Part 1: Theoretical Analysis (30%)

1. Short Answer Questions

Q1: Explain how AI-driven code generation tools (e.g., GitHub Copilot) reduce development time. What are their limitations?

AI tools like GitHub Copilot help speed up coding by suggesting code, completing lines, and generating basic code structures. This allows developers to spend more time on important tasks instead of repetitive coding. As a result, projects can be completed more quickly and with more frequent updates.

Limitations:

Understanding Context: AI may not fully grasp the project's overall goals, leading to suggestions that do not fit well.

Code Quality: The code generated might not follow best practices, which can create security issues or slow performance.

Training Data Issues: The quality of the AI's suggestions depends on the data it was trained on. If the data has biases or is lacking, the output can be flawed.

Troubleshooting Challenges: AI tools often struggle to find complex bugs that require more understanding than just basic code snippets.

Q2: Compare supervised and unsupervised learning in the context of automated bug detection.

Supervised Learning:

This method uses labeled data, where examples of buggy code and their fixes are provided. The model learns to recognize patterns that indicate bugs.

Advantages: It can accurately find known issues and apply this knowledge to new code.

Challenges: It needs a lot of labeled examples, which can take time and effort to create.

Unsupervised Learning:

Here, the model looks at unlabeled data to find patterns or unusual behavior without knowing what a bug is beforehand.

Advantages: It can discover new types of bugs and does not need labeled data.

Challenges: It may incorrectly identify issues since it lacks specific guidance on what constitutes a bug, making results harder to understand.

Q3: Why is bias mitigation critical when using AI for user experience personalization?

Fairness: AI can unintentionally favor certain groups over others, so reducing bias ensures all users are treated equally.

User Trust: If users think an AI system is biased, they may not trust it. Trust is vital for user satisfaction.

Inclusivity: Personalization should meet diverse user needs. Addressing bias helps AI support a wider variety of preferences.

Legal and Ethical Responsibility: Many laws require fairness in AI. By reducing bias, organizations can comply with these regulations and act ethically.

2. Case Study Analysis

Read the article: *AI in DevOps: Automating Deployment Pipelines.*

Answer: How does AIOps improve software deployment efficiency? Provide two examples.

Predictive Problem Detection:

AIOps tools can analyze historical data to foresee potential issues before they occur. For example, if a specific deployment pattern has previously caused failures, the system can alert the team to change their approach, preventing downtime and ensuring smoother releases.

Automated Rollbacks:

AIOps can automatically revert deployments that fail. For instance, tools like Harness use AI to detect when a deployment does not succeed and roll back to the previous version without requiring human intervention. This minimizes downtime and allows teams to focus on resolving issues rather than managing failed deployments.

Part 3 – Ethical Reflection

1. Possible Biases in the Model

- **Skewed Priority Labels:** In our dataset, only about 12% of the tickets were tagged as “Medium” priority. Because of that, the model tends to overlook this group. In the real world, this might mean certain issues like performance problems aren’t flagged quickly enough simply because they’re less common in the training data.
- **Using the Wrong Clues:** We based ticket priority entirely on a number called radius_mean. This is like judging ticket urgency by how many lines of code were changed or who submitted it. For example, tickets from junior devs might unfairly be seen as less important.
- **Missing Pieces:** If we left out certain areas of the system, like older software components or teams in different locations, then the model won’t learn about them properly. That leads to them getting overlooked when issues arise later.

2. Fixing Biases with IBM’s AI Fairness 360 Toolkit

- **Spotting Bias**: The toolkit can help us check if “Medium” priority tickets or ones from lesser-known teams are often being misclassified by mistake.
- **Balancing the Scales Before Training**: We can use a method called Reweighing to give more attention to the underrepresented tickets during training. That way, the model learns to take them more seriously.
- **Building Fairness Into Training:** By tweaking the training process itself, we can make sure that the model doesn’t let teams guess a ticket’s priority just by knowing which group submitted it. That keeps predictions fair across the board.
- **Adjusting Predictions After the Fact**: Even after training, we can adjust how the model makes decisions, so all groups have an equal shot at getting accurate results, regardless of team or priority level.