Week 4 Assignment — AI in Software Engineering

Theme: Building Intelligent Software Solutions

Submission contents

- Code: `code/`
- Report: this document (convert to PDF)
- Figures: `report/figures/` (login_test_result.png, confusion_matrix.png)
- Presentation: demo (not included)

Grading rubric (reference)

- Theoretical Depth & Accuracy — 30%
- Code Functionality & Quality — 50%
- Ethical Reflection — 10%
- Creativity & Presentation — 10%

Part 1 — Theoretical Analysis (30%)

Q1 — How AI-driven code generation tools reduce development time and limitations

AI code-completion tools (e.g., GitHub Copilot) reduce development time by suggesting boilerplate, common idioms, and API usage inline — decreasing the need to search docs and write repetitive code. They accelerate prototyping and can surface edge-case handling patterns from large corpora. Limitations include: incorrect or insecure suggestions requiring review; hallucinations or API misuse; unclear licensing/provenance of generated snippets; over-reliance that erodes deep understanding; and inability to capture project-specific context or architectural constraints. Integration testing and human oversight remain essential.

Q2 — Supervised vs. unsupervised learning in automated bug detection

Supervised learning requires labeled examples (buggy vs. non-buggy) and predicts known bug classes effectively when labels are representative. Unsupervised learning (clustering, anomaly detection) finds outliers without labels and can reveal novel or rare bug patterns, but generally produces more false positives and needs human validation. Use supervised methods for targeted detection and unsupervised approaches for discovery and exploratory monitoring.

Q3 — Why bias mitigation is critical for personalization

Personalization adapts product behavior for users. If training data under- or over-represents subgroups, the system may systematically favor some users and disadvantage others, harming trust and potentially violating fairness expectations or regulations. Bias mitigation (data balancing, fairness-aware algorithms, continuous monitoring) is necessary to ensure equitable user experiences and accountable personalization.

Case Study — AIOps in deployment pipelines

Predictive anomaly detection: ML models trained on monitoring and deployment metrics can predict problematic rollouts (e.g., rising error rates) and trigger pre-deployment checks or halt rollouts to prevent downtime.

Intelligent remediation automation: AIOps systems suggest or execute remediation steps (auto-scale, restart, rollback) based on historical incidents and runbooks, shortening MTTR and reducing manual toil.

Part 2 — Practical Implementation (60%)

Files and locations:

- `code/task1_sorting.py`, `code/task1_sorting_manual.py`, `code/task1_tests.py`
- `code/task2_selenium_login.py`
- `code/task3_predictive.py`
- Figures saved to: `report/figures/`

Task 1 — AI-Powered Code Completion

AI-suggested implementation uses Python's built-in `sorted()` with a lambda key extractor. Manual implementation constructs tuples to preserve stability and handle missing keys explicitly. Prefer the built-in approach for production due to performance and clarity. See `code/task1_tests.py` for unit tests.

Task 2 — Automated Testing with AI

Script: `code/task2_selenium_login.py`. The included demo screenshot was generated using the public test site https://the-internet.herokuapp.com/login with credentials `tomsmith` / `SuperSecretPassword!`. To test your target site, update `LOGIN_URL` and the CSS selectors in the script.

Figure: Login test screenshot placeholder (file expected at `report/figures/login_test_result.png`)

Task 3 — Predictive Analytics for Resource Allocation

Script: `code/task3_predictive.py`. Uses sklearn's breast cancer dataset and a synthetic mapping to a 3-class priority label for demonstration. Training pipeline: scaling, stratified split, RandomForest with class balancing, evaluation metrics, and confusion matrix generation.

Figure: Confusion matrix placeholder (file expected at `report/figures/confusion_matrix.png`)

Example outputs (your run may differ):

- Accuracy: 0.75
- F1 (macro): 0.70

Part 3 — Ethical Reflection (10%)

Potential biases:

- Underrepresentation: teams or platforms may be underrepresented, causing mis-prioritization.
- Label bias: historical priority labels reflect human subjectivity.
- Feature bias: features correlated with team, region, or customer tier could produce unfairness.

Mitigation strategies:

- Use fairness toolkits (e.g., IBM AI Fairness 360) to measure disparities and apply reweighing or post-processing.
- Maintain human-in-the-loop for high-impact decisions.
- Provide explainability (feature importance, SHAP) and continuous monitoring.

Bonus — Innovation Proposal: AutoDoc

Title: AutoDoc — Context-aware Automated Documentation Generator for Microservices

Purpose: Generate living documentation combining static analysis, runtime traces (PII redacted), and LLM synthesis to produce endpoint docs, example payloads, and diagrams.

Workflow

1. Static analysis extracts endpoints and signatures.
2. Runtime instrumentation collects example requests/responses with PII redacted.
3. LLM synthesizes docs and diagrams.
4. CI integration regenerates docs on merge; diffs reviewed in PRs.

Impact: Keeps docs current, reduces onboarding time, and prevents bugs from stale documentation.

Appendix — Produce PDF & submission checklist

How to reproduce figures (Colab or local)

- **Login screenshot (Task 2):** Run the provided Colab cell that uses the demo site; download `/content/login_test_result.png` and place it at `report/figures/login_test_result.png` in the repo.
- **Confusion matrix (Task 3):** Run `code/task3_predictive.py` or the Colab snippet; save `report/figures/confusion_matrix.png`.

Convert report to PDF

Desktop: open this markdown in VSCode or a Markdown viewer and Print > Save as PDF, or use pandoc:

```
pandoc report/week4_report.md -o report/week4_report.pdf
```

Final submission checklist

- [ ] All code files present in `code/` and documented.
- [ ] Figures present in `report/figures/` (login_test_result.png, confusion_matrix.png).
- [ ] Report PDF generated and placed in `report/` (optional but recommended).
- [ ] README updated with run instructions and links.
- [ ] Commit and push all changes to GitHub with message: `Week4: final submission — code, report, figures, docs`.

Commit commands (local)

```
git add .
git commit -m "Week4: final submission — code, report, figures"
git push origin main
```

Prepared for Week 4 — AI in Software Engineering Assignment.