Beginner Machine Learning Project Plan

Timeline: 7 Weeks
Goal: Build a complete, working ML system, while learning how real ML projects are developed collaboratively: raw data → dataset → machine learning model → local web app.
Team Structure: Two teams of 2–3 people.
Office Hours: 2 hours/week (open Q&A, not structured teaching).

What You'll Learn Overall

By the end of this project, each participant should understand:

- How to **set up a Python development environment** (VS Code, Git, Jupyter, LaTeX).
- How to **find, clean, and prepare datasets** for ML.
- How to **engineer and select features**.
- How to **train, evaluate, and tune ML models** using scikit-learn.
- How to **analyze feature importance** and interpret model behavior.
- How to **serve models with a Flask API**.
- How to **create a basic interactive front-end** using Gradio.
- How to **convert notebooks into deployable Python scripts**.
- How to **write concise technical documentation** using LaTeX and Markdown.

Basically, you'll learn how to use datasets found in the wild, train a machine learning model, and how to present what you found through a demo and markdown documents

---

Week 1 – Setup & Dataset Exploration

Focus: Environment setup, dataset selection, first visualizations, and LaTeX setup.

Objectives:

- Learn how to install and use the essential tools (VS Code, Python, Jupyter, Git).
- Learn how to use GitHub collaboratively.
- Understand what a dataset is, what “features” and “targets” mean, and how to visualize basic patterns.
- Learn to write and compile a short LaTeX document.

Meeting 1 Deliverables:

- Install VS Code, Python, Jupyter, and set up GitHub SSH keys.
- Install LaTeX environment (TeX Live or MiKTeX + compiler integration in VS Code).
- Each member creates a LaTeX document (`dataset_summary.tex`) including:
  • Name
  • Dataset title and source (Kaggle, SQL, API)
  • 3–4 main features and target (bullet points)
  • Short paragraph on why it is interesting
- Commit compiled PDF and `.tex` source to `/docs`.
  Include your name in the file name to prevent merge conflicts.

Meeting 2 Deliverables:

- Choose the final dataset.
- Load it into Pandas and display `head()`, `info()`, `describe()`.
- Create at least two visualizations (histogram and bar/scatter).
- Write a short LaTeX note summarizing data quality issues (missing values, data types, etc.).

Week 2 – Data Cleaning + Feature Engineering

Focus: Data preparation and basic feature design.

Objectives:

- Learn to clean and normalize real data.
- Learn encoding for categorical features and scaling for numeric features.
- Practice creating meaningful engineered features that may improve model learning.
- Continue using Pandas for analysis and visualization.

Meeting 1:

- Handle missing values (drop/fill).
- Normalize numeric columns and encode categorical ones.
- Create 1–2 engineered features (ratios, bins, derived columns).

Meeting 2:

- Add three visualizations of engineered features (correlation, distribution, scatter).
- Adjust features before merging (remove or fix weird/skewed columns).
- Merge cleaned data into `data/processed.csv`.
- Write a short LaTeX note explaining feature choices.

Week 3 – Baseline Model + Feature Importance

Focus: Build a first model and analyze which features matter.

Objectives:

- Learn how to split data into training and testing sets.
- Understand how a basic ML model is trained and evaluated.

- Learn about accuracy and error metrics.
- Understand how to inspect feature importance and modify features.

Meeting 1:

- Split data (train/test or train/val/test).
- Train a baseline model (Linear Regression or Logistic Regression).
- Record metrics (Accuracy, MAE, R^2).

Meeting 2:

- Compute feature importance or coefficients.
- Remove irrelevant or dominant features and retrain.
- Compare metrics before and after changes.
- Write a short LaTeX note explaining what was changed and why.

Week 4 – Model Tuning + Comparison

Focus: Experiment with hyperparameters and different algorithms.

Objectives:

- Learn to use scikit-learn model APIs.
- Understand what hyperparameters are and how they influence performance.
- Practice comparing multiple models using validation metrics.
- Learn to visualize and interpret model results.

