## Project Documentation: 3D Model Integrity Predictor

### Introduction

* **Project Title**: 3D Printer Material Prediction Using Machine Learning
* **Team leader**:
  + Yash Bavale

### Project Overview

* **Purpose**: This project aims to analyze sets of 3D coordinates (vertices) to predict whether the resulting geometric model contains common structural errors. This serves as a preliminary validation step before a model is sent for 3D printing or used in a rendering engine, saving time and resources.
* **Features**:
  + Accepts 3D coordinates in a structured format.
  + Analyzes coordinate data to predict potential geometric errors.
  + Identifies issues such as non-manifold geometry, holes, and intersecting faces.
  + Provides a simple pass/fail prediction with details on detected errors.

### System Architecture

* **Coordinate Input**: The system receives model data via a REST API endpoint. The data, containing a list of vertices and faces, is expected in JSON format.
* **Analysis Engine**: The core of the project is a Python-based analysis engine. It uses geometric algorithms to construct a mesh from the input coordinates and then performs a series of validation checks.
* **Prediction Model**: The prediction is currently based on a deterministic, rule-based algorithm. If any of the predefined error checks fail (e.g., an edge is shared by more than two faces), the model is flagged as having an error.
* **API Structure**: A simple API built with Flask provides an endpoint (/predict) to submit coordinate data and receive an analysis report.

### Folder Structure

* /api: Contains the Flask application for the API endpoints.
* /src: Includes the core analysis modules and geometric validation functions.
* /tests: Contains unit tests for the analysis functions and integration tests for the API.
* /data: Holds sample coordinate files (both valid and erroneous) for testing.

### Running the Application

* To start the API server locally, run the following command from the root directory:
* Bash
* python api/server.py
* You can then send a POST request with your coordinate data to http://127.0.0.1:5000/predict.

### API and Data Documentation

* **Key Components**:
  + /predict **[POST]**: The primary endpoint for submitting data.
    - **Request Body**: Expects a JSON object with a coordinates key containing an array of [x, y, z]vertices.
    - **Response**: Returns a JSON object with a prediction ('error' or 'no\_error') and a details array listing any specific issues found.
* **Error Types Predicted**:
  + **Non-Manifold Geometry**: Detects edges connected to more than two faces.
  + **Holes in Mesh**: Identifies boundaries where edges are not closed.
  + **Flipped Normals**: Checks for faces with inconsistent orientation.

### Analysis and Prediction Logic

* **Global State**: The system is stateless. Each API request is processed independently without relying on previous requests.
* **Error Detection Logic**: The prediction is not based on machine learning but on established geometric principles. The analysis engine iterates through the mesh created from the coordinates and applies validation algorithms for each of the supported error types. If any algorithm detects an issue, the prediction is marked as an "error."

### User Interface

* While the core of the project is the API, a simple HTML form is provided at the root URL (/) for manually uploading a JSON file with coordinates and viewing the prediction result directly in the browser.

### Testing

* **Testing Strategy**: We use the **PyTest** framework for testing.
  + **Unit Tests**: Verify the correctness of individual geometric analysis functions using predefined coordinate sets.
  + **Integration Tests**: Test the full request/response cycle of the /predict API endpoint.
* **Test Coverage**: Our test suite includes a variety of sample models, both valid and intentionally corrupted, to ensure the prediction logic is robust.

### Known Limitations

* It currently only accepts a simple list of vertices and assumes a basic triangulation; it does not support complex polygon meshes.

### Future Enhancements

* Implement a machine learning model to predict more subtle errors or the *likelihood* of a print failure.
* Add functionality to suggest or automatically perform basic repairs on detected errors.
* Support for direct file uploads of common 3D model formats like .STL and .OBJ.