How to use AI/ML Systems to Revolutionize Testing and Test automation?

Subhadeep Chakraborty

Abstract— Much of our day to day technology is powered by Artificial Intelligence - It is the study of agents those who perceive the world around them, form plans, and make decisions to achieve their goals. Machine Learning is a sub field under Artificial Intelligence. Its goal is to enable computers to learn on their own, without explicitly programming for each and everything. In this paper, we will specifically discuss "What does it mean to a testing community, how would it change the way we are testing?". Our approach on AI/ML based Quality Assurance in TechM is design based compliance with following steps – Discover > Learn > Sense > Respond cycle. The knowledge base constantly helps in storing and building pattern, which in turn helps self-learning and responding to actions.

Index Terms— Artificial Intelligence, Machine Learning, Software Testing, Predictive Analytics, Automation Testing, Software Quality, Risk Based Testing.

1 INTRODUCTION

Artificial intelligence – simply put, is the opposite to the natural intelligence shown/exercised by humans and animals. It is the study of agents those who perceive the world around them, form plans, and make decisions to achieve their goals. Its foundations include mathematics, logic, probability, linguistics, neuroscience, decision theory and even philosophy. Many fields fall under the umbrella of AI, such as computer vision, robotics, machine learning, and natural language processing. So, **Machine Learning** is a sub field under Artificial Intelligence. Its goal is to enable computers to learn on their own, without explicitly programming for each and everything. Machine learning algorithm identifies patterns in observed data, build models and explain it.

A summary of the 4 ways of how a machine (computer) learns has been given below, before we go into the details of how to use AI/ML Systems to revolutionize Testing and Test automation.

Supervised Learning:

In supervised learning problems, we start with a data set containing training examples with associated correct labels. For example, when learning to classify handwritten digits, a supervised learning algorithm takes thousands of pictures of handwritten digits along with labels containing the correct number each image represents. The algorithm will then learn the relationship between the images and their associated numbers and apply that learned relationship to classify completely new images (without labels) that the machine hasn't seen before. This is how we deposit a check by taking a picture with phone in bank app.

Let's take another example, where we want to predict the salary based on the college degrees. Here inputs will be sample/training data of college degrees, bachelors, masters, and various types of these and corresponding salaries. The learning algorithm will then predict the income based on a new data set given in the form of college degree.

Supervised learning works in two ways- **Regression** (not same as the testing terminology we are used to) and **Classification**.

In Regression way of function, the output is the form of a continued value and we are expecting the algorithm to estimate a continuous numerical value. How much will one earn, if one has this degree?

Classification, we want the algorithm to assign a label. Say the sample image given is a Truck or Car. Whether the borrower going to repay the loan or not? Classification predicts a discrete target value. Classification is the problem of assigning new observations to the class to which they most likely belong, based on a classification model built from labeled training data.

Unsupervised Learning:

The task here is to find underlying structure in data, to effectively represent the data in compressed format. There is not target value here. For example, and advertising platforms segments population into smaller groups with similar demographics and purchasing habits so that advertisers can reach their target market with relevant ads. Wonder what information we can get, feeding this model with heaps of defects identified in all stages of testing?

Neural network and deep learning:

this draws inspiration from the brain. The problem is approached by learning the layers of abstraction. Artificial neurons learn to detect abstract concept in the hidden layers. The example is autonomous vehicle learning dynamics of roads, signal, traffic, pedestrian etc.

Reinforcement Learning:

In reinforcement learning there is no answer key, but your reinforcement learning agent still has to decide how to act to perform its task. In the absence of existing training data, the agent learns from experience. It collects the training examples ("this action was good, that action was bad") through trial-and-error as it attempts its task, with the goal of maximizing long-term reward. The easiest context in which to think about reinforcement learning is in games with a clear objective and a point system.

2 Testing Software in the Age of Machine Learning

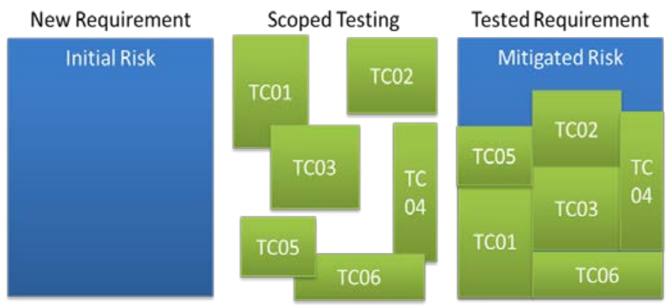
As the World Quality Report 17-18 as well as the State of Testing Survey 2017 outline, there are many trends underway in the testing industry: Development cycles will become even shorter, mobile and hybrid application testing will become priority number one and test automation will be dominating. Many experts agree on that machine learning will have a huge impact on software testing.

