Machine Learning Use Cases for DevOps and Continuous Everything

Juni Mukherjee

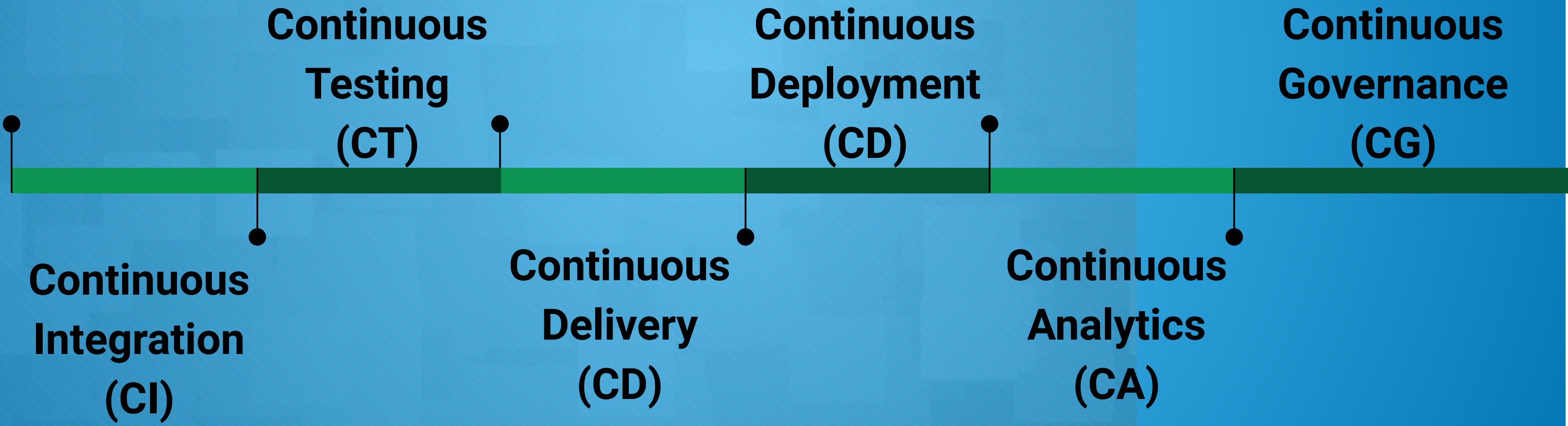
@JuniTweets

Agenda

- ❑ Warm Up: CE, ML, AI
- ❑ Use Cases: Supervised & Unsupervised Learning
- ❑ Pipeline Architecture & Design Patterns

continuity.world
@JuniTweets

CE=CI+CT+CD+CD+CA+CG+...



ML = M + El?

Back To Basics

A.Artificial Intelligence

AI is intelligence demonstrated by machines as opposed to natural intelligence displayed by humans.

Machines mimic "cognitive" functions of humans, like "learning" and "problem solving".