Meeting 1:

- Train two different models (e.g., Decision Tree, Random Forest).
- Apply random or grid search over key hyperparameters.

Meeting 2:

- Compare models visually (bar chart or metrics table).
- Save the best-performing model as `best_model.pkl`.
- Write a short LaTeX note summarizing tuning results and findings.

Week 5 – Flask API Integration

Focus: Expose the trained model through a local API.

Objectives:

- Learn the concept of REST APIs and how they interact with ML models.
- Understand request/response flow using JSON.
- Learn to use Flask to wrap and serve a model.

Meeting 1:

- Create a basic Flask app with a `/predict` endpoint returning dummy JSON.

Meeting 2:

- Update `/predict` to load a real model and return predictions.
- Verify requests with `curl` or Postman.
- Commit `app.py` and `requirements.txt`.

Week 6 – Gradio Front-End

Focus: Create a simple interactive UI for model testing.

Objectives:

- Learn what front-end inputs and outputs are in ML systems.
- Learn to connect a Gradio interface to a Flask backend.
- Understand how to create a quick end-to-end demo for users.

Meeting 1:

- Team A: build Gradio input/output components.
- Team B: connect the UI to the Flask endpoint.

Meeting 2:

- Fully functional demo (input → prediction → output visual).
- Write a short LaTeX note describing how testing went and any UI issues found.

Week 7 – Consolidation + Documentation

Focus: Convert everything to Python scripts and finalize the project repository.

Objectives:

- Learn to structure and modularize code.
- Learn to document a technical project clearly and professionally.
- Understand how to make a complete project reproducible.
- Reflect on what was learned and what could be improved.

Meeting 1:

- Convert all notebooks into Python scripts:
  `data_pipeline.py`, `train_model.py`, `api_server.py`.

- Ensure each script runs standalone (`python train_model.py`).

- Teams review imports, comments, and style consistency.

Meeting 2:

- Complete final README with:
  • Project summary and goal
  • Graphs showing dataset overview and model results
  • Reason for choosing the final model
  • Setup and run instructions

- Add each member's ½-page LaTeX reflection.

- Verify full end-to-end pipeline runs successfully.

- Each person documents what they worked on and learned, so they can use it for interviews or resumes.

Week 1

Meeting 1 (10/13)

Agenda

- Install and set-up VSCode
    - Github Account, SSH Key, and Clone a repo
        - Create a basic file, run it, and push it to the repo
- Intro to Machine Learning Slideshow
- Brief Introduction to Neural Networks
- Choose topic
- Meeting Schedule
- Assignment of tasks
    - Team Creation

Tasks:

- Find a dataset:
    - Wanted features:
- Download/open the dataset
    - Understand what each feature looks like
        - How would you use this feature?
        - What can it tell you
- Everyone should come back with a dataset they found online (even if its all the same one), and the three most important features they think will be helpful and why. And a feature(s) they think is interesting and could be engineered/combined with another feature to give some more interesting data/better training data.
- BONUS: try reading pandas documentation: https://pandas.pydata.org/docs/
    - Try writing another file where you load the data into some data structure (most likely a DataFrame), and print the first 5 lines (df.head()) or the column metadata (df.columns).
    - This should be attempted with a partner that you want to work with in the group. If one person works without a partner this is fine too, you will have a chance to work with other people later

Meeting 2 (10/16):

Agenda

1. Roundtable discussion regarding datasets
2. Pick a dataset
3. Using UV
4. Go over basic Pandas
5. How to create a table/graph in Python.

6. Task assignment

Tasks:

- Split features per person/group. Each group should work together to analyze each feature for interesting trends (e.g. cat pictures generally have more light (will the model be choosing cat pictures or just pictures that have more light?)). Basically what will a model learn that isn't necessarily just it's the correct answer. Are there any trends/features that a model will rely on the most?