Machine learning applies artificial intelligence to provide systems the ability to automatically learn without human intervention or explicit programming. Systems and testing automation would improve from experience and would automatically access data, run tests with it and learn from the results and improve the testing cycle. Tools such as machine learning derive patterns from operations data and enable the analysis of huge amounts of data. It is expected to deliver more accurate results in less time and provide effective ways to test IoT solutions and many more upcoming technologies.

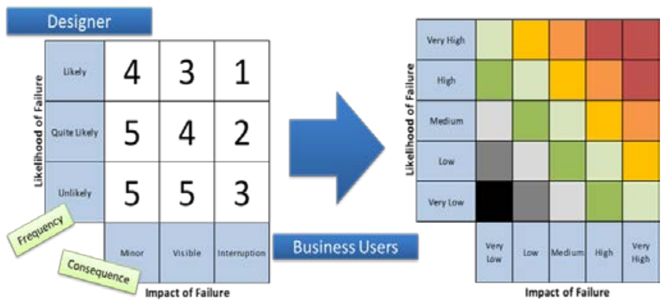
Most machine learning systems are based on **neural networks**. A neural network is a set of layered algorithms whose variables can be adjusted via a learning process. The learning process involves using known data inputs to create outputs that are then compared with known results. When the algorithms reflect the known results with the desired degree of accuracy, the coefficients are frozen and production code is generated. Today, this comprises much of what we understand as artificial intelligence.

By contrast, **predictive analytics** makes adjustments to the algorithms in production, based on results fed back into the software. In other words, the application better understands how to apply its rules based on how those rules have worked in the past.

Testing is about *understanding and mitigating the risk associated with a software delivery* and business may need this risk assessment to help them predict the expected success or failure of their requirements by the time of testing deadline or live launch date.



Risk has traditionally (and subjectively) been assessed through 3x3, 4x4 or even 5x5 RBT grids to predict how much risk may be mitigated for each individual test case

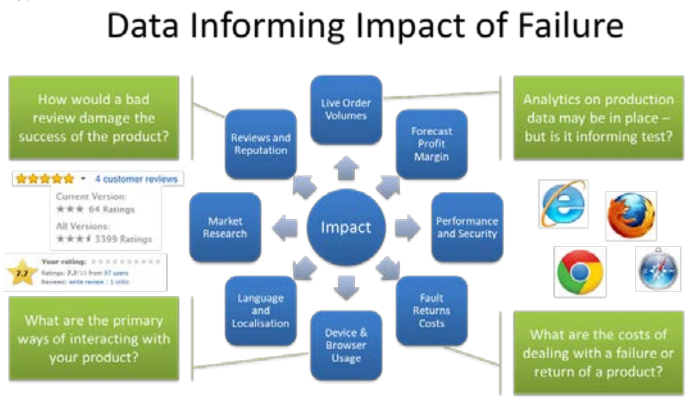


Predictive Analytics techniques can be applied to testing Risk assessment based on sound analytics provides deeper insights and can identify actions that will improve business outcomes.

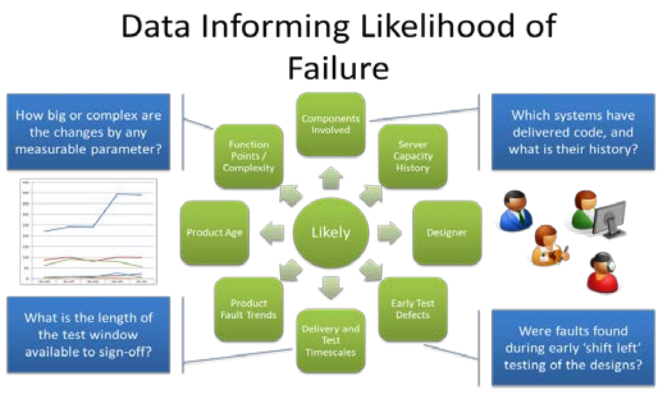
The approach is **Supervised Machine Learning**

