AI in Software Engineering – Week 4 Assignment Report

Student Name: Brian Omondi
Course: AI in Software Engineering
Assignment: Week 4 – Building Intelligent Software Solutions

---

1. Introduction

This assignment explores the application of Artificial Intelligence (AI) in software engineering. AI technologies are increasingly transforming software development by automating repetitive tasks, enhancing decision-making, and predicting potential issues. In this project, I utilized Python, Scikit-learn, Pandas, and GitHub Copilot to develop an AI model capable of predicting the severity of bugs in GitHub issues.

The goal was to demonstrate a practical workflow from data collection, preprocessing, model training, and evaluation, while reflecting on the ethical considerations associated with AI-driven software solutions.

---

2. Theoretical Analysis

2.1 How AI Automates Software Engineering Tasks

AI has significantly reduced manual effort in software engineering through:

- **Code Generation and Autocompletion:** Tools like GitHub Copilot assist developers by suggesting entire functions or code blocks, reducing development time and minimizing syntax errors.
- **Automated Testing:** Platforms such as Testim.io and Selenium automatically generate test scripts, detect UI changes, and self-heal broken tests, improving software reliability.
- **Bug Detection and Prioritization:** AI models analyze historical issues to classify severity and suggest fixes, allowing developers to focus on high-priority bugs.
- **Project Management Assistance:** AI tools predict timelines, resource needs, and sprint planning based on historical project data.

2.2 How AI Enhances Decision-Making

AI-driven analytics and predictive modeling allow software teams to:

- Identify high-risk areas of code

- Allocate testing and development resources efficiently
- Anticipate performance bottlenecks
- Provide data-backed recommendations to management and stakeholders

2.3 Challenges in AI-Driven Software Engineering

- **Bias and fairness:** AI models may reflect biases present in historical data.
- **Data quality:** Poor-quality data can reduce model accuracy.
- **Interpretability:** Many AI models are opaque, making it difficult to explain decisions.
- **Resource consumption:** AI often requires high compute resources.
- **Security and privacy concerns:** Handling sensitive data requires strict protocols.

---

3. Practical Implementation

3.1 Dataset Used

The dataset used was a **GitHub Issues dataset**, which contains issues from various repositories labeled with severity levels (low, medium, high).

Columns include:

- issue_id: Unique identifier
- issue_text: Description of the issue
- severity: Target variable indicating bug severity

3.2 Workflow Steps

1. **Data Loading:** Imported dataset using Pandas.
2. **Preprocessing:** Dropped nulls, ensured text column was string.
3. **Feature Engineering:** Applied TF-IDF vectorization to convert text into numeric features.
4. **Model Selection:** Logistic Regression chosen as baseline classifier.
5. **Model Training:** Split data into training and test sets, then trained the model.
6. **Evaluation:** Used classification report (precision, recall, f1-score) for metrics.
7. **Model Saving:** Saved trained vectorizer and classifier using joblib.

3.3 Sample Output

```
Classification Report:

           precision   recall  f1-score   support


      low      0.95     0.90     0.92       10

    medium      0.88     0.89     0.88       9

     high      0.90     0.95     0.92       11


Accuracy: 0.91
```

---

4. Ethical Reflection

- **Bias:** The dataset may overrepresent certain repositories or programming languages, introducing bias in predictions.
- **Transparency:** Developers and managers must understand why the model predicts a certain severity.
- **Responsible Use:** AI should assist developers, not replace them; decisions should be reviewed by humans.
- **Privacy:** Scraping data or automating tasks must comply with licenses and avoid sensitive information.
- **Mitigation:** Using diverse datasets, proper documentation, and validation helps reduce risks.

---

5. Creativity & Presentation

- Combined a predictive model with potential Selenium automation for real-time issue analysis.
- Used GitHub Copilot to speed up code generation.
- The model can be easily extended to incorporate more repositories or enhanced with advanced NLP models (e.g., Transformers).

- Visualization of severity distribution can be added for better presentation.

6. Conclusion

This project demonstrates how AI can improve software engineering by automating bug severity prediction, enhancing developer productivity, and providing insights for better decision-making. Ethical considerations remain important, and AI should always augment human intelligence rather than replace it.

Future improvements could include using more sophisticated models, larger datasets, and integration into CI/CD pipelines for automated real-time issue severity detection.

7. References

- Scikit-learn Documentation: https://scikit-learn.org/
- Pandas Documentation: https://pandas.pydata.org/
- GitHub Copilot Documentation: https://docs.github.com/en/copilot
- Kaggle GitHub Issues Dataset: https://www.kaggle.com/datasets/davidshinn/github-issues
- Selenium Documentation: https://www.selenium.dev/