The Islamia University of Bahawalpur Pakistan

**Department of Information Technology**

**ADINFT (1M)**

Software Requirements Specification

# AI-Based Deepfake Audio Detection

**Semester 7th (M1)**

**Session** 24-26

**Supervisor:**

Ms. Sara Fareed

**Project Team Members:**

Quriul Ain (F24BINFT7M04009)

## Table of Contents

## Abstract

The deepfake technologies are evolving at an extremely high rate threatening the privacy of the e-mail channels significantly. The deepfake audio may actually today be considered one of the most threatening considering that such a technology should be called the fulfilling, the true to life manner of recreating the voice, the tone, and the speech pattern of the people in the walls in which it is comparable. The use of such fake audio by the applicant as a wicked screening method as well as financial offenses, deceiving and misleading information could be applied to an evil context.

The provided project in its effort to avoid the potential encroachment helps in an embodiment of the machine learning that offers the determination of the deep fake audio files. This system will be founded on the strategy of achieving a fine spectral and temporal sense change of signal in acoustic features grounded on the method of isolating the acoustic features of recordings provided in **Mel frequency Cepstral Coefficients (MFCC)** that indicated that it was an effective method in the investigation of emotion in speech. This is further attained through the leaning in these properties through relying upon a **Random Forest Classifier**, which is the ensemble model which has shown to be efficient and powerful. The experimental task is high, and this means that there is relevance about the given approach and hence it will be an effective approach of avoiding the possible manipulation of synthetic audio.

# 1. Introduction

## 1.1 Background

Due to AI, voice synthesis technologies have become advanced. The equipment can now also create the speech that can be heard as being compared to a human being however they can create those voices that can be heard like the real sound. Despite the various veracious applications of this, voice assistants and commercial linguistic dubbing, it presents its capabilities to ill-intended applications, as well. One method of deepfaking voice recording is by using financial fraud and misinformation and by social engineering attack.

## 1.2 Problem Statement

The truthfulness of an audio sample demonstrates that once a natural or artistically re-synthesized sample is no longer so direct in the phase of the voice synthesis development in AI. The introduction about the legitimacy of the authentic speech or the deep fake audios is a hoax as a result of the fact that the former verification mechanisms does not present an amenious gap, a serious concern of privacy of the Internet. To overcome the latter, one must accept an implementation of the machine-learned based-detection system based on the number of algorithms, which will understand the presence of synthetic voice and eliminate fraud, impersonation, and forged information.

## 1.3 Objectives

- Generate a list of real and counterfeit audios
- Obtain feature through MFCCS.
- Using progressing bars (tqdm).
- Random forest classifier design, capable of attaining the feature of distinguishing between an authentic voice and an accomplished admit.
- Secure that there is a fair consideration which shall be realized, by the separation of training and examination element in the data compilation.
- Conduct Performance experimental analysis on the trained model using the accuracy classification and confusion matrices.
- Create a system that is able to determine if unseen audio samples are real or fake.

# 2. Literature Review

The deep speech technologies have seen an emergence of one of the attractive fields of research as the deep fake audio detecting procedure. Most of the techniques applied in the classical audio forensics that relied on statistics as the parameter of the alteration of speech anomalies have been unable to stop the extremely lifelike synthesized voice (Yi et al., 2023). Mel frequency Cepstral Coefficients (MFCCs) find extensive popularity on the basis of feature identification, i.e. spectral and time-related construction that is inherent in original and counterfeit audio (Zhang et al., 2025). Designs based on machine learning approximating, which include Special Virtual Machine, Deep Learning, and even the random trees, have been performing successfully in reference to the detecting process, however, there has been still the issue of high-quality man-made voices (Choi, Kim and Choi, 2025; Zhang et al., 2024).

The features which are implemented using MFCC are implemented in the present project with the consideration that Rand Forest-based classifier with both efficiency and performance advantage which is beneficial in detecting deepfake in audio makes it convenient to the end users.

