📚 Student Dropout Risk & Behavior Clustering – Complete Project Workflow

This project identifies and visualizes patterns in student learning behavior to detect dropout risks using unsupervised machine learning. It combines clustering, anomaly detection, FastAPI, Streamlit, CI/CD pipelines, and is designed to be fully production-ready and free.

# 🗓️ Week-by-Week Workflow

## Week 6: Unsupervised Learning & Dimensionality Reduction

- Tools: scikit-learn, PCA, t-SNE, UMAP  
- Actions:  
 • Preprocess student data (cleaning, encoding, scaling)  
 • Apply clustering techniques: KMeans, DBSCAN  
 • Use PCA/t-SNE for 2D visualization of student behavior  
 • Apply Isolation Forest or DBSCAN for anomaly detection  
 • Save clusters and risk flags  
- Dataset: UCI Student Performance or custom-generated  
- Tips: Visualize clusters using matplotlib/plotly and color-code risk

## Week 7: Model Deployment & API Creation

- Tools: FastAPI, Streamlit, Docker, GitHub Actions  
- Actions:  
 • Create FastAPI endpoints:  
 - /predict-risk: receives student data, returns cluster & risk  
 - /get-stats: returns summary data for dashboard  
 • Build interactive Streamlit dashboard:  
 - Upload CSVs  
 - Visualize risk clusters  
 - Display school-wise insights  
 • Dockerize both API and Streamlit apps  
 • Deploy via Render or HuggingFace (free tier)  
 • Set up GitHub Actions for code/test automation

## Week 8: ML Pipelines & Production Readiness

- Tools: scikit-learn pipelines, joblib, Docker  
- Actions:  
 • Build pipeline: Preprocessing → PCA → Clustering  
 • Save pipeline with joblib  
 • Load pipeline in API and dashboard  
 • Add error handling and validation  
 • Unit test pipeline steps  
 • Document everything in GitHub README

# 🚀 Next-Level Advancements

- Integrate a database (SQLite or Firebase) for persistent student data  
- Add role-based access (admin/teacher) in Streamlit  
- Incorporate time-series dropout behavior (longitudinal analysis)  
- Connect with LMS APIs (like Moodle, Canvas)  
- Convert dashboard to mobile app using Streamlit + Expo + PWA  
- Log model performance and data drifts using MLflow or WandB  
- Trigger retraining using GitHub Actions or Prefect  
- Add LLM-based feedback summarizer (e.g., GPT for free-text surveys)

# ✅ Final Tips

- Keep modular code (pipeline.py, api.py, dashboard.py)  
- Version your datasets and models  
- Write blog posts to explain your approach and decisions  
- Include screenshots or video demo in README for hiring managers