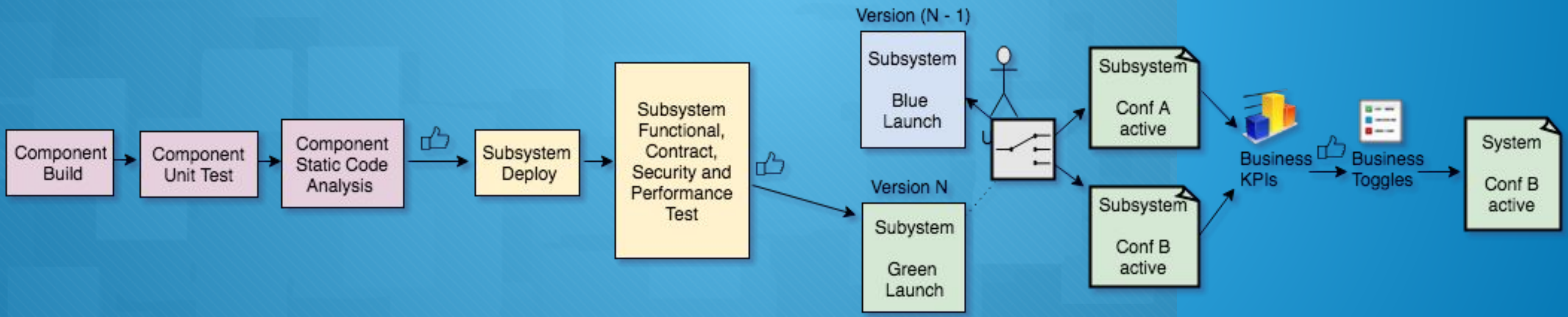
B.Machine Learning

ML algos build a mathematical model based on "training data", in order to make predictions or decisions without being explicitly programmed to perform the task.

Google Trend

Automated Pipelines DO (NOT THINK)

Pipeline as code: Explicitly programmed to perform each task



Terminology Hell

(DevOps & CE) for ML

Models need to undergo CI, CT, CD, CD, CA, CG, ...

====>

Continuous Everything

ML for (DevOps & CE)

- Every company is a data company
- Supervised Learning
- Unsupervised Learning
- Pipeline Architecture
- Pipeline Design Patterns

- Warm Up: CE, ML, AI
- Use Cases: Supervised & Unsupervised Learning
- Pipeline Architecture & Design Patterns

Agenda

continuity.world
@JuniTweets

Machines Can Learn

A.Supervised Learning

Supervised Learning infers a function from labeled training data consisting of a set of training examples.

B.Unsupervised Learning

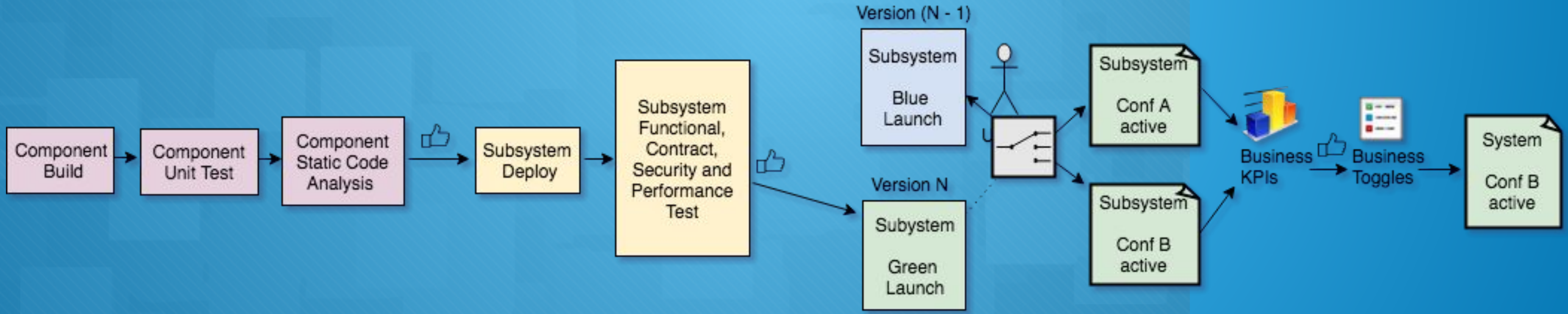
Unsupervised Learning is a type of self-organized learning that helps find unknown hidden structures in unlabeled data sets.

Supervised Learning

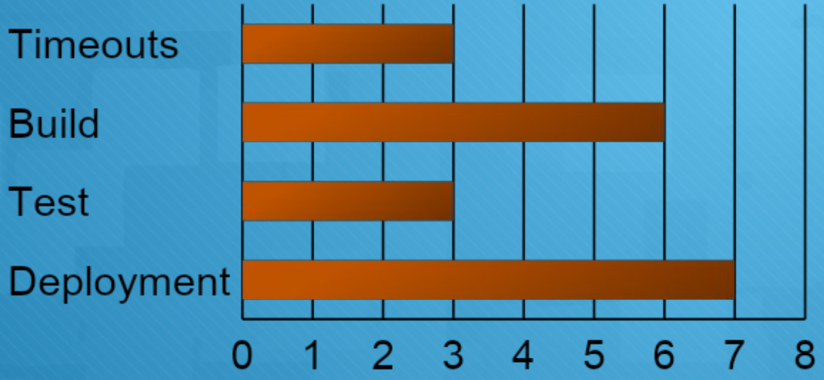
Learn a function that maps an input to an output based on example input-output pairs

