping the neurons during training. The dense layer units: [64, 128, 192, 256] to find a balance between model complexity and performance. Finally, the learning rate, which controls the step size during gradient updates, was tested with values [0.01, 0.001, 0.0001]. Training the CNN model took a longer period and demanded larger computational resources in contrast to SVM and XGBoost, yet the outcomes were remarkable. Resulting these best hyperparameters: conv_1_filters: 32 to capture basic features and prevent overfitting, conv_2_filters: 64 for learning complex patterns, dense_units: 64 for a balance between capacity and generalization; dropout_1: 0.3, dropout_3: 0.3, and dropout_2: 0.4 to reduce overfitting and additional regularization, a learning_rate of 0.01 for stable convergence.

Metrics are used to measure the effectiveness of classification models, including accuracy, recall, and learning curves were used to evaluate the models' performance. During training, validation accuracy was observed to identify any signs of overfitting by measuring how the model correctly predicts the outcome and ensuring that the model generalizes well to unseen data rather than memorizing the training set. The weighted recall was used to ensure that both "REAL" and "FAKE" audio samples were correctly identified, minimizing the risk of false negatives in deepfake detection.

An important assessment was conducted by testing the final deployed CNN model on a small, hand-selected dataset consisting of five audio samples (three REAL and two FAKE). The model accurately labeled four samples, showcasing its capacity to generalize. The single misclassification occurred with a very convincing deepfake, underscoring the difficulty in identifying very realistic alterations. The learning curve was explained in detail in the results section.

Additional libraries were used for implementation, including **TensorFlow** [27] and **Keras** [28], which used to build and train the CNN model, **scikit-learn** [29] facilitated feature selection, preprocessing, and implementation of SVM and XGBoost.

We used Google Colab with GPU acceleration (Tesla T4) to handle the computational demands of training the models, especially CNN. While lightweight tasks, such as preprocessing and SVM training, were executed on the CPU

7. Results and Discussion

This section will present the results of the implemented models and their performance, we will analyze and interpret the results and discuss key insights gained from the analysis.

In this project, we trained, evaluated, and compared the performance of the three machine learning models we implemented, which were mentioned before: SVM, XGBoost, and CNN. The objective was to determine the most effective model for our problem by analyzing key performance metrics, specifically **accuracy, recall, and learning curve**. These metrics provide insight into how well each model correctly predicts the outcomes (accuracy), identifies true positives (recall), which is especially crucial for applications where false negatives must be minimized, and that is what we need, and show the three models performance as the training set size increases (learning curve).

The results of the models are summarized as illustrated in Table 3. By comparing these metrics, we aim to obtain conclusions about their effectiveness and suitability for our task.

Table 3: Models results

Metrics	SVM	XGBoost	CNN
Accuracy	87.56%	84.17%	97.06%
Recall	87.56%	85.32%	97.06%

For our base model, SVM performs slightly better than XGBoost in both accuracy and recall; it has an accuracy and recall of 87.56%, while XGBoost has an accuracy of 84.17% and a recall of 85.32%. This means SVM is better at correctly classifying data and identifying positive cases in this dataset. The results from both models are close and both models performed well and provided strong predictive capabilities. Although SVM slightly outperforms better than XGBoost, the difference in performance is minor, and both models are effective for this dataset. The small gap in performance may be due to the SVM is generally better with small datasets, which is our case, or that our dataset needs a more complex model to uncover complicated relationships and patterns in our dataset, while XGBoost generally performs better on larger datasets. Overall, both models demonstrate good results and can be considered reliable for this task.

On the other hand, the CNN model significantly outperforms both SVM and XGBoost. Its accuracy and recall are both around 97%, which is higher than SVM and XGBoost. Since CNN is better at finding complex patterns in data, it helps work well with audio and high-dimensional data.

The fact that accuracy and recall are both very high and close shows that the CNN model correctly identifies positive and negative examples without being biased. It has a good balance, predicting positives accurately and not missing important cases.