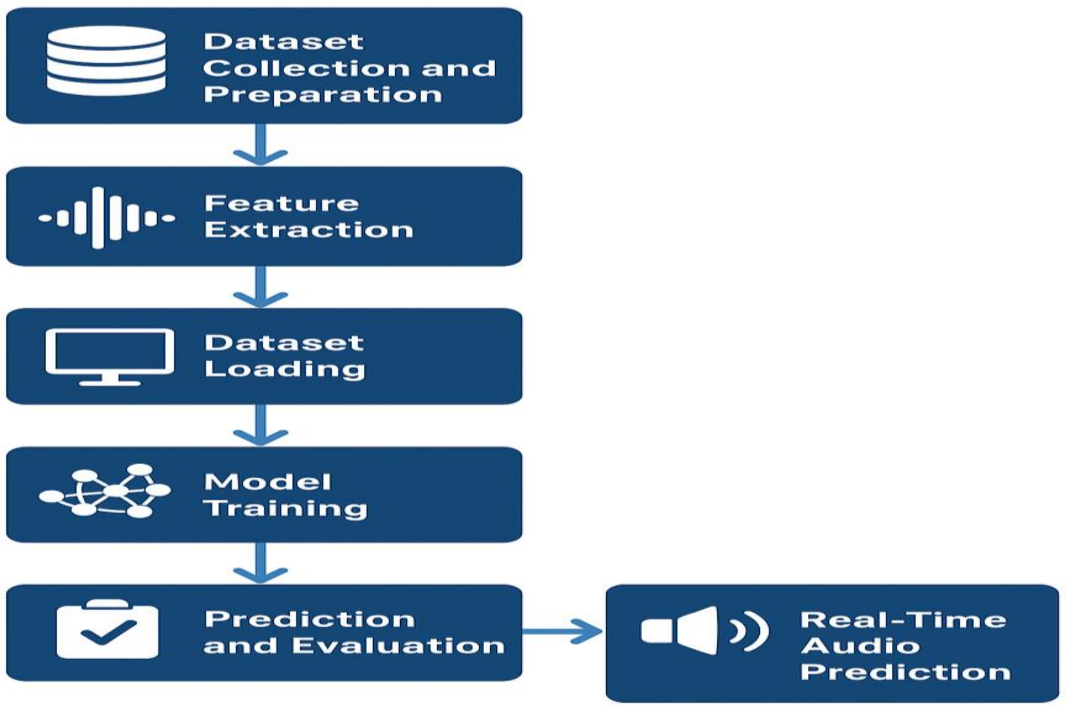
## 2.1 References

Yi, J., Wang, C., Tao, J., Zhang, X., Zhang, C.Y. & Zhao, Y. (2023) Audio deepfake detection: A survey. arXiv preprint arXiv:2308.14970. Available at: https://arxiv.org/abs/2308.14970

Choi, J., Kim, T. & Choi, J. (2025) ID-Flow: Leveraging voice identity for generalizing audio deepfake detection. IEEE International Conference on Consumer Electronics (ICCE). Available at: https://ieeexplore.ieee.org/document/10077834

Zhang, Y., Jiang, F., Duan, Z., Choi, J., & Kim, T. (2024) *'ID-Flow: Leveraging Voice Identity for Generalizing Audio Deepfake Detection'*. IEEE Signal Processing Letters, 28, pp. 937–941. Available at: https://ieeexplore.ieee.org/document/9417604

# 3. Methodology


Dataset Collection and Preparation
Feature Extraction
Dataset Loading
Model Training
Prediction and Evaluation
Real-Time Audio Prediction

*Methodology of AI-based Deepfake Audio Detection System*

## 3.1 Dataset Collection and Preparation

The datasets which is used in the project contains **real and AI-generated voices**, which are collected and organized from multiple sources.

The sources, other samples, organization and specification of the dataset used in this project are discussed in the following sections.

### 3.1.1 Sources of Dataset

The dataset was downloaded from the website: https://github.com/dstsmallbird/SiF-DeepVC_Dataset. It contains both real and synthetic voice recordings.

### 3.1.2 Additional Audio Samples

Besides that, even realistic and synthetic voices were modelled with hands to provide more variation and better stability of the models. Each sample was placed in the appropriate folder based on its type **(real or fake).**

### 3.1.3 Organization of Dataset for This Project

For simplicity of processing and labeling, all audio files were reorganized into two folders which are:

- **Real:** Which is made of original and processed human notes of the downloaded data set, and other real audio notes produced by hand.
- **Fake:** This includes all the AI recognized deepfakes instances of the downloaded information along with all other instances of fake audio created manually.