Pipelines Can Predict

Predict failures based on past failures



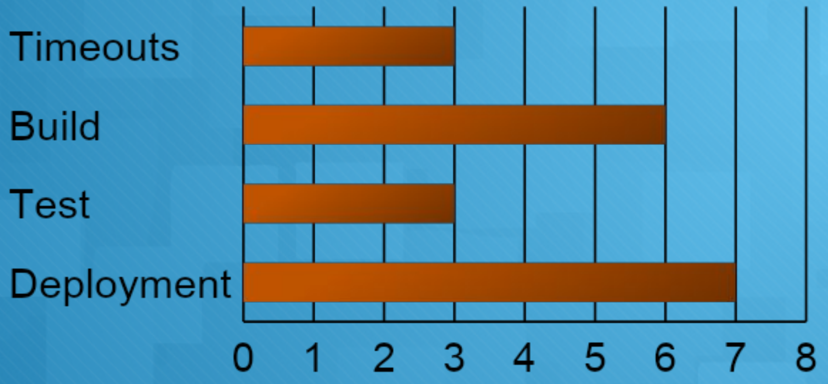
Breaking Bad

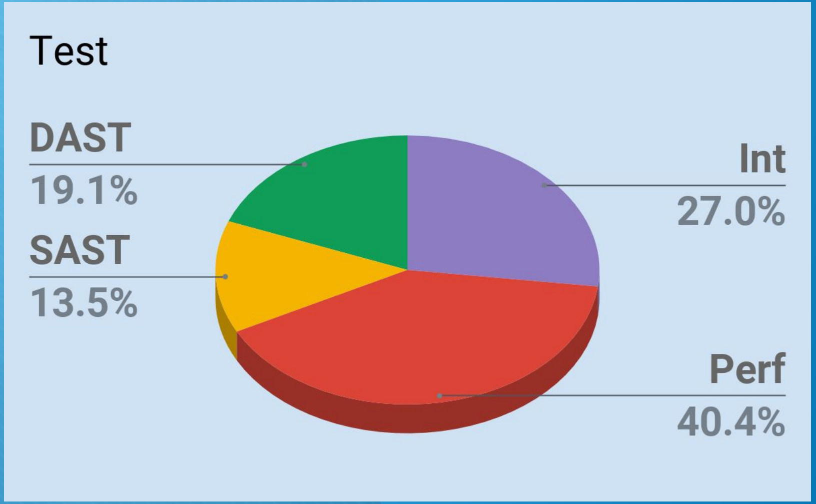


Pipeline Failures:

- Application
- Environmental
  - O&O
  - 3rd Party
- ....

Data Is The New Oil





Step #1/5: Gather Labeled Failure Data

Human labeling

- ❑ Crowdsource: Expose data
- ❑ Internal workforce: Data privacy
- ❑ 3rd party vendors: Domain expertise

Automatic labeling

- ❑ Apply ML with human labels

Consolidation: 1, 2, 3, ...?, Weights, Confidence score: 0.95, ...

Step #2/5: Design Input Feature Vector vs. Target Signal

Sample Features To Predict Pipeline Failures:

- Offending code snippets: 3rd party, O&O
- Vulnerable artifacts: 3rd party, O&O
- Data: PII (SSN/govt. ID, birth date), financial (credit card, bank account)
- Code Quality Index = Function (code coverage, cyclomatic complexity, code dupes)
- Stability Index = Function (responsible speed, escaped defects, repeat customers)
- …

Step #3/5: Choose The Algorithm

A. Author algo(s)

- High barrier of entry
- Got data scientists?
  - Central
  - Embedded

B. Reuse standard algos

- Low barrier of entry
- Empower developers
  - Democratize ML

Step #4/5: Train The Model

- Run the algorithm on the labeled training data

- Predict pipeline failures
  - Decision Tree
  - Leaves represent labels
  - Branches represent conjunctions of features that lead to labels