- Identify your data sources
- Train the machine to identify correlations
- Review predictions
- Find algorithms with sufficient accuracy
- Apply the predictions to new data
- Indicate the expected prediction accuracy
- Learn and evolve

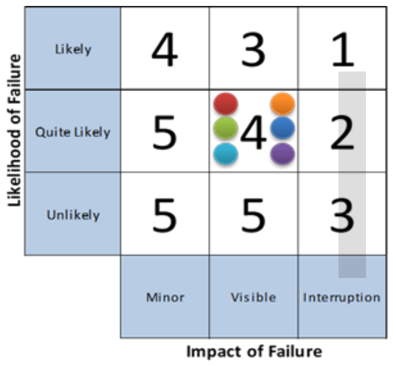
There are a number of Data Points which typically influences the **Impact of Failure** and **Likelihood of Failure** parameters.


Data Informing Impact of Failure
How would a bad review damage the success of the product?
Reviews and Reputation
Live Order Volumes
Forecast Profit Margin
Analytics on production data may be in place – but is it informing test?
4 customer reviews
Market Research
Impact
Performance and Security
Language and Localisation
Device & Browser Usage
Fault Returns Costs
What are the primary ways of interacting with your product?
What are the costs of dealing with a failure or return of a product?

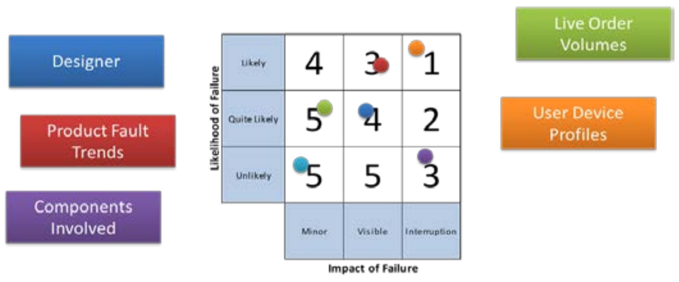
- *Subhadeep Chakraborty is currently Customer Delivery Executive in Tech Mahindra Ltd, India, PH-+919836270123. E-mail: subhc@techmahindra.com*


Data Informing Likelihood of Failure
How big or complex are the changes by any measurable parameter?
Which systems have delivered code, and what is their history?
What is the length of the test window available to sign-off?
Were faults found during early 'shift left' testing of the designs?
Components Involved
Function Points / Complexity
Server Capacity History
Product Age
Likely
Designer
Product Fault Trends
Delivery and Test Timescales
Early Test Defects

Supervised Machine Learning techniques can be used to train the machine to identify correlations between these data points and arrive at statistical predictions using algorithms with sufficient accuracy apply the predictions to new data, indicate the expected prediction accuracy and learn and evolve. For example, let us consider a business scenario where most of the tests have been categorized as "Quite Likely" to fail on a 3x3 Risk Based Testing grid based on traditional "subjective" Risk assessment.



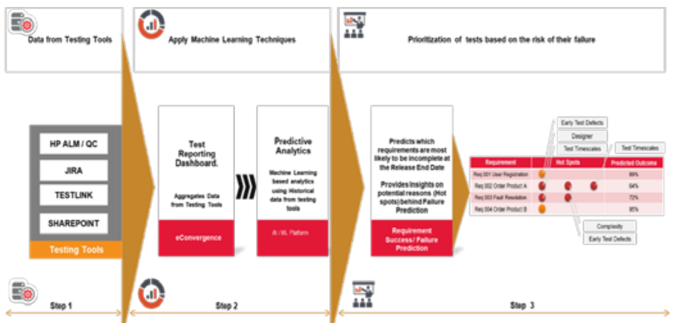
Once we apply Predictive Analytics to Risk Based Testing we will see that it has improved our understanding of risk and will give business more confidence. As we can see below in the example chosen, after applying predictive analytics the risk has been distributed across the grid in a more "objective" manner!

