Part 3: Ethical Reflection (10%)

Prompt:

Your predictive model from Task 3 is deployed in a company. Discuss:

1. **Potential biases in the dataset** (e.g., underrepresented teams).
2. **How fairness tools like IBM AI Fairness 360 could address these biases.**

Answer:

When deploying predictive models such as the one built in Task 3 in a real-world software engineering context (e.g., to classify the priority of reported issues), it is crucial to examine ethical considerations—especially bias and fairness.

---

1. Potential Biases in the Dataset

Even though the model is trained on a medical dataset for this simulation, in a real company setting, an analogous dataset might include issue reports filed by different departments, product teams, or developer roles. Several bias risks arise:

- **Underrepresentation of Certain Teams:** Some departments (e.g., QA or support) may raise fewer tickets than core development teams, resulting in data imbalance. The model may learn to deprioritize issues from underrepresented groups.
- **Severity Mislabeling:** Historical labels could reflect human bias—perhaps bugs reported by junior engineers are marked as low priority, even when technically severe.
- **Feature Bias:** If features include metadata like "author role" or "team name," the model may overfit to patterns of bias present in the organization's history.

These biases can lead to unequal treatment, where critical bugs raised by certain teams are ignored or delayed, undermining product quality and team morale.

---

2. How Fairness Tools Can Help (e.g., IBM AI Fairness 360)

IBM AI Fairness 360 (AIF360) is an open-source Python toolkit designed to detect and mitigate bias in machine learning models.

Key Capabilities:

- **Bias Detection:** Provides metrics like disparate impact, statistical parity difference, equal opportunity difference, etc., across demographic groups or departments.
- **Bias Mitigation Techniques:**
    - *Pre-processing:* Rebalance or reweight training data.
    - *In-processing:* Apply fairness constraints during training.
    - *Post-processing:* Adjust predictions to reduce bias after model training.

Application in Context:

If the issue prioritization model disproportionately predicts "Low" for one department, AIF360 can detect the disparity and apply a technique like *Reweighing* or *Reject Option Classification* to mitigate it.

Conclusion:

Ethical AI development requires vigilance against bias and proactive use of fairness tools. Models in software engineering should be regularly audited and refined to ensure equitable treatment for all contributors, regardless of role, team, or location.

Bonus Task: Innovation Challenge (Extra 10%)

Proposed Tool Name: AutoDocGen – AI-Powered Documentation Assistant

---

Problem:

Software teams often struggle to maintain up-to-date documentation. Developers prioritize code delivery over documentation, leading to knowledge loss, poor onboarding experiences, and technical debt.

Solution: AutoDocGen

AutoDocGen is an AI tool that automatically generates and maintains technical documentation from source code, user stories, and version control history using Natural Language Processing (NLP) and Large Language Models (LLMs).

Key Features:

1. **Function/Class Summarization:**
    - Parses code comments and logic to auto-generate docstrings and technical summaries.
2. **Automated README Generation:**
    - Based on project structure, dependencies, and main files.
3. **Architecture Diagrams:**
    - Uses static analysis and graph generation to produce flowcharts or UML diagrams.
4. **Changelog Compilation:**
    - Summarizes Git commits and pull requests to auto-create release notes.
5. **Live Sync with CI/CD:**
    - Automatically updates documentation after every major commit or deployment.

Workflow:

1. Developer pushes code to Git.
2. AutoDocGen scans the codebase and extracts structural, functional, and semantic metadata.
3. An LLM processes this data to produce natural language documentation.
4. Output is published to a documentation portal or Markdown repository.

Expected Impact:

- Reduces documentation debt by up to 80%.
- Enhances onboarding for new developers.
- Promotes knowledge sharing and better project understanding.
- Saves engineering time that would otherwise be spent writing docs manually.