Compared to SVM and XGBoost, CNN is much better at understanding and classifying audio as real or fake. This makes it the most effective model for our problem and dataset, thus will be used as the deployed model. However, it’s worth mentioning that all three models performed well and proved to be able to generalize well to new and unseen data, which was measured by the testing/validation accuracy, which we will detail next, using learning curves. As we use kinds of L1 and L2 regularization techniques.

Learning curves are another performance metric that is plotted to assess how well the model learns over increasing amounts of training data and to identify potential overfitting or underfitting; we will compare the models using this metric. In the graphs below (as shown in Figures 5, 6, and 7), the blue line represents the training score, and the orange line represents the cross-validation score, where the x-axis represents the training set size and y-axis represent the accuracy of each model.

For SVM learning curve (as shown in Figure 5), we observe that initially, the training score is lower, suggesting that the model is less effective when the dataset is very small. As the training set size increases, the training score (accuracy) improves, showing that the model becomes more effective with more data. The cross-validation score remains stable, indicating that SVM generalizes well across different dataset sizes but not for extremely small ones. Towards the end of the curve, as the two lines (training and cross-validation scores) converge, it signals that the model is well-fitted without overfitting or underfitting. SVM shows solid performance, especially when more data is provided, making it a good choice for smaller datasets.

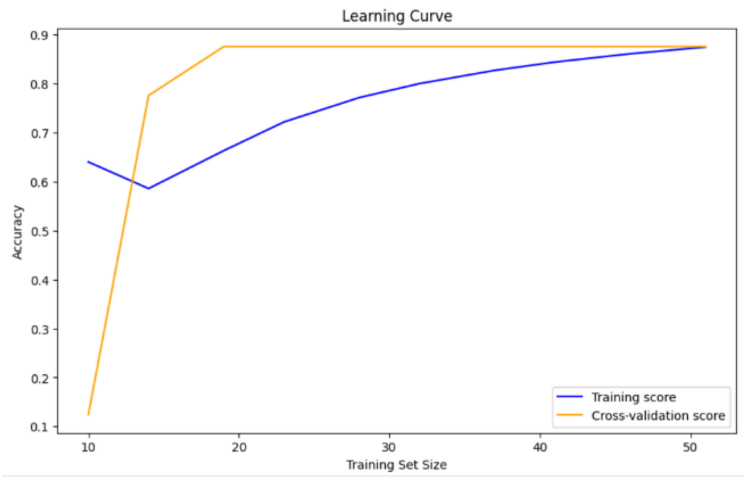

Figure 5: SVM's learning curve

As shown in Figure 6, the XGBoost learning curve indicates that the training accuracy increases as the dataset grows, eventually reaching around 0.95. The cross-validation accuracy starts at a high point, then lowers slightly at intermediate training sizes before increasing again as the dataset size grows. This pattern indicates that XGBoost benefits from more data, but its performance is not as stable as SVM's, showing some fluctuations. The difference in performance suggests that XGBoost may struggle with smaller datasets or require additional fine-tuning for better results.