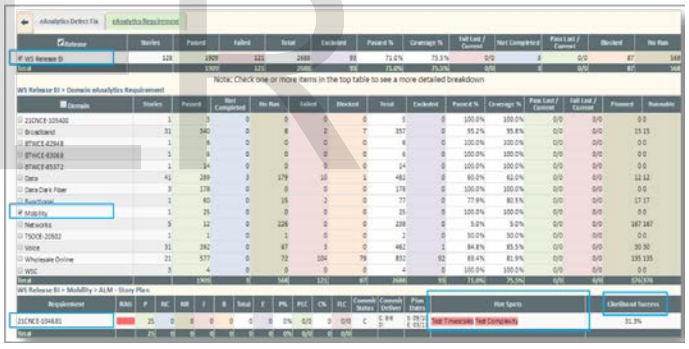


3 MACHINE LEARNING BASED TEST ANALYTICS IN TECHMAHINDRA

We have developed a Test Analytics platform that utilizes Machine Learning algorithm to derive actionable insights from data aggregated from testing tools. This feature is inbuilt in TechMahindra (hereafter referred to as TechM)'s proprietary tool **eConvergence**. It analyzes defects and predict the root cause and helps reduce test cycle time.

TechM's Test Analytics tool takes analytic feeds from test and defects data in eConvergence to arrive at **Defect Prediction Model** using machine learning algorithms. The tool was developed in line with the need for **TestOps** testing model to support continuous testing services in Dev-QA-Ops based delivery programs via machine learning algorithm. The tool is based on **Artificial Intelligence** and **Shift Right** Principles covering Test Analytics and has been designed to give insight into existing testing practices to derive foresight for Improving Business Outcomes and ensuring Brand Assurance for our customers.


Data from Testing Tools
Apply Machine Learning Techniques
Prioritization of tests based on the risk of their failure
HP ALM / QC
JIRA
TESTLINK
SHAREPOINT
Testing Tools
Test Reporting Dashboard
Aggregates Data from Testing Tools
eConvergence
Predictive Analytics
Machine Learning based analytics using Historical data from testing tools
AI / ML Platform
Predicts which requirements are most likely to be incomplete at the Release End Date
Provides insights on potential reasons (Hot spots) behind Failure Prediction
Requirement Success / Failure Prediction
Early Test Defects
Designer
Test Timescales
Complexity
Step 1
Step 2
Step 3



eAnalytics Defect Fix | eAnalytics Requirement

Release	Stories	Passed	Failed	Total	Excluded	Passed %	Coverage %	Fail Last / Current	Not Completed	Pass Last / Current	Blocked	No Run
WS Release BI	128	1909	121	2688	93	71.0%	73.5%	0/0	3	0/0	87	568
Total		1909	121	2688	93	71.0%	75.5%	0/0	3	0/0	87	568

Note: Check one or more items in the top table to see a more detailed breakdown

WS Release BI > Domain eAnalytics Requirement

Domain	Stories	Passed	Not Completed	No Run	Failed	Blocked	Total	Excluded	Passed %	Coverage %	Pass Last / Current	Fail Last / Current	Planned	Runnable
21CNCE-105400	1	5	0	0	0	0	5	0	100.0%	100.0%	0/0	0/0	0 0	
Broadband	31	340	0	8	2	7	357	0	95.2%	95.8%	0/0	0/0	15 15	
BTWCE-82948	1	6	0	0	0	0	6	0	100.0%	100.0%	0/0	0/0	0 0	
BTWCE-83068	1	6	0	0	0	0	6	0	100.0%	100.0%	0/0	0/0	0 0	
BTWCE-85372	1	14	0	0	0	0	14	0	100.0%	100.0%	0/0	0/0	0 0	
Data	41	289	3	179	10	1	482	0	60.0%	62.0%	0/0	0/0	12 12	
Data Dark Fiber	3	178	0	0	0	0	178	0	100.0%	100.0%	0/0	0/0	0 0	
Functional	1	60	0	15	2	0	77	0	77.9%	80.5%	0/0	0/0	17 17	
Mobility	1	25	0	0	0	0	25	0	100.0%	100.0%	0/0	0/0	0 0	
Networks	5	12	0	226	0	0	238	0	5.0%	5.0%	0/0	0/0	187 187	
TSOCE-20502	1	1	0	1	0	0	2	0	50.0%	50.0%	0/0	0/0	0 0	
Voice	31	392	0	67	3	0	462	1	84.8%	85.5%	0/0	0/0	30 30	
Wholesale Online	21	577	0	72	104	79	832	92	69.4%	81.9%	0/0	0/0	135 135	
WSC	3	4	0	0	0	0	4	0	100.0%	100.0%	0/0	0/0	0 0	
Total		1909	3	568	121	87	2688	93	71.0%	75.5%	0/0	0/0	376 376	

