Q1. AI code completion reduces development time by:

- Suggesting code snippets and boilerplate (functions, loops, tests) so a developer writes less repetitive code.
- Auto-completes API calls and common patterns, speeding up prototyping.
- Provides examples for unfamiliar libraries thus reduces context switching.

Limitations

- Suggestions can be incorrect or insecure and must be reviewed.
- It may encourage copy-paste without understanding the technical debt.
- Can leak copyrighted code or reveal undesirable patterns.
- Not good at high-level design or domain-specific correctness, it works best as an assistant.

Q2. Supervised vs Unsupervised learning for automated bug detection

- **Supervised learning**: trained on labeled examples (buggy vs non-buggy code). Its better when a developer has historical labeled bugs. Works well for classification as there is high precision when labels are reliable.
- **Unsupervised learning**: developers detects anomalies (unusual code or metrics) without labels, thus enables them to find unknown/new bug type however there is more false positives and requires human triage.

Q3. Why bias mitigation is critical for UX personalization

- Personalization adapts application behavior per user. If training data underrepresents groups, personalization can exclude users or give them poorer experience. It also reinforces stereotypes.
- Mitigation ensures fairness, legal compliance, and that personalization benefits all users.

How AIOps improves deployment efficiency

1. **Automated failure diagnosis** - AIOps correlates logs & metrics and suggests root causes, speeding recovery and reducing downtime.
2. **Auto-scaling and rollout decisions** - ML models predict load and trigger safe canary rollouts or scale resources automatically.

Two examples:

- Use anomaly detection to halt an automated deployment if error rate spikes during canary.
- Use predictive capacity planning to provision servers before peak traffic.

Remarks

Both AI-suggested and manual implementations use Python's which is the best we can do for a full sort in general. The AI version is concise and works well when the key exists in all dictionaries. The manual version adds explicit handling for missing keys and returns a stable, predictable order, dictionaries lacking the sorting key are treated consistently.

In production code, robust handling matters, the manual function avoids the type error when the key is absent and documents the behavior. From an efficiency standpoint, both are same complexity, in terms of safety and readability the manual version wins. The AI suggestion is very useful for quick prototyping, but I would prefer the manual version for production because it documents behavior and handles edge cases.

Screenshots for codes

```python
# AI-suggested version (simulated)
def sort_dicts_ai(dicts, key_name, reverse=False):
    """AI-suggested: simple sorted() usage"""
    return sorted(dicts, key=lambda d: d.get(key_name, None), reverse=reverse)

# Manual version (with error handling)
def sort_dicts_manual(dicts, key_name, reverse=False):
    """Manual version: handles missing keys"""
    def key_func(d):
        val = d.get(key_name)
        return (0, val) if val is not None else (1, None)
    return sorted(dicts, key=key_func, reverse=reverse)

# Test data
students = [
    {"name": "Alice", "age": 22},
    {"name": "Bob", "age": 19},
    {"name": "Clara", "age": 24},
    {"name": "Dan"}  # missing key
]

print("AI Sorted:", sort_dicts_ai(students, "age"))
print("Manual Sorted:", sort_dicts_manual(students, "age"))
```

```python
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.991
F1-score: 0.991

Classification Report:
              precision    recall  f1-score   support

        High       1.00      1.00      1.00        38
         Low       1.00      0.97      0.99        38
      Medium       0.97      1.00      0.99        38

    accuracy                           0.99       114
   macro avg       0.99      0.99      0.99       114
weighted avg       0.99      0.99      0.99       114
```