AI can be used for improving existing processes of testing in SDLC. The idea is to make use of the data generated in test processes to provide detailed analytics, more automation, deeper insights and patterns and ability to predict and take corrective actions.

AI can assist and support various activities of the testing and development process such as:

- Test planning
- Creating tests and test data
- Impact analysis and selection of test for regression testing
- Test result analysis and defect prediction
- Predicting the component causing the bug and defect assignment
- Test automation

Indeed the applications of AI in the development and testing process are limitless.

7.1 AI for Software Testing Life Cycle (STLC)

7.1.1 AI for STLC Methods

Test Planning

AI can help in the test planning by assisting in various test planning activities:

Test estimation – Estimation of resources and time needed for a project

Risk analysis – Contribution factors, likelihood and impact

Also, to prioritize risks, obtain more accurate metrics associated with schedule adherence, and help identifying the performance metrics of applications more accurately.

In the context of test planning and estimation, overall software project cost can be approximated, taking into account different inputs related to the AI projects, including data size, effort involved, platform choice, application type, data preparation time, training time, and testing time. Given the historical data of these inputs, ML models can be prepared to give more accurate estimations.

Test Estimation and Risk Analysis

Estimation based on AI techniques identifies the most influencing factors.

Organizations have data related to:

- Changes, regression and field defects
- Static and dynamic analysis data
- Risk registers and analyses

- Factors contributing to the risk
- Possible probabilities of the risk becoming a reality
- Ability to meet the goals and milestones
- Performance analysis of teams, individuals

AI-Based Test Data Generation/Augmentation

Generating realistic test data is one of the critical problems in testing

- Production data anonymization/masking has its own challenges
- Synthetic data may miss many nuances of real-world data
- Image data is harder to produce

For test design, using AI technologies, like natural language processing (NLP) and text mining, can help the automated test case generation from textual requirements documents. Additionally, AI applied on code analysis (both static and dynamic), along with the analysis of the data collected from tests, can flag potential issues of performance and other non-functional requirements. Moreover, running ML on past data can help identifying test data patterns and helps generating automated test data for both component tests and system tests.

In particular, for image data and GUI elements, AI can help identify incorrectly rendered elements automatically. Additionally, by an ML based data centric analysis of different possible flows, the right flows of data can be automated.

AI-Based Defect Prediction

- Is done using the classification technique (e.g. Logistic Regression) to predict faults
- Output variable: A code piece will generate/non-generate a fault
- Input variable: Code Metrics
    - Weighted Methods per Class (WMC)
    - Coupling Between Objects (CBO)
    - Depth of Inheritance Tree (DIT)
    - Number of Children (NOC)
    - Response For a Class (RFC)
    - Lack of Cohesion in Methods (LCOM)

For automated defect prediction, models using ML can predict defects based on code quality metrics.

Impact analysis using ML on code can help automate the identification of impacted modules and files based on change.

In terms of coverage analysis using AI can help achieve comprehensive test and code coverage via the analysis of the data flows captured.

- Adversarial attacks are emerging security threat for AI
    - Social media systems poisoned by attacks that propagated low-quality content designed to bias the 'watch-list' recommendation algorithms
    - Injecting bad ('adversarial') data into the training data
    - Impersonation attacks to trick a model I not misidentifying

Overcoming Adversarial Attacks

- Adversarial training
    - Mix adversarial examples with normal input examples which training
    - Makes for robust detection in training phase
- Ensemble model by creating a model mixing multiple small models
- Defensive Distillation
    - Train the model to output probabilities of different classes, rather than taking a decision on putting only one class as output.

7.1.2 AI for Reporting and Smart Dashboards

In the context of reporting and dashboards, using ML helps to generate focused insights and summarized data for presentation in smart dashboards, instead of pure analytics.

- Dashboard can be made dynamic and smart by using AI
- Data driven ML algorithms extract intelligence by crunching input data and produce smart outputs
- This context sensitive intelligence can help building a personalized custom insight platform
- Smart interactivity, smart drilldowns, smart reports, smart analytics, smart infographics, smart displays are some key distinguishing features of a smart dashboard.

7.2 AI based automation tools

- Need for Tools in Test Lifecycle
- Test case design
- Test execution
- Test/Regression suite selection
- Test case maintenance
- Coverage analysis

7.2.1 Tools

use AI for making automation easier or more maintainable.

AI tools could use GUI Spiders which traverse the complete GUI and record the app. Over iterations, they are able to learn, compare and identify bugs.

Some tools combine elements from visual GUI tests and use ML to figure out the changes and possible correlation between locators and elements and whether a change is a bug or an expected change.

Some tools use NLP, some work at DOM level, some at UI, some go to the log, some combine functional and performance testing and some combine some of the above-mentioned approaches.

Image based tools may use AI based image classification tools to flag UI defects. These may be working as a web browser plug-in with record and playback. AI based image comparison can be superior to simple image comparison.

Types of Tools

Visual, automated UI testing

- Using image processing technology instead of pixel comparison
- Can check various UI attributes such as color, positioning, size, etc.
- Position testing can also reveal a problem of overlapping elements, etc.
- It can find differences that are likely to be missed by testers.

GUI Spiders

- Using ML, the tool traverses the complete GUI and records the app
- Over iterations, it is able to learn, compare and identify bugs
- They may analyze GUI, DOM, performance, etc.

API testing

- Tools learn the APIs, the parameter variation and also sequences, etc.
- Especially when there is no GUI to be tested

AI-Based Test Automation Tools

- Image based tools use AI base image classification tools to flag UI defects
- Autonomous test lifecycle management tools using AI can create test cases by learning the structures of programs
- Self-improving Test Suites perform self-correction by analyzing false positives
- Integrated requirements document analysis by NLP combined with domain knowledge can help generating test case
- Dynamic test prioritization using the knowledge learnt from experiences is now made feasible via reinforcement learning

testing by suggesting a minimal regression test selection set

Commercial AI Tools

- MABL
- Appvance
- Automomiq
- Test.ai
- Applitools
- Testim.io