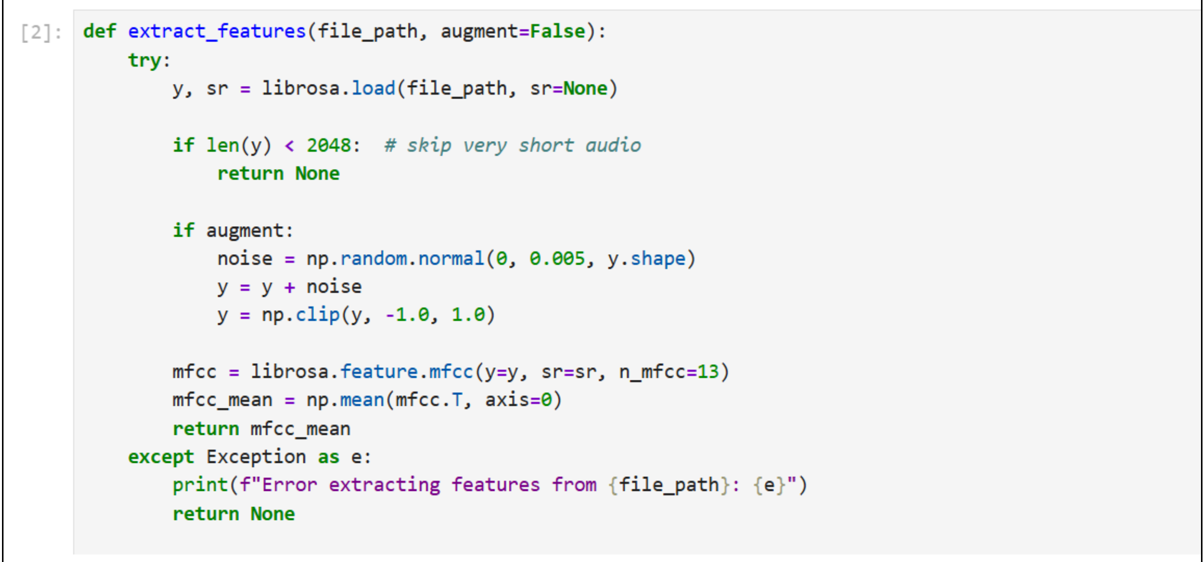
### 3.1.4 Audio Specifications

- **The supported formats:** .mp3, .wav, .flac and.wav.noisered.
- **Voice sample duration:** 2048 or longer (audio samples less than this are outlived to provide enough sample length to extract features, e.g. 0.128 seconds at 16 kHz sampling rate)
- **Feature extraction:** 13 MFCC features per audio file (standard of speech analysis)
- **Labels**: The samples are marked with 0 for real voices and 1 for fake voices to make them easy to train and use in evaluation of the model.

## 3.2 Feature Extraction

MFCC (Mel-Frequency Cepstral Coefficients) are extracted from each audio sample. The steps followed are:

- **Features extraction:** each audio is extracted in to thirteen (13) MFCC features.
- **Data augmentation (optimal):** It is possible to simply add some variety to the data by applying Gaussian noise of standard deviation.
- **Minimum length test:** the files which comprise of length of less than 2048 samples are left untouched and they are accidentally deleted. The formats
- **Supported** extensions will be of .mp3, .wav .flac and .wav.noisended.

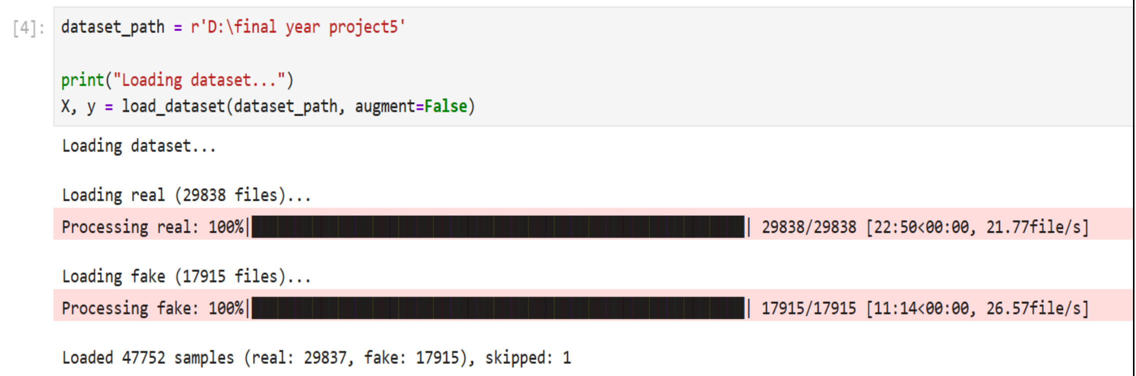


```
[2]: def extract_features(file_path, augment=False):
         try:
             y, sr = librosa.load(file_path, sr=None)

             if len(y) < 2048:  # skip very short audio
                 return None

             if augment:
                 noise = np.random.normal(0, 0.005, y.shape)
                 y = y + noise
                 y = np.clip(y, -1.0, 1.0)

             mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13)
             mfcc_mean = np.mean(mfcc.T, axis=0)
             return mfcc_mean
         except Exception as e:
             print(f"Error extracting features from {file_path}: {e}")
             return None
```

Python implementation of MFCC feature extraction for deepfake audio detection

## 3.3 Dataset Loading

The feature are extracted from the dataset. And then available for training. This includes absorbing all audio files, getting MFCC features, classifying all the samples as marking as fake or real, and making sure that only valid files are used.



```
[4]: dataset_path = r'D:\final year project5'

     print("Loading dataset...")
     X, y = load_dataset(dataset_path, augment=False)

     Loading dataset...

     Loading real (29838 files)...
     Processing real: 100%|██████████████████████████████| 29838/29838 [22:50<00:00, 21.77file/s]

     Loading fake (17915 files)...
     Processing fake: 100%|██████████████████████████████| 17915/17915 [11:14<00:00, 26.57file/s]

     Loaded 47752 samples (real: 29837, fake: 17915), skipped: 1
```

Python implementation of Dataset Loading

### 3.3.1 Dataset Summary

- **Total audio files processed:** 47,752
- **Real samples:** 29,838
- **Fake samples:** 17,915

