AI Tools in Software Engineering — Report

Part 1: Theoretical Analysis (30%)

Q1: AI-driven code generation tools like GitHub Copilot reduce development time by automating boilerplate code, providing API usage examples, and speeding up prototyping. They save time by offering intelligent suggestions, but limitations include inaccurate or insecure suggestions and potential overreliance.

Q2: Supervised learning detects known bug patterns from labeled data, while unsupervised learning identifies anomalies without labels. Supervised models are precise but require labeled data, whereas unsupervised models discover new bugs but can yield false positives.

Q3: Bias mitigation is crucial in AI-driven UX personalization to prevent discriminatory outcomes. Ensuring balanced datasets and fairness-aware algorithms promotes inclusivity and trust.

Case Study: AIOps enhances deployment efficiency by predicting failures (selective test execution) and automating rollbacks (self-healing deployments), reducing downtime and manual intervention.

Part 2: Practical Implementation (60%)

Task 1: AI-Powered Code Completion
AI-generated and manual Python functions were compared for sorting dictionaries. Both use O(n log n) sorting complexity. AI's version was concise and efficient; the manual version added data validation for robustness.

Task 2: Automated Testing with AI
Selenium AI was used to automate login tests (valid/invalid). AI enhances coverage by identifying flaky tests and generating new test scenarios. Manual testing had lower coverage and higher maintenance overhead.

Task 3: Predictive Analytics for Resource Allocation
Using the Breast Cancer dataset, a Random Forest predicted task priority (low/medium/high). Accuracy: 0.991, Macro F1: 0.991, demonstrating strong model performance.

Part 3: Ethical Reflection (10%)

Potential bias arises if underrepresented teams or features skew training data. Fairness tools like IBM AI Fairness 360 detect and mitigate disparities using methods like reweighting and adversarial debiasing. This ensures model fairness and transparency during deployment.

Bonus Task: Innovation Challenge

Tool Name: DocuGen-AI (Automated Contextual Documentation Generator)

Purpose: Generate and maintain accurate, contextual documentation synchronized with evolving codebases.

Workflow: Parses code and commits → extracts function metadata → uses AI to draft docstrings and examples → reviews via pull requests → auto-syncs with updates.

Impact: Reduces onboarding time, ensures accurate documentation, and increases productivity by automating repetitive documentation tasks.