**Description of Folders and Python Programs**

**1. data\_ingestion/**

Handles the ingestion of various data types into the system.

* **clinical\_notes\_ingestion.py**: Functions to load and store textual clinical examination notes.
* **image\_ingestion.py**: Functions to import intraoral and extraoral photographs.
* **three\_d\_ingestion.py**: Functions to load 3D models of dental arches.
* **radiograph\_ingestion.py**: Functions to import OPG and lateral cephalometric X-rays.

**2. data\_preprocessing/**

Responsible for preprocessing the ingested data.

* **text\_preprocessing.py**: Cleans and encodes textual data.
* **image\_preprocessing.py**: Normalizes and augments image data.
* **three\_d\_preprocessing.py**: Simplifies and aligns 3D models.
* **radiograph\_preprocessing.py**: Enhances and processes radiographic images.

**3. feature\_extraction/**

Extracts features from the preprocessed data.

* **text\_feature\_extraction.py**: Uses NLP models to extract textual features.
* **image\_feature\_extraction.py**: Utilizes CNNs to extract features from photographs.
* **three\_d\_feature\_extraction.py**: Processes 3D data using 3D CNNs or PointNet.
* **radiograph\_feature\_extraction.py**: Extracts features from X-ray images using specialized CNNs.

**4. models/**

Contains the machine learning models used for prediction.

* **cnn\_models.py**: Defines CNN architectures for image and radiograph data.
* **nlp\_models.py**: Implements NLP models like BERT or GPT for text data.
* **three\_d\_models.py**: Contains models for processing 3D dental arch data.
* **fusion\_layer.py**: Handles the fusion of features from different modalities.
* **classification\_layer.py**: Contains the final classification layers and loss functions.

**5. decision\_layer/**

Makes the final decision based on model outputs.

* **thresholding.py**: Applies thresholding to model probabilities to decide on brace requirement.
* **explanation\_module.py** *(Optional)*: Generates explanations or visualizations for decisions.

**6. evaluation/**

Evaluates model performance and handles continuous learning.

* **performance\_metrics.py**: Calculates metrics like accuracy, precision, recall, F1-score, and AUC-ROC.
* **feedback\_loop.py**: Implements mechanisms for continuous learning and model updates.

**7. deployment/**

Handles the deployment aspects of the system.

* **api\_integration.py**: Integrates the system with insurance and other external APIs.
* **user\_interface.py**: Defines dashboards and interfaces for clinicians and agents.

**8. utils/**

Utility functions and helpers.

* **data\_utils.py**: General data handling utilities.
* **model\_utils.py**: Utilities for model training, saving, and loading.
* **visualization\_utils.py**: Functions for plotting and visualizing data and results.

**9. configs/**

Configuration files for the project.

* **config.py**: Contains configuration variables and settings.

**10. Root Files**

* **main.py**: The entry point of the application, orchestrating the workflow.
* **requirements.txt**: Lists all Python dependencies and libraries.
* **README.md**: Documentation and setup instructions for the project.

**Additional Notes**

* **Modularity**: Each folder represents a layer in your system architecture, promoting separation of concerns.
* **Scalability**: This structure allows for easy addition of new features or data types.
* **Maintainability**: Organized codebase simplifies debugging and collaborative development.
* **Visual Studio Code Integration**: The structure is compatible with VS Code's project layout, facilitating easy navigation and code management.

**Getting Started**

1. **Clone the Repository**: Begin by cloning the OrthoAI repository to your local machine.
2. **Install Dependencies**: Navigate to the project directory and run pip install -r requirements.txt.
3. **Configure Settings**: Update configs/config.py with necessary configuration settings.
4. **Run the Application**: Execute python main.py to start the system.