WS Release BI > Mobility > ALM - Story Plan

Requirement	RAG	P	NC	NR	F	B	Total	E	P%	P/C	C%	F/C	Commit Status	Commit Deliver	Plan Dates	Hot Spots	Likelihood Success
21CNCE-104681		25	0	0	0	0	0	0	0%	0/0	0	0/0	C	C: 8/6 D:	S: 09/1 E: 03/1	Test Timescales Test Complexity	31.3%
Total		25	0	0	0	0	0	0	0%	0/0	0	0/0					

A working demonstration of the tool can be watched here

The programming language which has been used is Python and the algorithms which have been used are from **Scikit-Learn** (http://scikit-learn.org/stable/)

The following use cases have been implemented in the tool –

Use Case #1: Defect Fix Date Prediction

- Scenario: Predict the expected Fix Date of a defect shortly after it has been raised
- Benefits: Enables testers and test leads to forecast when a defect will be fixed, allowing them to plan other work while waiting for the fix and/or to escalate to delivery/platform teams to accelerate fixes that will significantly disrupt critical path testing schedules

Use Case #2: Requirement Success/Failure Prediction

- Scenario: Predict the expected Success or Failure of a Requirement at a fixed Release End Date (plus identify weighted reasons behind Failure predic-

tion)
- Benefits: Enables delivery teams to review the Requirements most likely to be incomplete at the fixed Release End Date and the reasons why our algorithms expect they may fail so appropriate action can be taken - for example; if algorithms believe involvement of a particular designer is a significant contributing factor to a likely failure then additional design reviews could be schedule; if algorithms identify a particular combination of applications with a reduced test window is a factor to failure then additional time to test could be negotiated

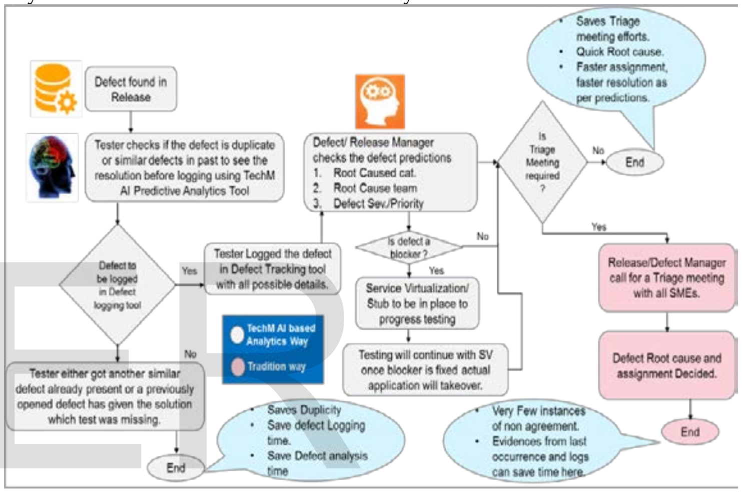
Use Case #3: Defect Volume and Root Cause Prediction
- Scenario: Predict the expected number of defects and likely causes (e.g. Design, Development, Interface etc) for new requirements
- Benefits: Using requirement, application, and defect data from previous releases we would forecast the likely defect volume and root causes of requirements; enabling testers and delivery to schedule additional reviews or walkthroughs of designs or interface specifications etc. to mitigate potential problems before testing commences

Use Case #4: Test Likelihood of Failure RBT Prediction
- Scenario: Predict the test case Likelihood of Failure RBT value
- Benefits: Using information including test steps, journey information, applications involved etc. we would forecast a more realistic and objective Likelihood of Failure parameter that could provide a more accurate Risk Based Testing assessment and Earned Confidence

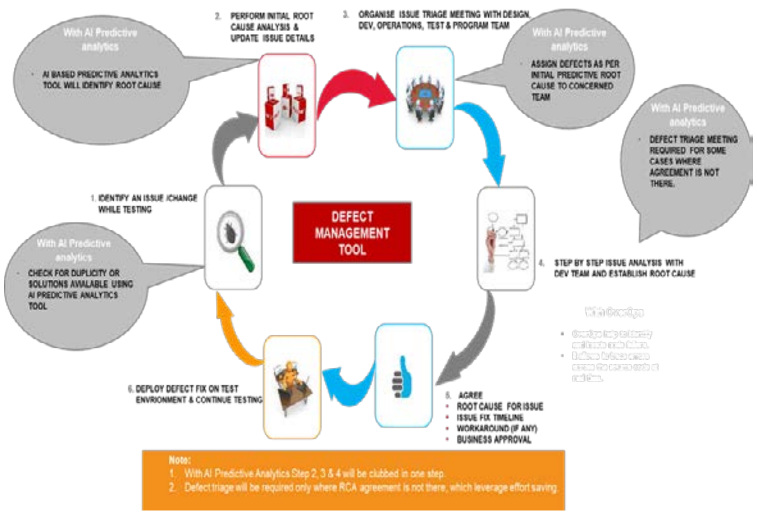
Use Case #5: Test Impact of Failure RBT Prediction
- Scenario: Predict the test case Impact of Failure RBT value
- Benefits: Using information including live order volumes, business benefits, order entry point popularity etc. we would forecast a more realistic and objective Impact of Failure parameter that could provide a more accurate Risk Based Testing assessment and Earned Confidence

