**Machine Learning Project Documentation**

**Deployment**

**1. Overview**

The deployment phase involves making the machine learning model accessible for real-world use. In our case, the deepfake detection model is deployed using Flask, a micro web framework for Python. The deployment process ensures that users can interact with the model through a user-friendly web interface.

**2. Model Serialization**

The trained model is serialized to facilitate its storage and deployment. In this project, the model is serialized using the Keras ‘load\_model’ function, which saves the model architecture, optimizer, and learned weights. This serialized model is stored in the "savedmodel" directory of the project.

**3. Model Serving**

The serialized model is served using a Flask web application. The web app provides an endpoint that accepts video files, preprocesses them, and makes predictions using the loaded model.

**4. API Integration**

In this deployment, the API is integrated into the Flask web application. Users can interact with the model by uploading video files through the web interface. The API endpoint for predictions is the root ("/") of the web application.

If applicable, detail how the machine learning model is integrated into an API for easy access. Include information on API endpoints, input formats, and response formats.

**5. Security Considerations**

File Upload Security: A check is implemented to ensure that only files with the ".mp4" extension can be uploaded. This mitigates the risk of uploading malicious files.

Directory Creation: The application ensures that the "uploads" directory is created securely to prevent unauthorized directory creation.

**6. Monitoring and Logging**

The Flask application logs relevant information, including the current working directory and any errors that occur during processing. Logging aids in debugging and monitoring.