Project Tips and Checklist

This guide summarizes best practices for your midterm and capstone projects.

It's based on Alexey's project discussion video: https://www.youtube.com/live/GuJkBzyGxyc?si=xdgNWXemt-1U_Sph

1. Choosing Your Problem & Dataset

- **Pick something you actually find interesting.**
  A relatable problem keeps you motivated through debugging and deployment.
- **Keep scope realistic.**
  If a Kaggle notebook already solves a similar task, it's likely doable in 2 weeks.
- **Dataset sanity checks**
  - Publicly available and easy to download
  - Not too small (< 500 rows) or too big (> 1 GB)
  - Clear target variable
  - Reasonable amount of missing data (you'll handle it in preprocessing)

> *Tip:* Start by browsing past cohort projects for inspiration.

2. Problem Framing

- Clearly **describe the problem** in the README: what you're predicting, who benefits, and how the model will be used.
- Define an **evaluation metric** appropriate to your task:
  - Balanced classification → Accuracy or AUC-ROC
  - Imbalanced classification → Precision, Recall, or PR Curve
  - Regression → RMSE / MAE
- Add a short note on **why this problem matters**.

3. Data Preparation & EDA

- Document data source and retrieval steps.
- Explore distributions, missing values, correlations, outliers.
- Feature-engineer only what's necessary — keep it interpretable.
- Keep preprocessing steps **reproducible** (functions or scripts).

4. Modeling

- Start with a **baseline** (Logistic/Linear model or shallow tree).
- Train at least **3 models**, tune key hyperparameters, and select the best.
- Use **cross-validation** and keep a summary table of scores.
- Avoid data leakage — don't include target-derived features.

5. From Notebook → Script

Refactor your final notebook into:

```
train.py      # trains model and saves artifact
predict.py    # loads model, runs inference
serve.py      # starts web service
```

6. Web Service

- Implement a **Flask/FastAPI** (or similar) app exposing:
  - `POST /predict` – accepts JSON input, returns prediction
  - `GET /health` – simple 200 OK check
- Provide an example request in the README:

```
curl -X POST -H "Content-Type: application/json" \
     -d '{"feature1": 3.2, "feature2": "A"}' \
     http://localhost:9696/predict
```

7. Dockerization

☑ Required for the project evaluation. Include a working `Dockerfile`, e.g.:

```
FROM python:3.11-slim
WORKDIR /app
COPY . .
RUN pip install -r requirements.txt
EXPOSE 9696
CMD ["python", "serve.py"]
```

Add build/run instructions:

```
docker build -t myproject .
docker run -it -p 9696:9696 myproject
```

8. (Bonus) Cloud Deployment

Optional but earns extra points:

- Deploy to AWS, GCP, Azure, or PythonAnywhere.
- Include:

- Access instructions or screenshots
- A short demo of successful prediction
- Teardown info to avoid extra costs

9. Documentation & Structure

A good README includes:

1. Problem statement
2. Dataset description
3. EDA summary
4. Modeling approach & metrics
5. How to run locally and via Docker
6. API usage example
7. Known limitations / next steps

Add one simple **diagram or architecture sketch** (data → model → API) if possible.

10. Peer Review Readiness

Before submission:

- Clone your repo fresh and ensure everything runs from scratch.
- Tag or note the **exact commit hash** you'll submit.
- Check that all files needed for reproducibility are included.
- Write clear setup and run instructions — reviewers should need <10 minutes to verify.

11. Learning in Public

Share short updates during your build — screenshots, threads, or LinkedIn posts. You can add up to **14 public learning links** in the submission form.

12. Common Pitfalls to Avoid

🚫 Don't skip peer review — fewer than 3 reviews → project fail 🚫 Don't replace Flask/FastAPI + Docker with Gradio/Streamlit (they're great extras, not substitutes) 🚫 Don't copy-paste other students' projects — inspiration ≠ duplication 🚫 Don't leave Docker setup or deployment for the last day

13. Time & Effort

Plan for **15–20 hours** total:

- 5h exploring & EDA
- 5h modeling & tuning
- 3h refactoring + testing
- 5h packaging & deployment
- 2h documentation & peer review prep

14. Evaluation Criteria Summary

| Category | Max Points | Key Expectation | | -- | - | | | Problem description & clarity | 2 | Clear, contextual README | | EDA & feature prep | 2 | Well-explained and clean | | Modeling & tuning | 2 | Several models + selection rationale | | Reproducibility | 2 | Runs end-to-end with provided instructions | | Deployment | 2 | Functional web service with Docker | | Learning in public | +up to 14 | Optional daily posts | | Bonus | +2 | Cloud deployment or notable improvement |

15. Quick Pre-Submission Checklist

- ☐ Problem clearly stated
- ☐ Dataset linked and documented
- ☐ Notebook → script conversion complete
- ☐ Web service works locally
- ☐ Docker container builds & runs
- ☐ README explains setup + API
- ☐ Commit hash noted
- ☐ Peer-review criteria covered
- ☐ Project tested from fresh clone

Good luck, and remember: **Deployment takes longer than expected — start early!**