- **Skipped files:** 1 (due to insufficient duration)
- **Processing time:**
    - Real folder: 22 minutes 50 seconds
    - Fake folder: 11 minutes 14 seconds

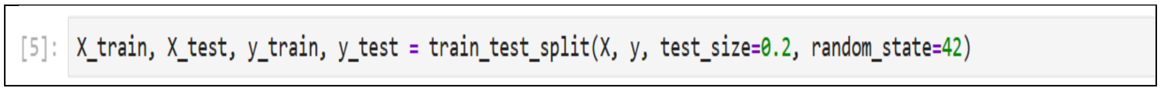
The process of loading datasets that ensures that all the **valid audio samples** have been properly prepared and their progress monitored and tags final rather than being trained and tested on the machine learning system.

## 3.4 Model Training

The model training algorithm that was used was the **Random Forest Classifier** since it is a solid algorithm that is capable of working with high-dimensional features space. To make the dataset reproducible, **80 percent** was used as the **training** and **20 percent** as a **testing** set with fixed random state (42).

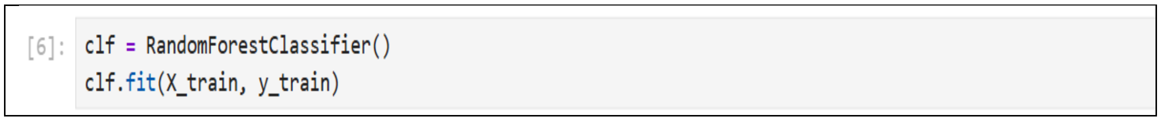
The training process involved the following steps:

1. **Data Splitting**



```
[5]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

2. **Model Initialization and Training**

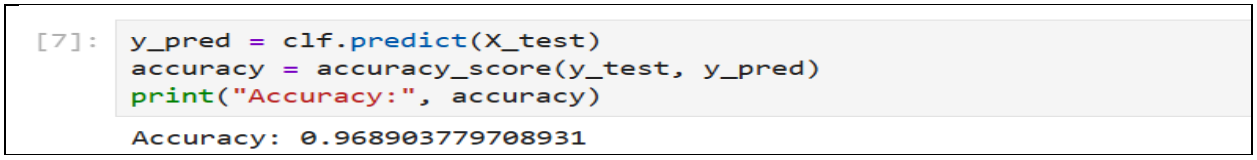


```
[6]: clf = RandomForestClassifier()
     clf.fit(X_train, y_train)
```

It ensured that the random forest classifier was trained on the dataset while reserving a portion for performance evaluation.

## 3.5 Prediction and Evaluation

The model was tested after being trained on the test set. The trained prediction of random forest classifier came up with :



```
[7]: y_pred = clf.predict(X_test)
     accuracy = accuracy_score(y_test, y_pred)
     print("Accuracy:", accuracy)