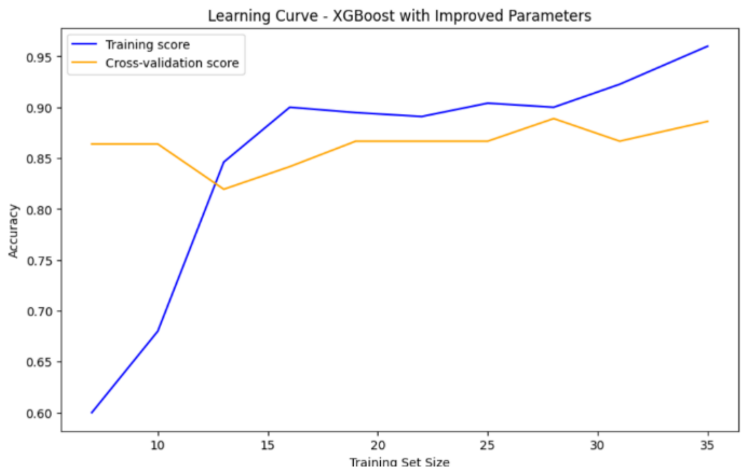

Figure 6: XGBoost's learning curve

Figure 7 illustrates the CNN learning curve across 20 epochs, demonstrating a quick increase in validation accuracy. This suggests that CNN generalizes unseen data well at the start of training. The point of similarity of training and validation accuracy is stabilized from epoch five onwards, indicating that the model has learned the patterns in the data and is not overfitting. However, there are slight fluctuations in validation accuracy, suggesting that we may further tune regularization to enhance stability across epochs, which is expected since we did a random search.

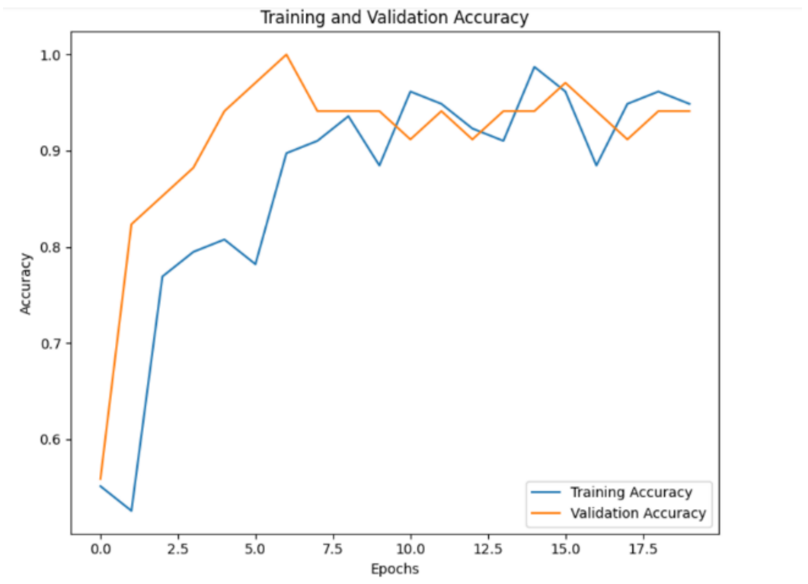

Figure 7: CNN's learning curve

Interesting is how the Spectral Centroid emerged as the most contributing feature in both the SVM and XGBoost models. This indicates the role of frequency distribution in distinguishing real from fake audio. The sharp transitions and higher frequency peaks in fake audios suggest that deepfake generation methods fail to replicate the smoothness and natural variation of real audio. Furthermore, the project showed that while SVM performed well with smaller datasets, the CNN model performed better with larger feature sets and complex patterns, achieving a test accuracy of 97.06%. This again underlines the potential of CNNs in detecting subtle characteristics in audio that may not be captured by traditional simpler models.

The fact that all fake samples were correctly classified during the live testing would suggest that synthetic audio possesses consistent characteristics that are detectable even in previously unseen data.

This insight can help in the enhancement of detection algorithms and audio communication security in the future.

However, CNN is the most effective model, particularly for our dataset. It outperforms both SVM and XGBoost regarding fast learning and high accuracy. Our dataset is complex and high-dimensional, and CNN can handle these characteristics, giving it a clear advantage over SVM and XGBoost.

8. Conclusion

To conclude, the SoundShield project addresses the critical issue of Audio DeepFakes, which presents serious risks in various areas, including misinformation and fraud. In this project, we started by outlining our motivation, selecting our dataset, performing background research on key terms related to our project, and literature review on related works to enrich our understanding of the field. We analyzed and described the dataset to get a better grasp of its nature and size. We then performed data preprocessing and preparation which included feature extraction, noise cancellation, and resampling. Next, We proposed a solution using three ML algorithms to classify audio files to “REAL” or “FAKE”. For that, we suggested using Support vector machine (SVM), Extreme Gradient Boosting (XGBoost), and Convolutional Neural Network (CNN) algorithms. We utilized many techniques like feature importance, feature selection, hyperparameter tuning, model training, and testing for each model. We compared the performance of these algorithms in terms of their accuracy and recall in which we figured that the best model that will be used for deployment is CNN, with accuracy and recall of 97%. By developing the CNN model, we aim to improve the reliability of digital audio communications, as the model provides promising accuracy and recall during testing.