Step #5/5: Test Model Accuracy

- Define model performance metrics
- Adjust parameters
- Cross-validate
  - Make test data mutually exclusive from training data

Democratize ML: Amazon/AWS Ecosystem

#	Name of service	Utility
1	**SageMaker**	**Built-in algorithms to build, train and deploy**
2	**Mechanical Turk**	**Crowdsource human labeling**
3	**Ground Truth**	❑ **Automatic labeling** ❑ **20% human ----> 80% ML**
4	**Simple Storage Service (S3)**	❑ **Training and test data** ❑ **Models**
5	**IAM**	**Manage secure access to (test and training) data**

Democratize ML: Amazon/AWS Ecosystem

#	Name	Utility

6	CloudWatch	❑ Monitor model metrics
		  ❑ Interim monitors
		  ❑ Final evaluation

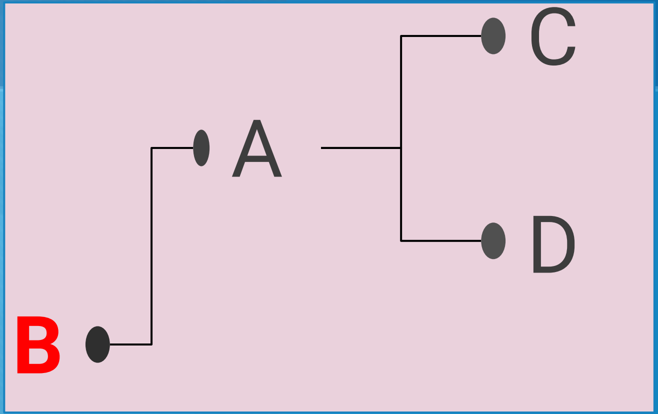
7	[Serverless] Lambda	❑ Write and upload functions ❑ Triggers from S3, CloudWatch, …
8	Step Functions	Orchestrate overall workflow
9	EC2	Storage. Leverage spot economy
10	CloudTrail	Generate audit trails

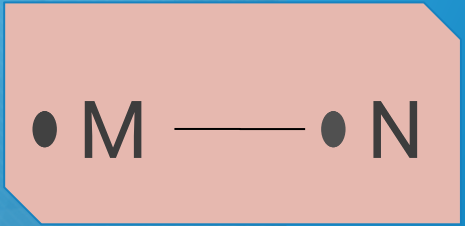
Unsupervised Learning

…Because Past Performance Is Not Indicative Of Future Results

Clustering Reveals Hidden Structures

- Identify incident root-cause vs. symptoms
  - Prioritize B. A, C, D should go away
  - Prevent death by a thousand P1, S1 alerts!
- Track related incidents
  - If M or N happen, proactively alert for both





Find Patterns: Disparate Data Sources

Jenkins X | Spinnaker | CircleCI | Jenkins | AWS CodePipeline ...

GitHub | GitLab | Bitbucket ...

Cloud Foundry | OpenShift | Heroku ...

Splunk | New Relic | Dynatrace | Datadog ...

Liquibase/Datical | Flyway | DBMaestro ...

JUnit | TestNG ...

Sonar | ESLint ...

AWS | GCP | Azure ...

Artifactory | Nexus Repo | S3 ...

SauceLabs | Device Farm ...

BlazeMeter

Coverity

OWASP Zap

Jira ::::::::::