4 AI BASED PREDICTIVE ANALYTICS IN DEFECT TRIAGE

TechM's proprietary **AI based Predictive Analytics tool (Cha.AI)** uses the historical data and AI algorithms to predict the root cause and team, this process is automated hence it leverages faster response as soon as the defect is logged, and scheduler of AI based predictive analytics will run all details which predicts the root cause of the defect and the same can be pushed back to Defect tracking tool.
As per current POC almost 70% of the cases are rightly predicted first time so it directly saves triage meeting efforts and time, only 30% cases may require a triage meeting in case of conflict but in those cases also the tool will help to provide previous occurrence and logs to prove how tool has predicted the root cause analysis.
One more feature of the tool allows tester to search the open and closed defects of similar nature to avoid duplicity and find the solution at the early stages which can be readily implemented, this makes defect process faster and saves efforts and quality of Defect management process.
Along with this, TechM integrates notification mechanism in case of a blocker defect to root cause applications to start Service Virtualization/Stubbing so that even when the testing is blocked, and team can still move on to progress till the time the blocker issue is fixed. Once the blocker is fixed the real interface will take over in place of SV/Stubbed interface.
The diagram below shows how TechM IP AI Based Predictive Analytics tool does the root cause analysis:


Defect found in Release
Tester checks if the defect is duplicate or similar defects in past to see the resolution before logging using TechM AI Predictive Analytics Tool
Defect/ Release Manager checks the defect predictions
1. Root Caused cat.
2. Root Cause team
3. Defect Sev./Priority
Is Triage Meeting required ?
No
Yes
End
Saves Triage meeting efforts.
Quick Root cause.
Faster assignment, faster resolution as per predictions.
Defect to be logged in Defect logging tool
Yes
No
Tester Logged the defect in Defect Tracking tool with all possible details.
Is defect a blocker?
Yes
No
Service Virtualization/ Stub to be in place to progress testing
Testing will continue with SV once blocker is fixed actual application will takeover.
TechM AI based Analytics Way
Tradition way
Tester either got another similar defect already present or a previously opened defect has given the solution which test was missing.
End
Saves Duplicity
Save defect Logging time.
Save Defect analysis time
Very Few instances of non agreement.
Evidences from last occurrence and logs can save time here.
Release/Defect Manager call for a Triage meeting with all SMEs.
Defect Root cause and assignment Decided.
End

Predictive analytics tools add intelligence to defect management process


With AI Predictive analytics
AI BASED PREDICTIVE ANALYTICS TOOL WILL IDENTIFY ROOT CAUSE
2. PERFORM INITIAL ROOT CAUSE ANALYSIS & UPDATE ISSUE DETAILS
3. ORGANISE ISSUE TRIAGE MEETING WITH DESIGN, DEV, OPERATIONS, TEST & PROGRAM TEAM
With AI Predictive analytics
ASSIGN DEFECTS AS PER INITIAL PREDICTIVE ROOT CAUSE TO CONCERNED TEAM
With AI Predictive analytics
DEFECT TRIAGE MEETING REQUIRED FOR SOME CASES WHERE AGREEMENT IS NOT THERE.
1. IDENTIFY AN ISSUE /CHANGE WHILE TESTING
DEFECT MANAGEMENT TOOL
With AI Predictive analytics
CHECK FOR DUPLICITY OR SOLUTIONS AVAILABLE USING AI PREDICTIVE ANALYTICS TOOL
4. STEP BY STEP ISSUE ANALYSIS WITH DEV TEAM AND ESTABLISH ROOT CAUSE
6. DEPLOY DEFECT FIX ON TEST ENVRIONMENT & CONTINUE TESTING
5. AGREE
ROOT CAUSE FOR ISSUE
ISSUE FIX TIMELINE
WORKAROUND (IF ANY)
BUSINESS APPROVAL
Note:
1. With AI Predictive Analytics Step 2, 3 & 4 will be clubbed in one step.
2. Defect triage will be required only where RCA agreement is not there, which leverage effort saving.

