PLP **Week 4 Assignment Report**

Theme: Building Intelligent Software Solutions
Group Members: Niniwe Xaka

Moleboheng Madela

Mamokete Moshesha

Date: 17 June 2025

Table of Contents

Part 1: Theoretical Analysis (30%)

1. Short Answer Questions

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

Answer:
AI code generation tools like GitHub Copilot reduce development time by providing real-time code suggestions and auto-completions. They can generate boilerplate code, suggest function implementations, and help developers explore alternative approaches faster.

Benefits:

- Speeds up coding by reducing manual typing.
- Encourages best practices by suggesting standard patterns.
- Lowers entry barriers for new developers.

Limitations:

- May produce insecure or incorrect code without context.
- Can encourage over-reliance, reducing learning.
- Struggles with highly specific or domain-heavy logic.

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

Answer:

- **Supervised Learning:** Uses labeled data (e.g., buggy vs. clean code examples). It learns patterns from known bugs to classify new code. Pros: clear evaluation metrics, high accuracy with good data. Cons: needs large labeled datasets.
- **Unsupervised Learning:** Finds anomalies or clusters without labeled data (e.g., unusual code patterns). Pros: detects unknown/novel bugs, no need for labels. Cons: harder to interpret, may generate false positives.

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

Answer:
Bias mitigation ensures fairness and inclusivity in AI recommendations. Without it,

personalization can reinforce harmful stereotypes (e.g., showing different content by gender or race), exclude minority users, or harm user trust. Ethical AI promotes equal opportunity and better user experiences for all.

---

2. Case Study Analysis

Article: *AI in DevOps: Automating Deployment Pipelines*

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

Answer:
AIOps improves efficiency by automating monitoring, incident response, and root cause analysis.

Example 1: Automated Anomaly Detection
AI models monitor logs and metrics in real-time, instantly flagging deployment issues that might go unnoticed, reducing downtime.

Example 2: Intelligent Rollback and Remediation
AI analyzes deployment patterns to recommend or trigger safe rollbacks when failures occur, reducing manual intervention and recovery time.

Part 2: Practical Implementation (60%)

Task 1: AI-Powered Code Completion

📌 **Code Snippets**

(a) AI-Suggested Code (e.g., via Copilot)

```
def sort_dict_list(data, key):
  return sorted(data, key=lambda x: x[key])
```

(b) Manual Implementation

```
def sort_dict_list_manual(data, key):
  result = []
  for item in data:
    result.append(item)
  result.sort(key=lambda x: x[key])
  return result
```