     Accuracy: 0.968903779708931
```

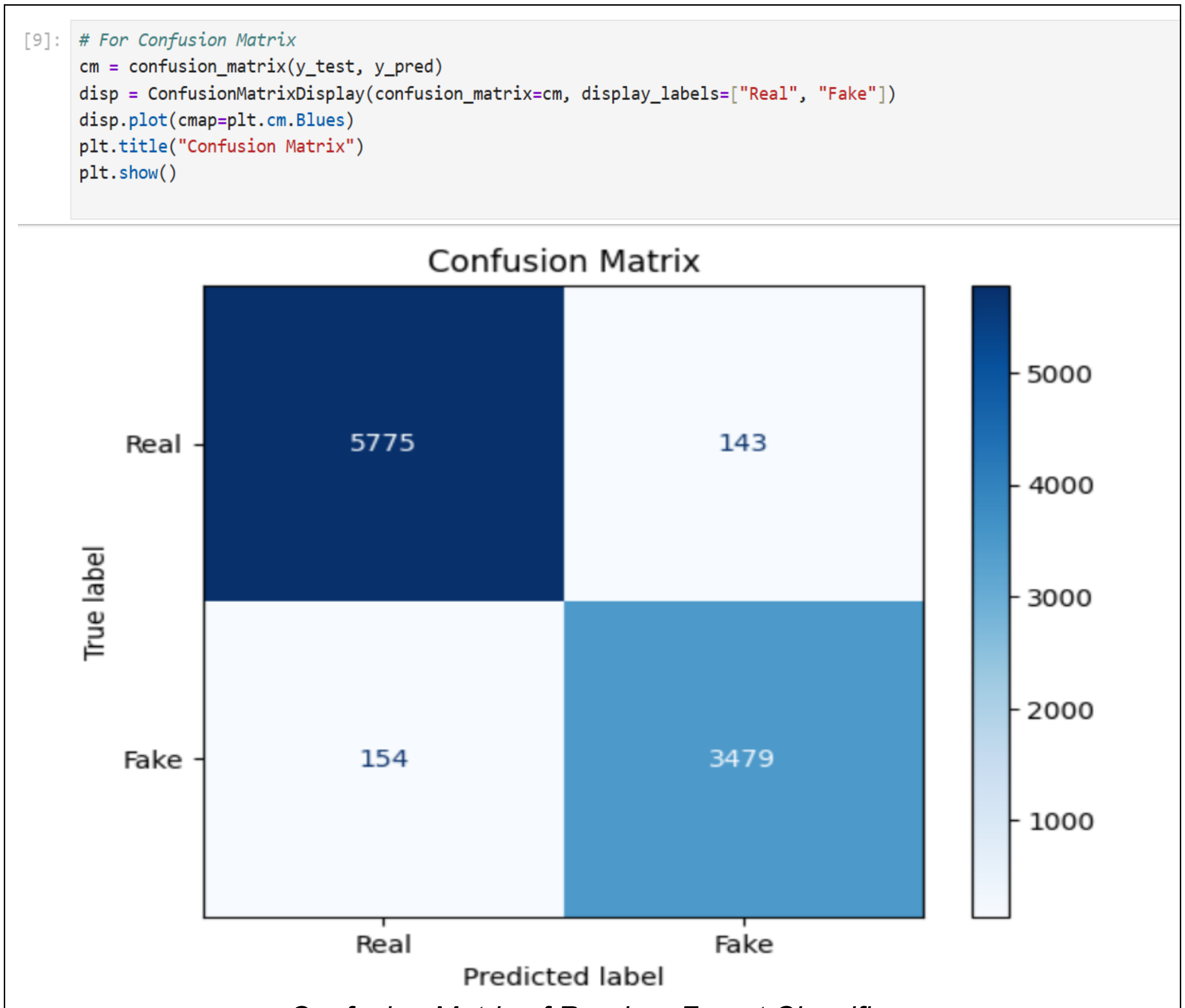
Comparison to other samples resulted in a very high accuracy borne by the model at **96.89%,** which is indeed a good aspect of the model.

The following metrics were used to determine the evaluation process:

- **Accuracy**-The Rate of correct prediction.

- **Confusion Matrix**-This gives a comparative analysis of the performance of classification.
- **Classification Report**-Summaries precision, recall, and F1 score of each class. The model was represented in a confusion matrix plotted visually.

The below figure shows the **confusion matrix** that can be used to obtain the classification performance:



```
[9]: # For Confusion Matrix
     cm = confusion_matrix(y_test, y_pred)
     disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Real", "Fake"])
     disp.plot(cmap=plt.cm.Blues)
     plt.title("Confusion Matrix")
     plt.show()
```

*Confusion Matrix of Random Forest Classifier*

Model had **5,775** predictions on **real** and **3,479** predictions on fake audio correct and the minorities were only wrongly classified (143 incorrect samples of real audio treat ed as fake audio and 154 incorrect form of fake audio treated as real audio).

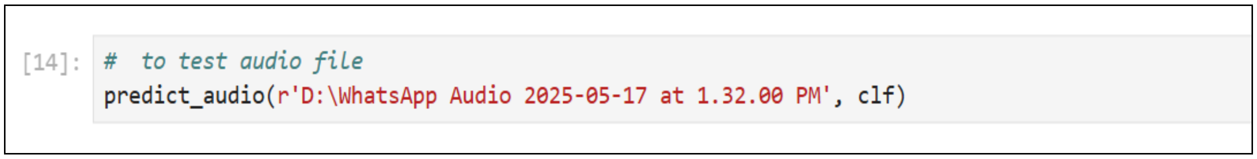
## 3.6 Real-Time Audio Prediction

It is possible to make predictions of any sound sample, which can not be seen with the help of trained Classifier, but with the help of the Random Forest, according to the system.

**Predict audio** venture transforms both.mp3 and.wav, the.flac and even the wav.noisered audio, automatically extracts MFCC features, and classifies the input as **Real** or **Fake**.

It handles full paths with or without extensions and reports missing files. Upon being achieved by the preprocessing and feature extraction of the real dataset folder and the fake dataset folders, the classifier will be able to automatically predict the authenticity of the new audio samples with the right method and reliable results.

**Example Usage:**