Death by a thousand vendors: # of tool integrations inflate Time2Code

Find Relationships: Correlation | Causation

Source Code Repositories

- ❑ Microsoft GitHub
- ❑ GitLab
- ❑ Atlassian Bitbucket

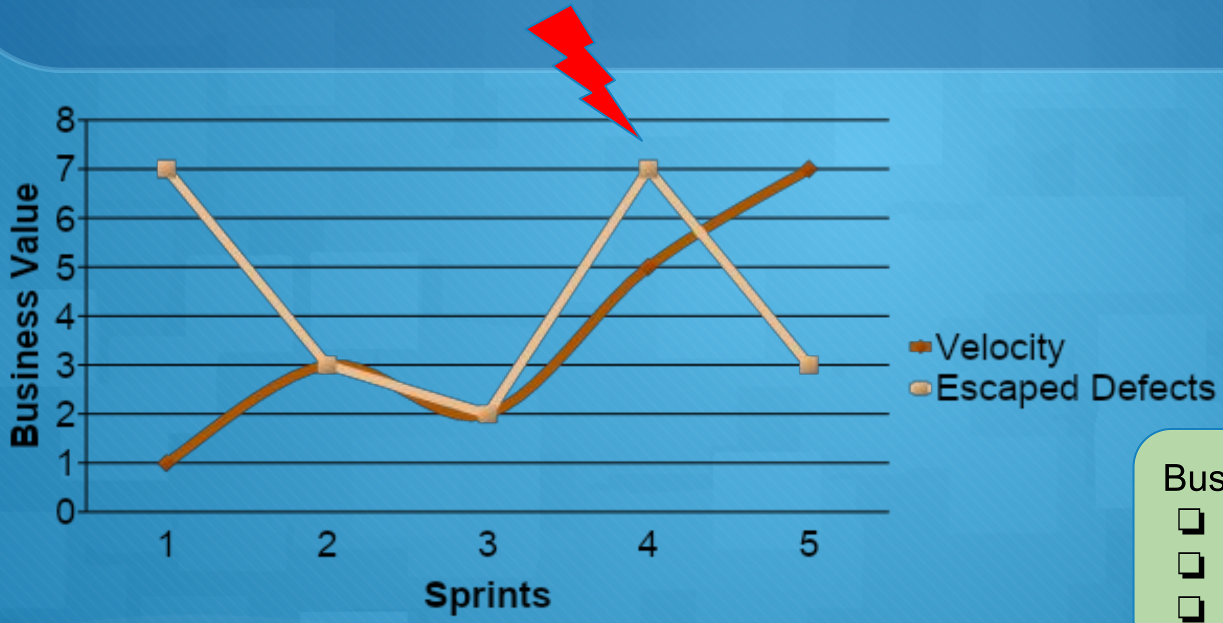
Artifact Repositories

- ❑ JFrog Artifactory
- ❑ Sonatype Nexus Repository
- ❑ Amazon S3

Death by a thousand dependencies: # of dependencies inflate

- ❑ Time2Code
- ❑ Test Cycle Time => Feature Lead Time => Time To Market

Usual Suspect: Tech Relationship With Biz

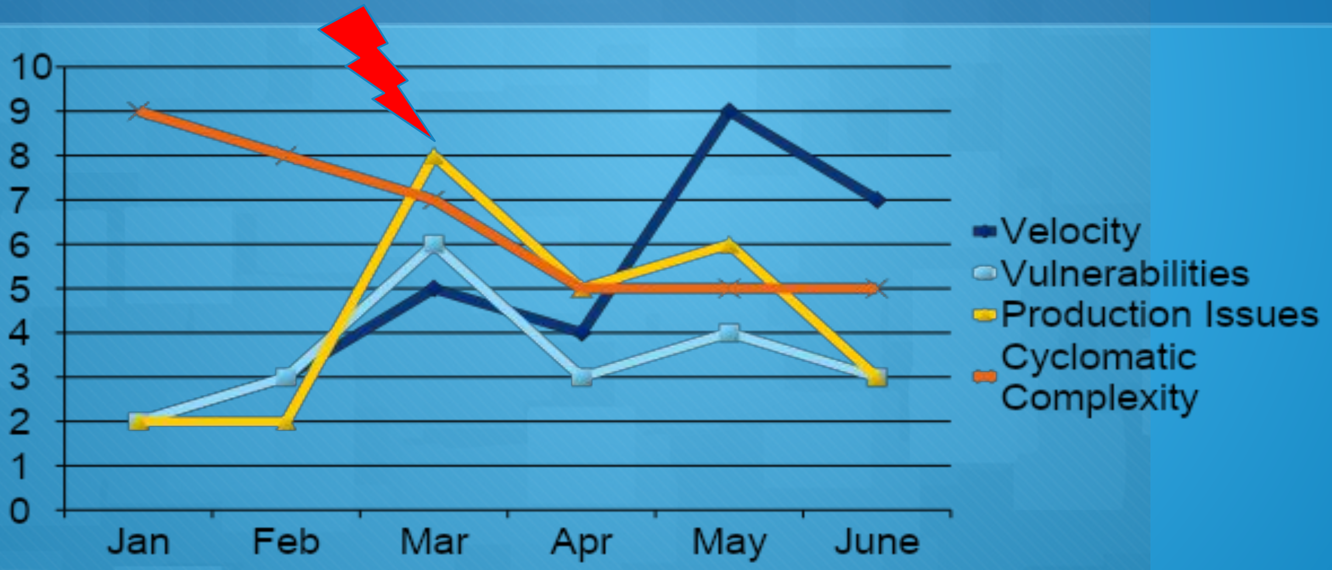


Quality: # of automated tests executed

Release: # of releases completed

Business Value:
- ❑ Revenue goal
- ❑ New customer acquisitions
- ❑ Number of downloads ::::

House of Cards: March Madness



Strange Things: Anomaly Detectors

Source Control Repo

Daily Average:

- ❑ #A reads
- ❑ #B writes

Artifact Repo