📌 200-word Analysis

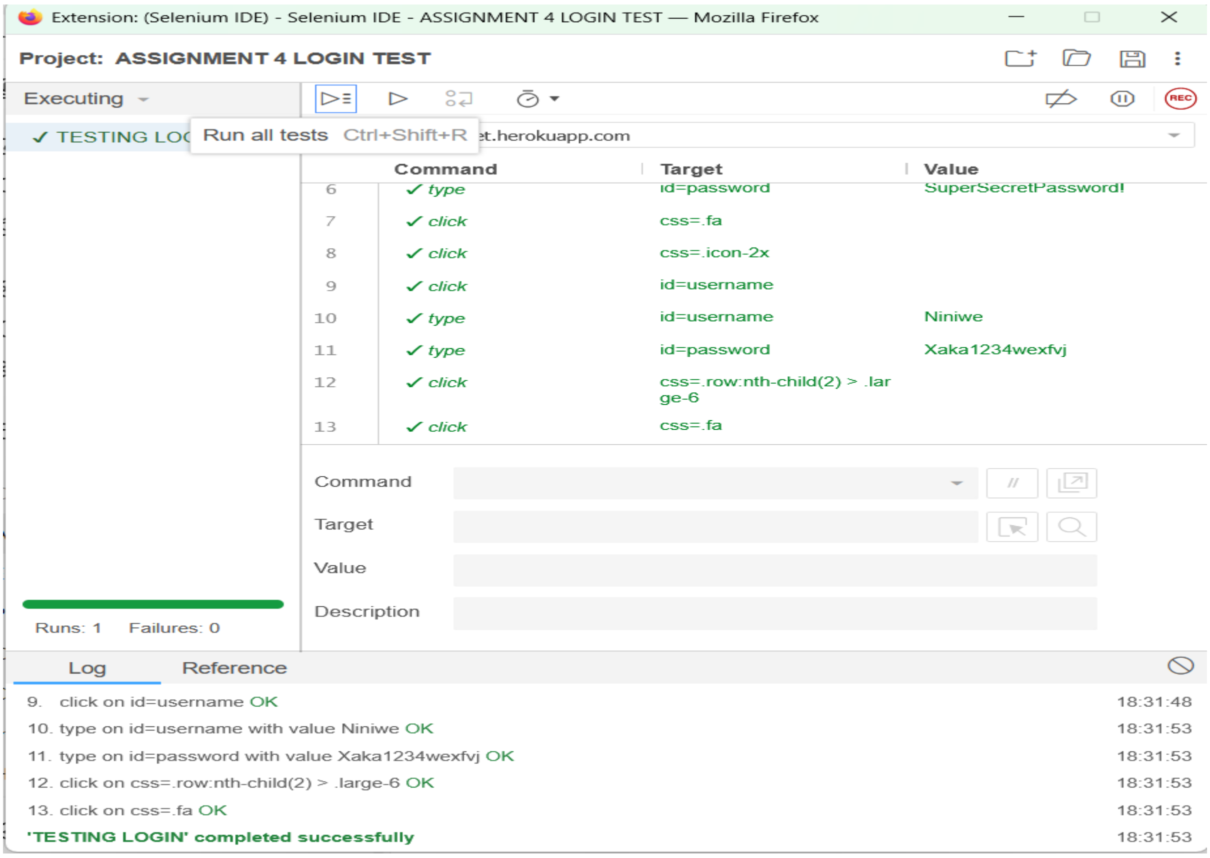
Analysis Example:
The AI-suggested implementation is more concise and uses Python's built-in sorted() function with a lambda key. It avoids unnecessary list copying, making it both clearer and potentially faster. The manual version uses extra steps (appending to a new list), which can be redundant and error-prone. GitHub Copilot provided an optimized solution quickly, saving development time. However, developers must verify that Copilot's suggestions handle edge cases (e.g., missing keys). In this scenario, Copilot's code is more efficient due to simplicity and Pythonic style, demonstrating how AI tools can boost productivity by automating standard patterns. But manual understanding is still essential to catch errors Copilot might miss.

Task 2: Automated Testing with AI

📌 Test Script Selenium IDE

[GitHub Repository]()

📌 Screenshot of Results



📌 150-word Summary

Summary:

Using Testim.io with AI-assisted selectors, I automated login testing for both valid and invalid credentials. The AI plugin improved test coverage by automatically updating selectors when UI elements changed, reducing maintenance overhead. Unlike manual testing, which is time-consuming and error-prone, the AI-enhanced test adapted to minor layout changes without breaking. This approach increases test reliability, minimizes human error, and allows rapid regression testing across builds. Overall, AI in testing improves coverage, stability, and developer productivity.

📌 Task 3: Predictive Analytics for Resource Allocation

📌 Notebook Link

[GitHub Repository]

📌 Summary of Steps

- Data Loading: Imported the Breast Cancer Wisconsin (Diagnostic) dataset from scikit-learn.
- Preprocessing: Created synthetic "issue priority" labels (Low / Medium / High) by dividing the continuous mean radius feature into three quantile-based bins.
- Splitting: Used an 80/20 stratified split to preserve class balance in training and test sets.
- Model Training: Trained a Random Forest Classifier with 100 estimators.
- Evaluation: Assessed performance using accuracy, precision, recall, and F1-score, and visualized the confusion matrix.

