MLOps (Machine Learning Operations) is **DevOps for ML models** — it combines **ML development** with **automated deployment**, **monitoring**, and **management** in production.

| **Area** | **Tools/Skills** |
| --- | --- |
| **Python** | Core language for ML |
| **ML Frameworks** | scikit-learn, TensorFlow, PyTorch |
| **Version Control** | Git, GitHub |
| **Experiment Tracking** | MLflow (FREE) |
| **Containerization** | Docker |
| **CI/CD Pipelines** | GitHub Actions |
| **Model Serving** | FastAPI / Flask |
| **Monitoring** | Prometheus + Grafana (for advanced) |
| **Data Versioning** | DVC (Data Version Control) |

BUT for free sources

| **Platform** | **Use** | **Cost** | **Why Use It** |
| --- | --- | --- | --- |
| **Google Colab** | ML model training | Free | Easy, no setup |
| **GitHub** | Version control, CI/CD | Free | Student-friendly |
| **DockerHub** | Image repository | Free | For container hosting |
| **MLflow** | Experiment tracking | Free (local) | Track parameters, models |
| **DVC** | Data + model versioning | Free | Works with Git |
| **Streamlit** | Model app frontend | Free | Show models live |
| **Render / Hugging Face Spaces** | Deployment | Free tier | Deploy Streamlit/FastAPI |

**DVC** stands for **Data Version Control**.

DVC is like Git, but for **data, models, and experiments**.

It is an **open-source** tool that:

* Tracks large data files and ML models (without pushing to Git)
* Stores actual data in **cloud/remote** (Google Drive, S3, Azure, etc.)
* Keeps lightweight **metadata pointers** in Git
* Helps **reproduce** ML pipelines with exact data and code

1. dvc init                     ← Initialize DVC in project

2. dvc add data.csv             ← Track data file

3. git add data.csv.dvc         ← Add metadata pointer to Git

4. git commit -m "Track dataset"

5. dvc push                     ← Push actual data to remote (e.g., S3)

6. dvc pull                     ← Get exact data version later

Imagine you're working on an ML model to predict housing prices.

* V1: data.csv had 10,000 records
* V2: You added 2,000 more records
* Using **DVC**, you can switch back and forth:

git checkout v1

dvc checkout ← Loads old dataset

1. **Data Pipeline**: Load → Clean → Feature Engineer → Save version
2. **Model Pipeline**: Train → Evaluate → Save model (joblib/pickle)
3. **Track Experiments**: Use MLflow to compare runs
4. **CI/CD Integration**: GitHub Actions to trigger model test/build
5. **API Integration**: Use FastAPI to serve models
6. **Deployment**: Streamlit app to display results

**Pipeline Tools** (Free):

* scikit-learn Pipelines
* MLflow for model stages
* GitHub Actions for automation

Where to Deploy (For Free)?

| **Platform** | **Use** | **Free Tier** | **How to Use** |
| --- | --- | --- | --- |
| **Render** | Deploy Streamlit / FastAPI | Yes | Connect GitHub, auto deploy |
| **Hugging Face Spaces** | Host ML projects | Yes | Great for student showcase |
| **Streamlit Cloud** | Streamlit apps | Yes | One-click GitHub deploy |
| **Replit** | Simple hosting | Yes | Good for small demos |
| **Fly.io / Railway** | FastAPI or Flask APIs | Yes | Easy setup |

