Problem Statements for Intel® Unnati Industrial Training 2025

Problem Statement 1

Bug Detection and Fixing

Build a machine learning model that can automatically identify bugs (or potential errors) in a given piece of code and suggest fixes. This project requires designing, training, and evaluating an ML model that can parse source code, classify bug types, and generate fix recommendations.

Key Objectives

1. **Data Extraction & Preparation**:
   - Collect code snippets from various programming languages (at least one or two languages to start with).
   - Label them as “buggy” or “bug-free” and, if buggy, include the correct “fixed” version of the code.
2. **Model Architecture**:
   - Build or fine-tune an existing model (e.g., Transformer-based, BERT-like, or an LLM-based model) to detect code smells or errors.
   - Incorporate a suggestion system that either completes or rectifies the code.
3. **Evaluation & Testing**:
   - Use standard metrics such as precision, recall, F1-score for bug detection.
   - Measure the accuracy and relevance of the fixes suggested.

Prerequisites

- Proficiency in Python.
- Understanding of Machine Learning (particularly natural language processing and code analysis models).
- Basic knowledge of software debugging and version control (to handle various versions of code).

Challenges

- **Data Requirements**:
  - The main challenge is assembling a sufficiently large and diverse dataset of buggy and fixed code.
  - Quality of labeled data is crucial: incorrect labels or partial fixes will degrade model performance.

- **Model Complexity**:
    - Code is more structured than natural language. Parsing abstract syntax trees (ASTs) or using token-based approaches needs careful design.
- **Overfitting & Generalization**:
    - Ensuring the model generalizes to unseen code from different domains or libraries.

Expected Outcome

- **Bug Detection**: A model or pipeline that, given a code snippet, highlights potential bugs.
- **Fix Recommendation**: The system proposes a fix or correction for the detected bug.
- **Metrics & Reporting**: A dashboard or report showcasing how the model performs on various types of bugs.
- **Suggested Tools/Technologies**:
    - **Python** for main development.
    - **ML Frameworks** such as TensorFlow, PyTorch, or Scikit-learn.
    - **Data Storage** can be done using SQL or NoSQL databases.
    - **Version Control** with Git/GitHub for collaboration.
    - **Deployment & Testing**: Containerization (Docker) or cloud platforms (AWS, Azure, GCP) for hosting the final model.
- **Collaboration & Project Management**:
    - Use Agile methodology or a similar iterative approach.
    - Regular checkpoints/sprints to track progress and adjust scope.
- **Evaluation Criteria**:
    - **Technical Accuracy**: Correctness and quality of the ML models.
    - **Innovativeness**: Novel approaches to data handling, model architecture, or user experience.
    - **Scalability & Robustness**: How well the solution can handle diverse or increasing data loads.
    - **User Experience**: The intuitiveness and utility of the final system from a learner’s or developer’s perspective.

Problem Statement 2