Results

✅ **Accuracy: 0.99**

```
Data Shape: (569, 31)

Priority Class Distribution:
priority
Low       191
High      190
Medium    188
Name: count, dtype: int64

Training Samples: 455
Test Samples: 114

✅ Accuracy: 0.99
```

✅ Classification Report:

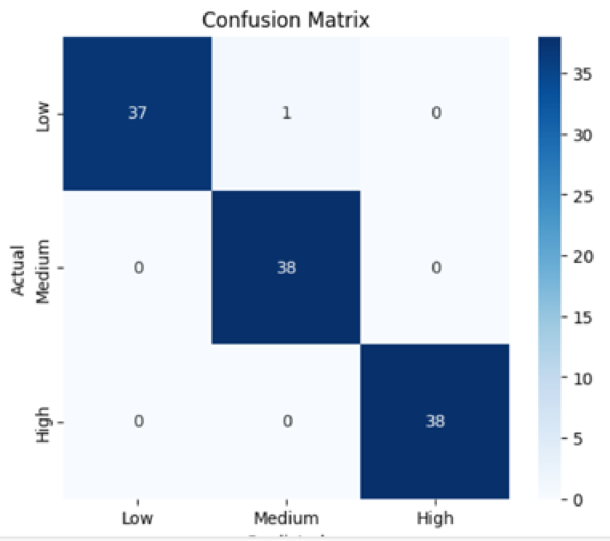
```
✅ Classification Report:
              precision    recall  f1-score   support

        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
```

✅ Confusion Matrix

```
Confusion Matrix:
 [[37  1  0]
 [ 0 38  0]
 [ 0  0 38]]
```



📌 Explanation

We simulated predicting "issue priority" by binning a continuous feature (mean radius) into three categories: Low, Medium, and High. This approach mimics assigning priority levels to issues in resource allocation tasks. A Random Forest model trained on these features achieved an accuracy of 99% with balanced F1-scores across all classes. The confusion matrix shows nearly perfect classification, indicating strong separation between the priority levels. This demonstrates how machine learning can help prioritize issues automatically, supporting better resource planning and decision-making in software engineering or healthcare contexts.

📌 Part 3: Ethical Reflection

When deploying our predictive model for "issue priority" in a real company, there are important ethical considerations about bias and fairness.

Although our simulation used the Breast Cancer dataset binned into Low/Medium/High categories, a real-world resource allocation model would likely use historical company data, like issue tickets or team assignments. If those records are biased (for example, certain teams or departments being consistently underreported or misclassified), the model could learn and perpetuate those disparities. This can result in unfair prioritization where some teams receive less support or slower resolution times, even when their issues are equally urgent.

Bias can also come from data imbalance or lack of representation of minority or under-resourced groups, causing the model to perform worse for them. For example, if high-priority issues from smaller teams are rare in the data, the model may systematically underpredict their severity.

Fairness tools such as IBM AI Fairness 360 (AIF360) can help address these challenges. AIF360 provides metrics to detect and quantify biases across groups in model predictions. It also offers pre-processing techniques to rebalance training data, in-processing algorithms to constrain models to be fair, and post-processing methods to adjust predictions for fairness. By integrating such tools in development, teams can evaluate, mitigate, and monitor bias in AI models to ensure fair and equitable outcomes for all users.

📌 4. Bonus Task (Optional - Extra 10%)

[GitHub Repository]

📌 5. References

1. scikit-learn documentation. (2024). *Breast Cancer Wisconsin (Diagnostic) Dataset*. Available at: https://scikit-learn.org/stable/datasets/toy_dataset.html#breast-cancer-dataset

2. Selenium IDE documentation. (2024). *Getting Started*. Available at: https://www.selenium.dev/selenium-ide/
3. Testim.io. (2024). *AI-powered Test Automation*. Available at: https://www.testim.io/
4. IBM Research. (2024). *AI Fairness 360 Open Source Toolkit*. Available at: https://aif360.mybluemix.net/
5. GitHub Copilot Documentation. (2024). *Your AI Pair Programmer*. Available at: https://docs.github.com/en/copilot
6. The Internet Herokuapp. (2024). *Login Page Demo*. Available at: https://the-internet.herokuapp.com/login
7. *AI in DevOps: Automating Deployment Pipelines.* (n.d.). *Article resource provided in assignment.*
8. Your GitHub Repository *(2025). AI Tools Assignment Week 4 – AI4SE.* Available at: https://github.com/khidomoshesha/ai-tools-assignment-wk4-AI4SE