```
[14]: #  to test audio file
      predict_audio(r'D:\WhatsApp Audio 2025-05-17 at 1.32.00 PM', clf)
```

# 4. Results

## 4.1 Dataset Statistics

The dataset was loaded from the path:
**D:\final year project5**

| Class | Samples |
|---|---|
| **Real** | 29,838 |
| **Fake** | 17,915 |
| **Skipped** | 1 |

## 4.2 Model Performance

Random Forest model that had been worked out with the effect of **96.89%** showed that it is precise in the setting of differentiating the reality and the fake audio samples.

This is also handled comprehensively in the competitive performance in the confusion table:

| Actual \ Predicted | Real | Fake |
|---|---|---|
| **Real** | 5,775 | 143 |
| **Fake** | 154 | 3,479 |

## 5. Discussion

**Strengths:**

- High detection **(96.89%)** with low numbers of misclassification represented in form of and confusion matrix where there are few number of false as real (143) and false as fake (154) samples.
- Quick, deployed, trainable and interpretable machine learning (Random Forest). It can be in real-time, as well as the prediction of audio may be present.

**Limitations:**

- It is likely that the performance graver will exist throughout the lifetime of the aggregate of specific good-quality, deepfakes, or a deepfakes of the emergent, correlating on new notions, which has yet to be solidified in the selection of the ensure.
- The dataset requires continuous updating to cover emerging deepfake generation techniques.
- The features acquired are the only features of MFCC and the model may be formed by adding more and more features in an attempt to increase the accurate depiction.

**Future Work:**

- Explore deep learning approaches (e.g., CNNs on spectrograms) for potentially higher accuracy.
- Extend the dataset to include multilingual voices and diverse audio conditions.
- Deploy the model as a web or mobile application for real-time deepfake audio detection.