5 MACHINE LEARNING FOR AUTOMATION TESTING

By using Machine Learning for test automation, we can dynamically write new test cases based on user interactions by data-mining their logs and their behaviour on the application / service for which tests are to be written, live validation so that in case if an object is modified or removed such as done by most of the IDE's in the form of Intelli-sense like Visual Studio or Eclipse.

Machine Learning in "Test Automation" can help prevent the following cases:

- Saving on Manual Labour of writing test cases
- When something goes wrong a traditional framework is most likely to either drop the testing at that point or skip some steps which may result in incorrect result.
- Tests are not validated until and unless that test is run.

Machine Learning will be able to recover from tests on the fly by applying **fuzzy matching**, which means if an object gets modified or removed, then the script must be able to find the closest object to the one it was looking for and then continue the test. For example, if a web page has options "small, medium, large" at first and the script was written according to that and if another choice i.e. "extra-large" is added then the script must be able to adapt to that and anticipate that change so that the test run can continue running without fail.

6 TESTING GUIDELINES FOR AI APPLICATIONS

Software testing, in theory, is a straightforward activity. For every input, there should be a defined and known output. We enter values, make selections or navigate an application and compare the actual result with the expected one. If they match, we consider the test has passed. If they don't, we possibly have a bug and test has failed.
Testing systems that don't always return the same answers require new approaches. This is especially true when testing AI systems whose responses adapt to what they have learned from previous transactions.
The product is actually tested during the training process, which takes time. Training either brings convergence to accurate results or it diverges.
Here are the guidelines which need to be followed for testing AI applications –

- Acceptance criteria must be objective. We must know the amount of error the users are willing to accept.
- Testing must be conducted with new data. Once we've trained the network and frozen the architecture and coefficients, we must use fresh inputs and outputs to verify its accuracy.
- We cannot count on all results being accurate.
- We must understand the architecture of the network as a part of the testing process. Understanding how the network is constructed will help testers determine if another architecture might produce better results.
- We must communicate the level of confidence we have in the results to management and users. Machine learning systems offer us a unique opportunity to describe confidence in statistical terms, so we must use them.

One important thing to note is that the training data itself could well contain inaccuracies.

7 STEPS FOR APPLYING MACHINE LEARNING TO QA

The typical implementation steps for applying machine learning to QA for any customer organization are as follows –

Due Diligence

- Identify use cases based on severity / priority / pain areas
- Identify Data Sources
- Identify Data characteristics
- Sufficiency
- Relevance
- Correctness
- Consistency
- Uniformity
- Identify POC scope

Proof of Concept

- Load and pre-process Data
- Train the model
- Evaluate the model
- Repeat 1-2-3
- Demonstrate Prediction accuracy using various metrics.
- Feedback on data capturing
- Prepare and present implementation plan
- Sign-off from customer

Implementation

- Establish Data extraction jobs
- Establish Data pre-processing jobs
- Establish Model Training jobs
- Establish prediction jobs
- Establish prediction visualization
- Establish continuous model training workflow
- Establish model evaluation workflows.

8 CONCLUSION

Monitoring and analysing production usage to access the data signal for Software Quality is becoming the standard way to implement **TestOps** model in near future. TechM have implemented a pioneering tool to intake analytic feeds from new requirements impact area, test & production incidents trend, key production volumetric (order volumes, returns) to arrive at Risk Based Testing & Defect Prediction Model using an algorithm for the testing fraternity.

ACKNOWLEDGMENT

The author wishes to thank Dr. Satish Pai – COO (Communication, Media & Entertainment) – Americas, TechMahindra and L. Ravichandran – President & Chief Operating Officer at Tech Mahindra.

REFERENCES

[1] Machine Learning is Fun! by Adam Geitgey (https://medium.com/@ageitgey/machine-learning-is-fun-80ea3ec3c471)

[2] "Predicting Business Outcome - Exploring Operations into Test" - address at UNICOM Next Generation Testing World Conference in Bangalore, 2015 by Shubhadeep Chakraborty, Delivery Head, TechM (https://youtu.be/_fvtzSfgN8U))