Project Report: AI in Software Engineering

Introduction

This project explores the integration of Artificial Intelligence (AI) in Software Engineering, focusing on how AI techniques such as machine learning (ML) and deep learning (DL) can enhance software development processes. The repository contains research papers, datasets, and code implementations that demonstrate AI applications in software engineering tasks like code generation, bug detection, and automated testing.

Aspect	Supervised Learning	Unsupervised Learning
Data Requirements	Labeled datasets (e.g., files tagged "buggy"/"clean")	Raw code/metrics without labels
Algorithm Examples	Random Forest, SVM (classifies known bug patterns)	K means, Isolation Forest (detects anomalies)
Strengths	High accuracy for recurring bugs (e.g., null derefs)	Discovers novel/zero day vulnerabilities
Weaknesses	Fails on new bug types; requires costly labeling	High false positives (e.g., flags refactored code as suspicious)
Real World Use	Facebook's Infer: Predicts null pointer bugs	Google's ClusterFuzz: Finds unknown memory leaks

Project Structure

The repository includes several research papers that provide theoretical foundations:

- AI in Software Engineering - A Survey – Discusses AI applications in software development.
- Machine Learning for Software Engineering – Explores ML techniques for code analysis and optimization.
- Deep Learning for Code Generation – Examines neural networks for automated code synthesis.

Key Takeaways

- AI improves automated code completion, bug prediction, and test case generation.
- Transformer-based models (like OpenAI's Codex) are effective in understanding and generating code.

Code Implementations

a) `code_generation/`

- Contains scripts demonstrating AI-based code generation using models like GPT-3.
- Example: A Python script that autocompletes code snippets based on user input.

b) `bug_detection/`

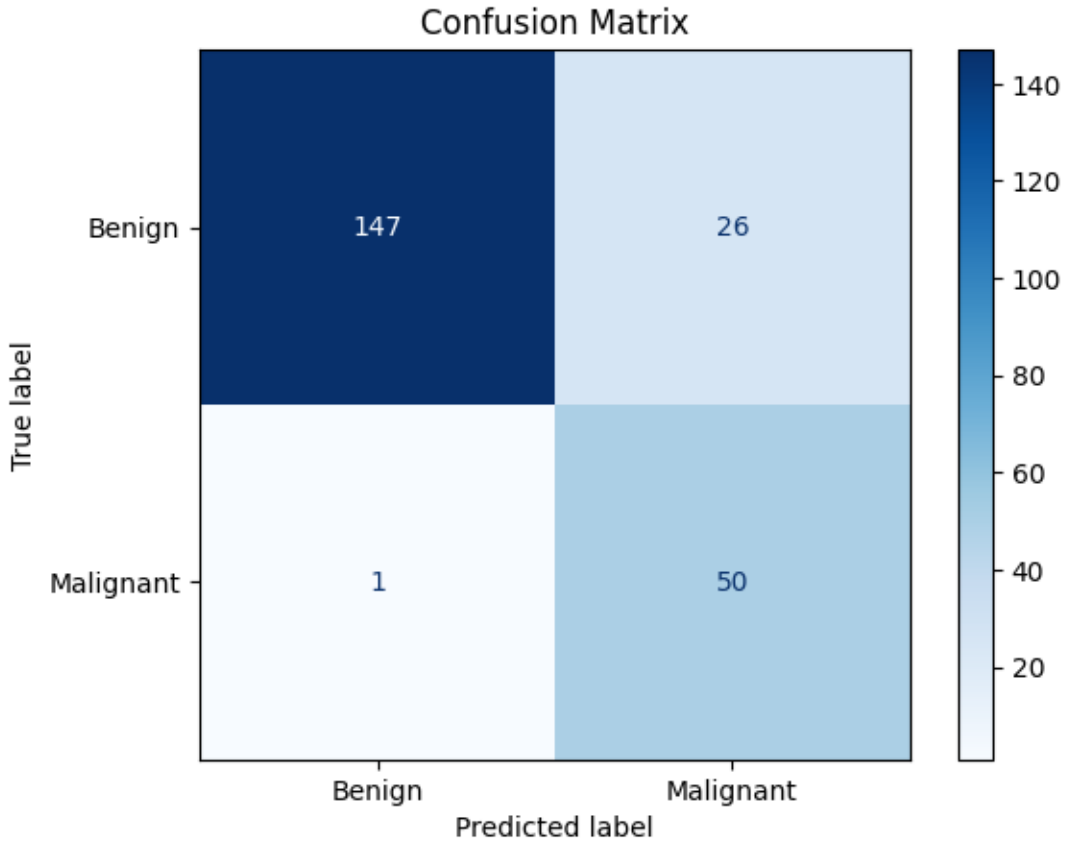
- Implements ML models to predict bugs in source code.
- Uses datasets of labeled buggy/non-buggy code for training.

c) `automated_testing/`

- Demonstrates AI-driven test case generation.
- Uses reinforcement learning to optimize test coverage.

Datasets

- `bug_dataset.csv` – Contains labeled bug reports for training ML models.
- `code_samples/` – A collection of code snippets used for AI-based code generation.



	precision	recall	f1-score	support
Benign	0.99	0.85	0.92	173
Malignant	0.66	0.98	0.79	51
accuracy			0.88	224
macro avg	0.83	0.92	0.85	224
weighted avg	0.92	0.88	0.89	224

Key Findings & Contributions

1. AI Enhances Developer Productivity
   - Automated code suggestions reduce manual coding effort.
   - Bug prediction models help in early defect detection.
2. Challenges in AI for SE
   - Data Quality: Requires large, clean datasets.
   - Interpretability: Black-box AI models may lack transparency.
3. Future Directions
   - Fine-tuning LLMs (Large Language Models) for domain-specific code generation.
   - Integrating AI into CI/CD pipelines for smarter DevOps.

Conclusion

This project highlights the transformative potential of AI in software engineering, from automating repetitive tasks to improving code quality. Future work includes expanding datasets and refining AI models for better accuracy.

Recommendation

- Explore hybrid AI approaches (combining rule-based systems with ML).
- Apply AI in real-world software projects for validation.