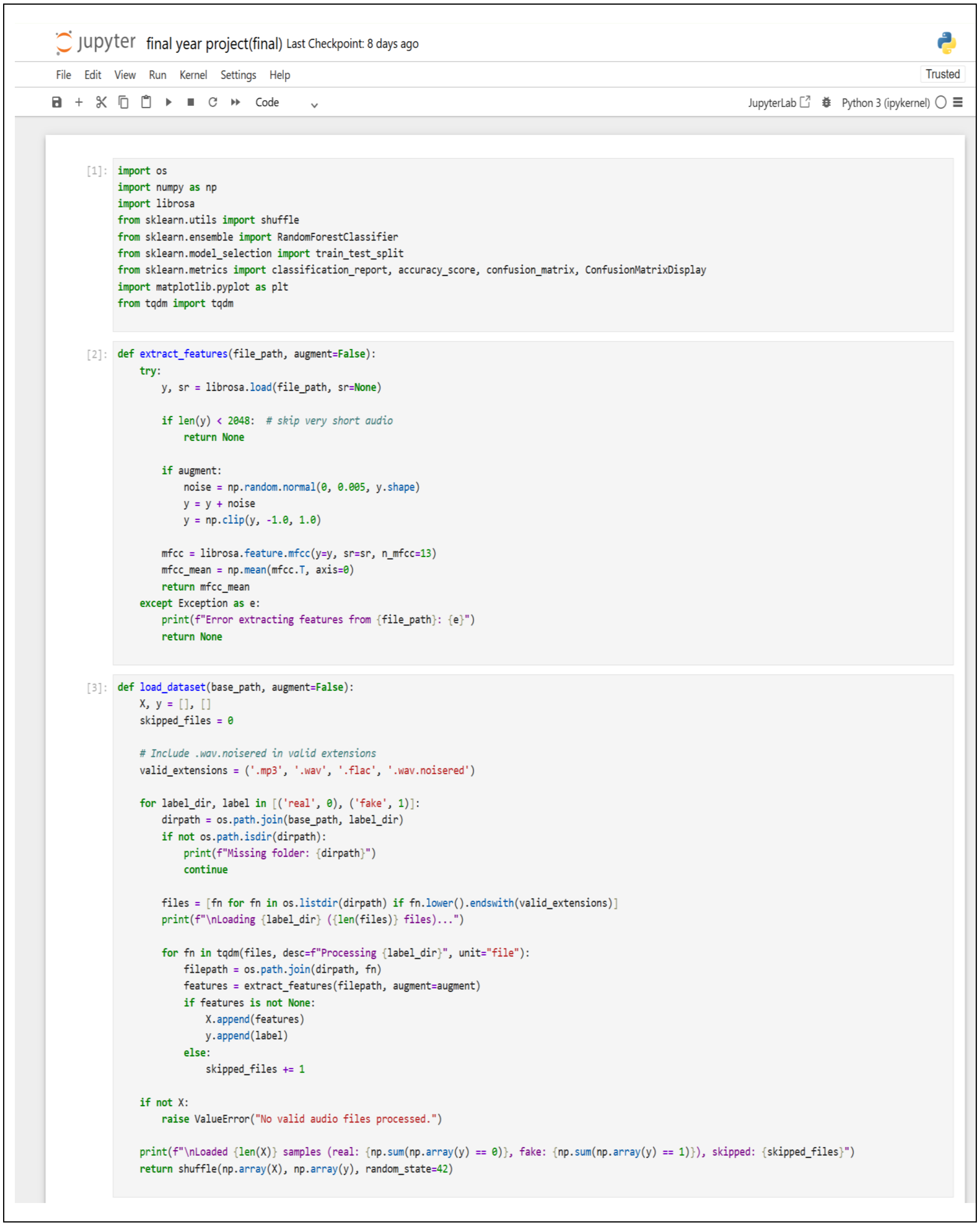
## 6. Conclusion

The framework of the considered project offers an efficient audio deepfake detector system utilizing the aid of the AI system **(MFCC involves extraction and a Random Forest classifier)**: Capable of achieving the score of **96.89**% on the test set, the system under question can identify authentic audio speech and audio fake speech with the same degree of anticipation and the figures prove this detail of the findings of the discussed system with some few exceptions. The latter also contributes to the fact that the combination of the MFCC attributes with the Random Forest is a viable offering when it comes to being able to prove an acceptable balance between simplificiancy and the ability to interpret, and the performance, as well as, the fact that the target solution can also be produced with this particular predict-audio feature allows the classifying the unknown samples in real-time, which consequently makes the solution adequately good to attend to.

In **summary**, this project successfully establishes a **foundation for deepfake audio detection** with high accuracy, real-time prediction capability, and practical deployment potential. Future enhancements in dataset diversity, feature engineering, and advanced modeling will help ensure resilience against evolving deepfake generation methods.

# 7. Appendix

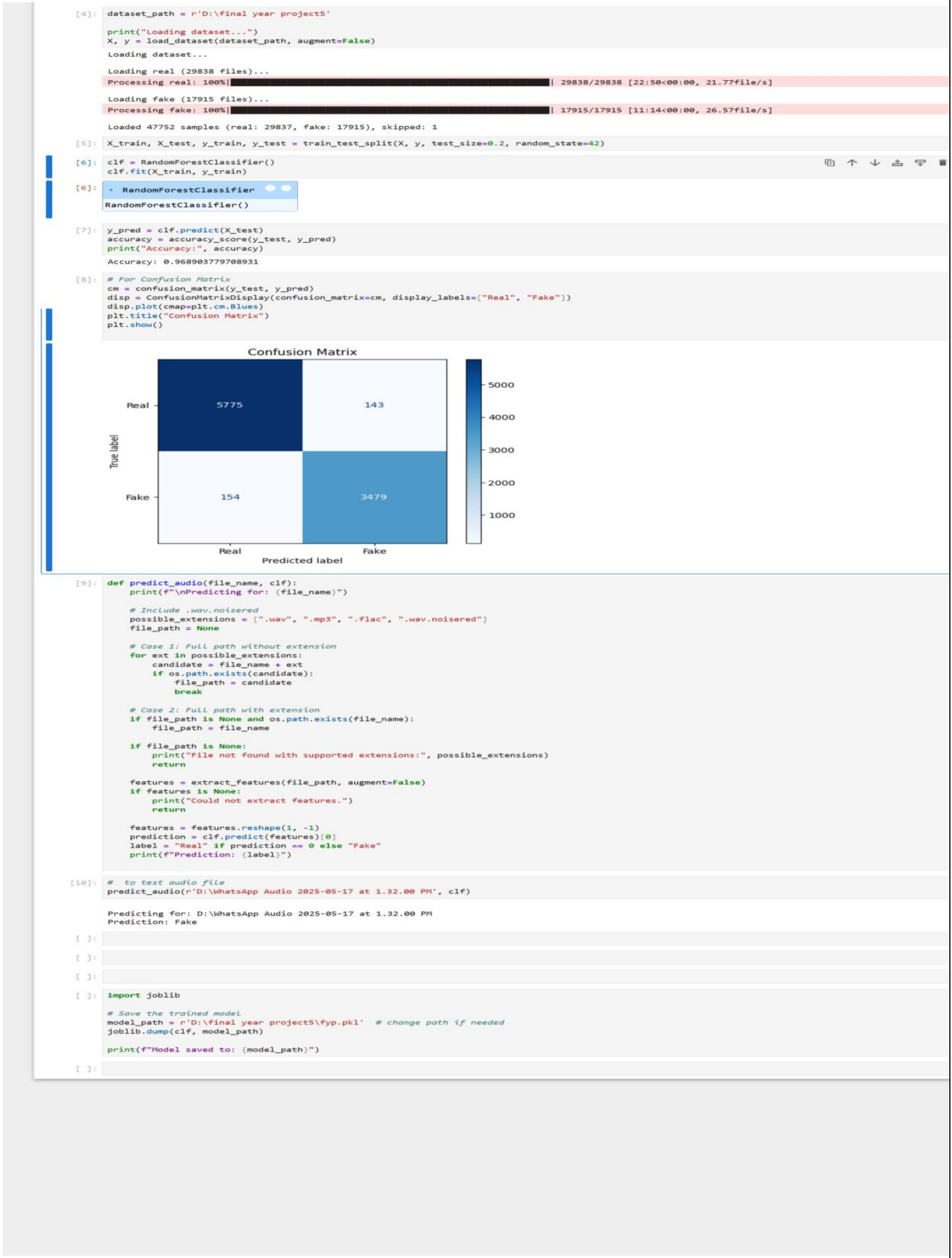
## 7.1 Code Snippets


jupyter final year project(final) Last Checkpoint: 8 days ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

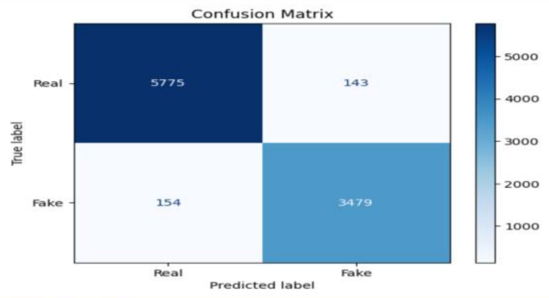
[1]:
```python
import os
import numpy as np
import librosa
from sklearn.utils import shuffle
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt
from tqdm import tqdm
```

[2]:
```python
def extract_features(file_path, augment=False):
    try:
        y, sr = librosa.load(file_path, sr=None)