Throughout this project, we have encountered various challenges due to the high dimensionality of our data. One major challenge was selecting the algorithm that suits our data, as we need a complicated model that detects all the patterns since the complicity of audio features is an obstacle to interpreting the model's decision process. Additionally, as we need complex models and techniques to detect deepfakes, this often comes with the drawback of long computational time. Lastly, reducing the overfitting in our models is considered one of the challenges since we need to do hyperparameter tuning and data augmentation to avoid it.

Future work could include exploring another dataset to apply these algorithms and then comparing the used algorithm's performance to validate them against deepfake. Another interesting future work could be improving these algorithms by performing more techniques like feature engineering, trying other hyperparameter tuning techniques, and gathering more data. Lastly, this work could contribute to further studying of more complicated models, such as different neural network models. Overall, the SoundShield project will have a sufficient effect in audio communications, contributing to safeguarding the integrity of digital audio communications.

9. References

[1] A. Chadha, V. Kumar, S. Kashyap, and M. Gupta, “Deepfake: An Overview,” *Lecture Notes in Networks and Systems*, vol. 203 LNNS, pp. 557–566, 2021, doi: 10.1007/978-981-16-0733-2_39.

[2] J. J. Bird and A. Lotfi, “DEEP-VOICE: DeepFake Voice Recognition.” Accessed: Sep. 09, 2024. [Online]. Available: https://www.kaggle.com/datasets/birdy654/deep-voice-deepfake-voice-recognition

[3] S. ELGAMMUDI, “Deep Fakes Generation Using LSTM based Generative Adversarial Networks ,” 2021.

[4] Gnani Marketing, “The Advantages and Disadvantages of Voice Authentication in 2022.” Accessed: Sep. 19, 2024. [Online]. Available: https://www.gnani.ai/resources/blogs/the-advantages-and-disadvantages-of-voice-biometrics-in-2022/

[5] E. Carrero, “Voice deepfake: Is it possible to detect a fake voice? - Mobbeel.” Accessed: Sep. 19, 2024. [Online]. Available: https://www.mobbeel.com/en/blog/voice-deepfake/

[6] V. S. A. Srinagesh, “Efficient Deepfake Audio Detection Using Spectro-Temporal Analysis and Deep Learning,” *Journal of Electrical Systems*, vol. 20, no. 5s, pp. 10–18, Apr. 2024, doi: 10.52783/JES.1829.

[7] A. Pianese, D. Cozzolino, G. Poggi, and L. Verdoliva, “Training-Free Deepfake Voice Recognition by Leveraging Large-Scale Pre-Trained Models”, Accessed: Sep. 19, 2024. [Online]. Available: https://www.wsj.com/articles/fraudsters-use-ai-to-mimic-ceos-voice-in-

[8] S. Y. Lim, D. K. Chae, and S. C. Lee, “Detecting Deepfake Voice Using Explainable Deep Learning Techniques,” *Applied Sciences 2022, Vol. 12, Page 3926*, vol. 12, no. 8, p. 3926, Apr. 2022, doi: 10.3390/APP12083926.

[9] A. Hamza *et al.*, “Deepfake Audio Detection via MFCC features using Machine Learning,” *IEEE Access*, 2022, doi: 10.1109/ACCESS.2022.3231480.

[10] N. V. Kulangareth, J. Kaufman, J. Oreskovic, and Y. Fossat, “Investigation of Deepfake Voice Detection Using Speech Pause Patterns: Algorithm Development and Validation.,” *JMIR Biomed Eng*, vol. 9, no. 1, p. e56245, Mar. 2024, doi: 10.2196/56245.

