To structure your **machine learning application repository**, you should create a well-organized folder structure that supports **development, testing, logging, data management, and deployment**. Below is a recommended **directory structure** along with steps to set it up and complete your ML application.

**📂 Recommended Folder Structure for ML Application**

bash

CopyEdit

📂 ml\_project/

│── 📂 data/ # Stores raw, processed, and intermediate data

│ ├── raw/ # Raw input data (never modified)

│ ├── processed/ # Processed data ready for training/testing

│ ├── external/ # External datasets (if applicable)

│

│── 📂 notebooks/ # Jupyter notebooks for exploratory data analysis (EDA)

│ ├── 01\_data\_exploration.ipynb

│ ├── 02\_feature\_engineering.ipynb

│ ├── 03\_model\_training.ipynb

│ ├── 04\_model\_evaluation.ipynb

│

│── 📂 src/ # Source code for the project

│ ├── 📂 data\_processing/ # Scripts for data cleaning, transformation

│ ├── 📂 models/ # ML model training and inference scripts

│ ├── 📂 utils/ # Utility functions (logging, config loading, etc.)

│ ├── train.py # Main script to train the model

│ ├── evaluate.py # Script for model evaluation

│ ├── predict.py # Script to make predictions

│

│── 📂 tests/ # Unit tests for code and ML models

│ ├── test\_data\_processing.py

│ ├── test\_model.py

│

│── 📂 config/ # Configuration files for model, training, and logging

│ ├── config.yaml # Stores hyperparameters, paths, etc.

│ ├── logging.conf # Logging configuration

│

│── 📂 logs/ # Log files for debugging

│ ├── training.log

│ ├── inference.log

│

│── 📂 artifacts/ # Stores trained models, output files

│ ├── models/ # Saved trained models

│ ├── reports/ # Evaluation reports

│

│── 📂 deployment/ # Scripts for deploying model as an API

│ ├── app.py # Flask/FastAPI script for model inference

│ ├── requirements.txt # Dependencies for deployment

│

│── .gitignore # Ignore unnecessary files

│── README.md # Project documentation

│── requirements.txt # List of dependencies

│── setup.py # Installation script (optional)

**🔧 Steps to Set Up the Repository**

Follow these steps to structure your **ML project**:

**Step 1: Clone Your Repository**

sh

CopyEdit

git clone <repository\_url>

cd ml\_project

**Step 2: Create Folder Structure**

Use the following command to create the directories:

sh

CopyEdit

mkdir -p data/raw data/processed data/external notebooks src/data\_processing src/models src/utils tests config logs artifacts/models artifacts/reports deployment

**Step 3: Initialize a Virtual Environment**

sh

CopyEdit

python -m venv venv

source venv/bin/activate # On Mac/Linux

venv\Scripts\activate # On Windows

**Step 4: Install Dependencies**

Create a **requirements.txt** and install dependencies:

sh

CopyEdit

pip install numpy pandas scikit-learn matplotlib seaborn jupyterlab pytest pyyaml

pip freeze > requirements.txt

**Step 5: Set Up Logging**

Modify config/logging.conf:

ini

CopyEdit

[loggers]

keys=root

[handlers]

keys=consoleHandler,fileHandler

[formatters]

keys=formatter

[logger\_root]

level=DEBUG

handlers=consoleHandler,fileHandler

[handler\_consoleHandler]

class=StreamHandler

level=DEBUG

formatter=formatter

args=(sys.stdout,)

[handler\_fileHandler]

class=FileHandler

level=DEBUG

formatter=formatter

args=('logs/training.log', 'w')

[formatter\_formatter]

format=[%(asctime)s] %(levelname)s - %(message)s

**Step 6: Add a README**

Write a clear **README.md** with:

* Project objective
* How to run scripts
* How to train the model
* Deployment instructions

**💡 Next Steps**

1. **Load & preprocess data** (src/data\_processing/)
2. **Train model** (train.py)
3. **Evaluate & log results** (evaluate.py)
4. **Deploy model as an API** (deployment/app.py)
5. **Write tests** (tests/)