        if len(y) < 2048:  # skip very short audio
            return None

        if augment:
            noise = np.random.normal(0, 0.005, y.shape)
            y = y + noise
            y = np.clip(y, -1.0, 1.0)

        mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13)
        mfcc_mean = np.mean(mfcc.T, axis=0)
        return mfcc_mean
    except Exception as e:
        print(f"Error extracting features from {file_path}: {e}")
        return None
```

[3]:
```python
def load_dataset(base_path, augment=False):
    X, y = [], []
    skipped_files = 0

    # Include .wav.noisered in valid extensions
    valid_extensions = ('.mp3', '.wav', '.flac', '.wav.noisered')

    for label_dir, label in [('real', 0), ('fake', 1)]:
        dirpath = os.path.join(base_path, label_dir)
        if not os.path.isdir(dirpath):
            print(f"Missing folder: {dirpath}")
            continue

        files = [fn for fn in os.listdir(dirpath) if fn.lower().endswith(valid_extensions)]
        print(f"\nLoading {label_dir} ({len(files)} files)...")

        for fn in tqdm(files, desc=f"Processing {label_dir}", unit="file"):
            filepath = os.path.join(dirpath, fn)
            features = extract_features(filepath, augment=augment)
            if features is not None:
                X.append(features)
                y.append(label)
            else:
                skipped_files += 1

    if not X:
        raise ValueError("No valid audio files processed.")

    print(f"\nLoaded {len(X)} samples (real: {np.sum(np.array(y) == 0)}, fake: {np.sum(np.array(y) == 1)}), skipped: {skipped_files}")
    return shuffle(np.array(X), np.array(y), random_state=42)
```



```
[4]: dataset_path = r'D:\final year project5'

     print("Loading dataset...")
     X, y = load_dataset(dataset_path, augment=False)

     Loading dataset...

     Loading real (29838 files)...
     Processing real: 100%|████████████████████████████████████████| 29838/29838 [22:50<00:00, 21.77file/s]

     Loading fake (17915 files)...
     Processing fake: 100%|████████████████████████████████████████| 17915/17915 [11:14<00:00, 26.57file/s]

     Loaded 47752 samples (real: 29837, fake: 17915), skipped: 1

[5]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

[6]: clf = RandomForestClassifier()
     clf.fit(X_train, y_train)

[6]: ▾ RandomForestClassifier
     RandomForestClassifier()

[7]: y_pred = clf.predict(X_test)
     accuracy = accuracy_score(y_test, y_pred)
     print("Accuracy:", accuracy)

     Accuracy: 0.968903779708931

[8]: # For Confusion Matrix
     cm = confusion_matrix(y_test, y_pred)
     disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Real", "Fake"])
     disp.plot(cmap=plt.cm.Blues)
     plt.title("Confusion Matrix")
     plt.show()
```



```
[9]: def predict_audio(file_name, clf):
         print(f"\nPredicting for: {file_name}")

         # Include .wav.noisered
         possible_extensions = [".wav", ".mp3", ".flac", ".wav.noisered"]
         file_path = None

         # Case 1: Full path without extension
         for ext in possible_extensions:
             candidate = file_name + ext
             if os.path.exists(candidate):
                 file_path = candidate
                 break

         # Case 2: Full path with extension
         if file_path is None and os.path.exists(file_name):
             file_path = file_name

         if file_path is None:
             print("File not found with supported extensions:", possible_extensions)
             return

         features = extract_features(file_path, augment=False)
         if features is None:
             print("Could not extract features.")
             return

         features = features.reshape(1, -1)
         prediction = clf.predict(features)[0]
         label = "Real" if prediction == 0 else "Fake"
         print(f"Prediction: {label}")

[10]: #  to test audio file
      predict_audio(r'D:\WhatsApp Audio 2025-05-17 at 1.32.00 PM', clf)

      Predicting for: D:\WhatsApp Audio 2025-05-17 at 1.32.00 PM
      Prediction: Fake

[ ]:

[ ]:

[ ]:

[ ]: import joblib

     # Save the trained model
     model_path = r'D:\final year project5\fyp.pkl'  # change path if needed
     joblib.dump(clf, model_path)

     print(f"Model saved to: {model_path}")

[ ]:
```