[11] “RVC Voice Changer - voice.ai.” Accessed: Sep. 10, 2024. [Online]. Available: https://voice.ai/hub/tools/rvc-voice-changer/

[12] “RVC Vocal Models: Revolutionizing Voice Technology | Speechify.” Accessed: Sep. 10, 2024. [Online]. Available: https://speechify.com/blog/rvc-vocal-models/

[13] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, “Spleeter: a fast and efficient music source separation tool with pre-trained models,” *J Open Source Softw*, vol. 5, no. 50, p. 2154, Jun. 2020, doi: 10.21105/JOSS.02154.

[14] J. J. Bird and A. Lotfi, “REAL-TIME DETECTION OF AI-GENERATED SPEECH FOR DEEPFAKE VOICE CONVERSION”, Accessed: Sep. 10, 2024. [Online]. Available: https://www.kaggle.com/datasets/birdy654/deep-voice-deepfake-voice-recognition

[15] “Reducing Dataset Size Without Losing Model Performance | Encord.” Accessed: Nov. 21, 2024. [Online]. Available: https://encord.com/blog/dimentionality-reduction-techniques-machine-learning/

[16] “What Is Support Vector Machine? | IBM.” Accessed: Nov. 21, 2024. [Online]. Available: https://www.ibm.com/topics/support-vector-machine

[17] “XGBoost – What Is It and Why Does It Matter?” Accessed: Nov. 21, 2024. [Online]. Available: https://www.nvidia.com/en-us/glossary/xgboost/

[18] L. Nanni, Y. M. G. Costa, R. L. Aguiar, R. B. Mangolin, S. Brahnam, and C. N. Silla, “Ensemble of convolutional neural networks to improve animal audio classification,” *EURASIP J Audio Speech Music Process*, vol. 2020, no. 1, Dec. 2020, doi: 10.1186/S13636-020-00175-3.

[19] T. Sainburg and T. Q. Gentner, “noisereduce library,” *Front Behav Neurosci*, vol. 15, Dec. 2021, doi: 10.3389/FNBEH.2021.811737/FULL.

[20] “librosa — librosa 0.10.2 documentation.” Accessed: Nov. 21, 2024. [Online]. Available: https://librosa.org/doc/latest/index.html

[21] “RFE — scikit-learn 1.7.dev0 documentation.” Accessed: Nov. 21, 2024. [Online]. Available: https://scikit-learn.org/dev/modules/generated/sklearn.feature_selection.RFE.html

[22] “StandardScaler — scikit-learn 1.7.dev0 documentation.” Accessed: Nov. 21, 2024. [Online]. Available: https://scikit-learn.org/dev/modules/generated/sklearn.preprocessing.StandardScaler.html

[23] “SMOTE — Version 0.12.4.” Accessed: Nov. 21, 2024. [Online]. Available: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html

[24] “imbalanced-learn documentation — Version 0.12.4.” Accessed: Nov. 21, 2024. [Online]. Available: https://imbalanced-learn.org/stable/

[25] “GridSearchCV — scikit-learn 1.5.2 documentation.” Accessed: Nov. 21, 2024. [Online]. Available: https://scikit-learn.org/1.5/modules/generated/sklearn.model_selection.GridSearchCV.html

[26] “RandomSearch Tuner.” Accessed: Nov. 21, 2024. [Online]. Available: https://keras.io/api/keras_tuner/tuners/random/

[27] “TensorFlow.” Accessed: Nov. 21, 2024. [Online]. Available: https://www.tensorflow.org/

[28] “Keras 3 API documentation.” Accessed: Nov. 21, 2024. [Online]. Available: https://keras.io/api/

[29] “scikit-learn: machine learning in Python — scikit-learn 1.5.2 documentation.” Accessed: Nov. 21, 2024. [Online]. Available: https://scikit-learn.org/stable/