mlops\_project/

├── .github/

├── data/

│ ├── preprocessed/

│ │ ├── .gitignore

│ │ ├── cleaned.csv

│ │ └── cleaned.csv.dvc

│ ├── raw/

│ │ └── students.csv

│ └── users.csv

├── logs/

│ └── metrics\_log.csv

├── models/

│ ├── .gitignore

│ ├── columns.pkl

│ ├── model.pkl

│ └── model.pkl.dvc

├── notebooks/

│ └── EDA.ipynb

├── pages/

│ ├── login.py

│ ├── monitor.py

│ └── register.py

├── src/

│ ├── evaluate.py

│ ├── train.py

│ ├── serve\_model.py

│ └── preprocess.py

├── app.py

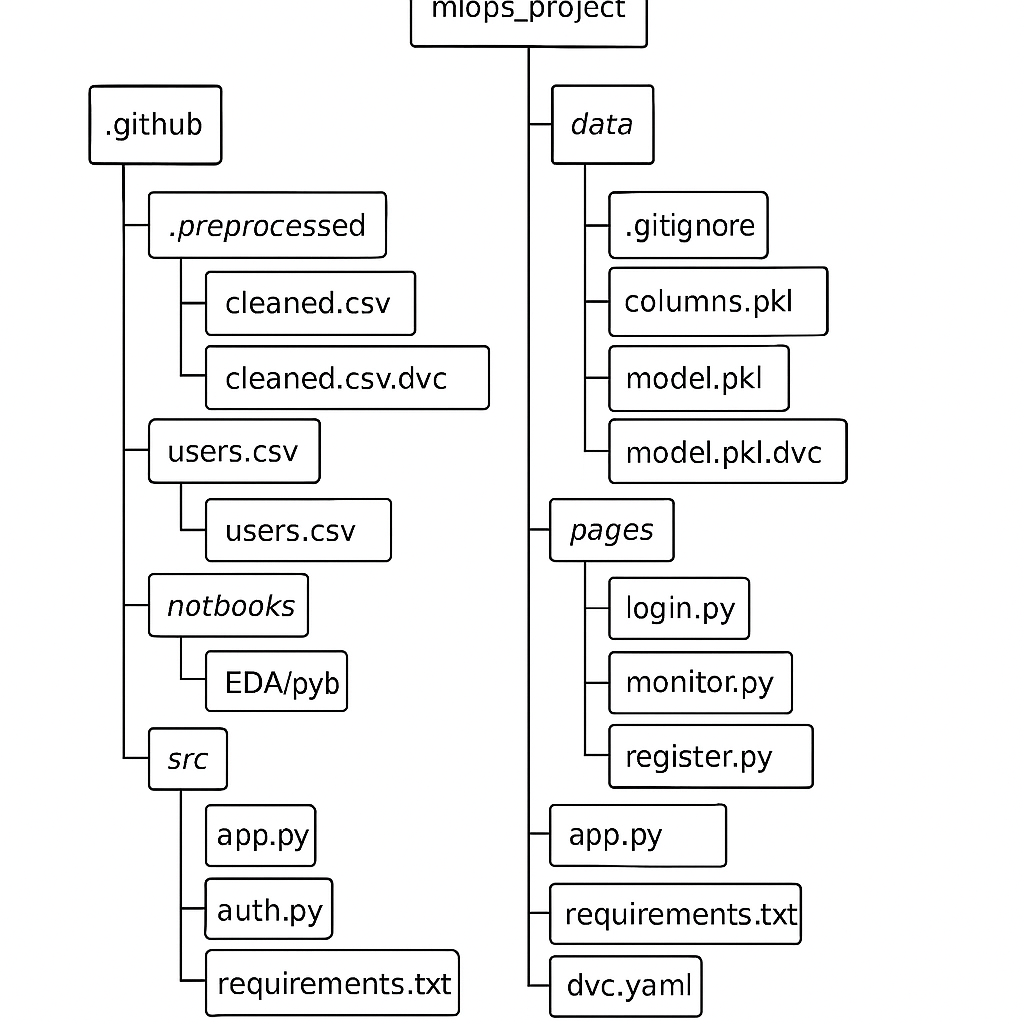
├── auth.py

├── venv/

├── requirements.txt

├── dvc.yaml

└── .dvcignore



STEPs :

mkdir mlops\_project

cd mlops\_project

python -m venv venv

venv/bin/activate

**Install requirements :**

Pip install –r requirements.txt

pandas

scikit-learn

matplotlib

seaborn

joblib

dvc

mlflow>=2.0

fastapi

uvicorn

streamlit

numpy

pydantic

**EXECUTION FLOW STEPS :**

python src/preprocess.py

python src/train.py

mlflow ui

git init

dvc init

dvc add data/processed/cleaned.csv

git add data/.gitignore data/processed/cleaned.csv.dvc

git commit -m "Track cleaned data with DVC"

dvc add models/model.pkl

git add models/model.pkl.dvc

# To run the Fast API

uvicorn src.serve\_model:app --reload

streamlit run app/app.py

For testing :

# for testing

pytest tests/

Update everything in GitHub

git init

git remote remove origin  # optional if already set

git remote add origin https://github.com/KiranVersatile/mlops-fastapi-api.git

git add .

git commit -m "🚀 FastAPI backend for student predictor"

git push -u origin main

git init

git remote remove origin  # only if already exists

git remote add origin https://github.com/KiranVersatile/mlops-student-app.git

git add .

git commit -m "🎯 Streamlit frontend for student predictor"

git push -u origin main

command to paste in render to start

uvicorn serve\_model:app --host 0.0.0.0 --port 10000

…next update API url in the app.py code