**Part 1: Research and Selection**

After reviewing the GitHub repository, I have selected the 3 models that most promise to work for our Audio Deepfake detection use case.

Points to be followed for selecting the ideal model.

* Detecting AI-generated human speech
* Potential for real-time or near real-time detection
* Analysis of real conversations

**1. AASIST: Audio Anti-Spoofing Using Integrated Spectro-Temporal Graph Attention Networks**

**Why I Chose It:**

* Uses Graph Attention Networks (GATs) to analyse both spectral and temporal aspects of audio. This makes it robust to different types of AI-generated speech.
* Achieves state-of-the-art performance with low Equal Error Rate (EER), meaning it's highly accurate at distinguishing real from fake audio.
* Ideal for deepfake detection in conversational settings, as it captures fine-grained distortions in speech patterns.

**Potential Challenges:**

* GATs can be computationally expensive, making real-time deployment difficult.
* Might need optimization for low-latency inference to work efficiently in real-time applications.

**2. RawNet2 with Sinc Filters**

**Why I Chose It:**

* Processes raw waveforms directly (rather than relying on handcrafted features like MFCCs), reducing preprocessing time.
* Uses Sinc filters for feature extraction, which improves efficiency while maintaining accuracy.
* Shows **strong performance (EER of 1.12%)**, making it a solid choice for detecting deepfake speech.

**Potential Challenges:**

* May not generalize well across different types of synthetic speech, especially if trained on a limited dataset.
* It might need fine-tuning or additional augmentation techniques to adapt to real-world conversational data.

**3. Extreme Gradient Boosting (XGBoost) for Real-Time Detection**

**Why I Chose It:**

* Extremely fast inference time (~0.004 ms per second of audio) makes it ideal for real-time applications.
* Uses lightweight, interpretable features instead of deep learning, reducing computational overhead.
* Achieves **99.3% accuracy**, making it a strong candidate for efficient AI speech detection.

**Potential Challenges:**

* Performance depends on handcrafted temporal features, which might not be as effective against advanced AI speech models.
* Could struggle with adversarial attacks or evolving deepfake technologies that manipulate temporal cues.

**Part 2: Implementation**

**Feature Extraction**

To effectively classify audio samples, the following features were extracted:

* **Linear Predictive Coding (LPC)**: Captures the spectral envelope of speech.
* **Mel-Frequency Cepstral Coefficients (MFCCs)**: Represents the spectral characteristics of audio signals.
* **Zero Crossing Rate (ZCR)**: Measures the number of times the signal changes sign.
* **Spectral Features**: Includes spectral centroid, bandwidth, contrast, and roll-off.
* **Harmonics-to-Noise Ratio (HNR)**: Measures periodicity in speech.
* **Formants**: Represents resonant frequencies in speech signals.

**Data Preprocessing**

* The dataset metadata was loaded from a TSV file, and file paths were processed.
* Features were extracted for each audio sample.
* Data normalization was applied using **StandardScaler**.

**Model Training**

Two models were trained on the extracted features:

* **XGBoost Classifier**: A gradient-boosted decision tree model.
* **Support Vector Machine (SVM)** with an RBF kernel: Effective for high-dimensional spaces.

Both models were trained using an 80-20 train-test split.

**Hyperparameter Optimization**

* **Grid Search on XGBoost**: Exhaustively searched the best combination of hyperparameters to improve model performance.
* **Random Search on SVM**: Randomized hyperparameter tuning to efficiently explore a broad search space.

**Results and Observations**

* **XGBoost** performed well in handling high-dimensional feature space and provided interpretability through feature importance.
* **SVM with RBF Kernel** effectively captured complex feature distributions and provided competitive performance.
* Grid Search on XGBoost improved accuracy by fine-tuning parameters through exhaustive search.
* Random Search on SVM allowed for efficient hyperparameter tuning, leading to competitive results.
* Feature extraction was a crucial step; missing values in any feature set led to sample rejection.

**Part 3: Analysis**

**Challenges Encountered**

* **Feature Extraction Failures**: Some audio files resulted in missing or erroneous feature values.
  + **Solution**: Implemented error handling and ensured correct file paths.
* **Dimensional Mismatch**: Different feature sets had varying lengths, causing model input issues.
  + **Solution**: Standardized feature vector sizes using padding or truncation.
* **Overfitting Risks**: With a high-dimensional feature space, models risked overfitting to training data.
  + **Solution**: Used cross-validation and regularization techniques.

**Assumptions Made**

* All audio files were assumed to have a consistent sampling rate of 16 kHz.
* Feature extraction methods were assumed to be robust across different synthetic and real speech variations.
* Model performance was primarily evaluated on the given dataset, assuming it represents real-world scenarios.

**Model Selection and Analysis**

Two models were selected for implementation:

* **XGBoost Classifier**: Chosen for its ability to handle high-dimensional data and interpretability.
* **SVM with RBF Kernel**: Selected due to its strength in modeling non-linear feature relationships.

**How the Models Work**

* **XGBoost**: Uses gradient boosting to iteratively improve weak learners (decision trees) and optimize classification.
* **SVM with RBF Kernel**: Maps features into a higher-dimensional space to better separate classes using a hyperplane.

**Performance Results**

* **XGBoost** achieved strong results in handling diverse feature sets.
* **SVM with RBF Kernel** effectively modeled complex distributions but required careful hyperparameter tuning.

**Observed Strengths and Weaknesses**

**Strengths:**

* XGBoost provided feature importance insights, aiding interpretability.
* SVM effectively handled non-linearly separable data.

**Weaknesses:**

* Both models were sensitive to missing features, requiring robust preprocessing.
* SVM was computationally expensive with large datasets.

**Suggestions for Future Improvements**

* Fine-tune hyperparameters using automated search techniques.
* Explore deep learning models like CNNs or RNNs for improved accuracy.
* Integrate additional features such as prosodic and phonetic cues.

**Significant Challenges in Implementation**

* Ensuring robustness in feature extraction across various audio conditions.
* Addressing feature dimensionality issues and preventing overfitting.
* Managing computational complexity for real-time inference.

**Real-World Performance vs. Research Datasets**

* Real-world data may contain more noise, requiring additional preprocessing.
* Variability in speech characteristics could impact model generalization.
* Adversarial deepfake attacks may evolve, necessitating continuous model updates.

**Additional Data and Resources for Improvement**

* A larger, more diverse dataset encompassing different synthetic speech methods.
* Advanced augmentation techniques to enhance model robustness.
* Computational resources for training deep learning-based alternatives.

**Deployment Considerations**

* **Scalability**: Optimize inference time for real-world application.
* **Robustness**: Ensure the model generalizes well to unseen deepfake techniques.
* **Integration**: Deploy via a web API or embedded system for real-time detection.