Daily Average:

- ❑ #M downloads
- ❑ #N uploads

1000x activity!

On specific days?

At specific times?

On a specific coast?

Stranger Things: Anomaly Detectors

Pipelines

- One took too long?
  - All tests passed.
  - Lurking print stmt?

Tests

- Never failed?
  - Never ran!
    - Test duration ~ 0?
  - Gamed test(s)?

Agenda

- ❑ Warm Up: CE, ML, AI
- ❑ Use Cases: Supervised & Unsupervised Learning
- ❑ Pipeline Architecture & Design Patterns

continuity.world
@JuniTweets

Asynchronous | Services Architecture

SplitInputData (:::)

- ❑ Split input data (K)
  - ❑ Training (K - 1)
  - ❑ Test (1)
- ❑ How BIG is big data?
  - ❑ Version control
  - ❑ Data storage

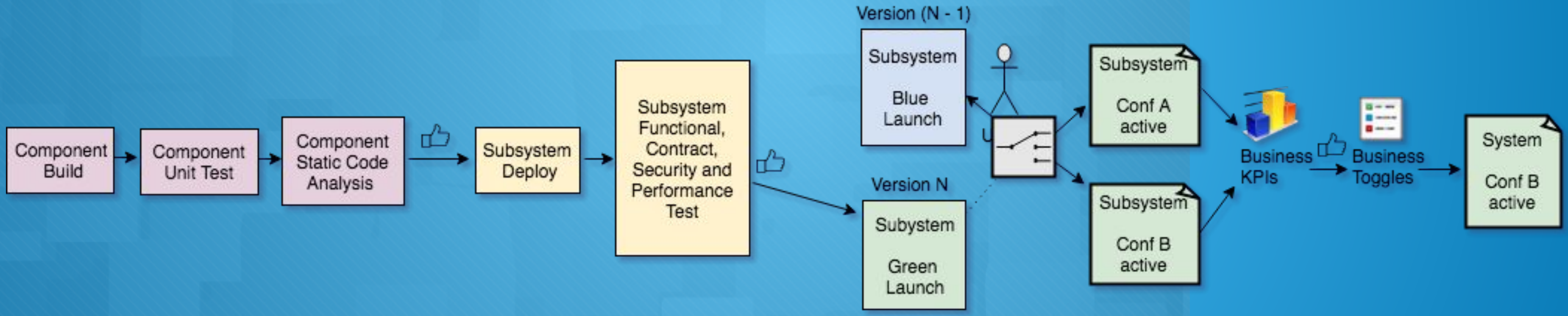
TrainModel (:::)

- ❑ (K - 1) training data
- ❑ Build or buy?
  - ❑ OSS
  - ❑ Commercial?
- ❑ Blackbox or configurable?

CrossValidate (:::)

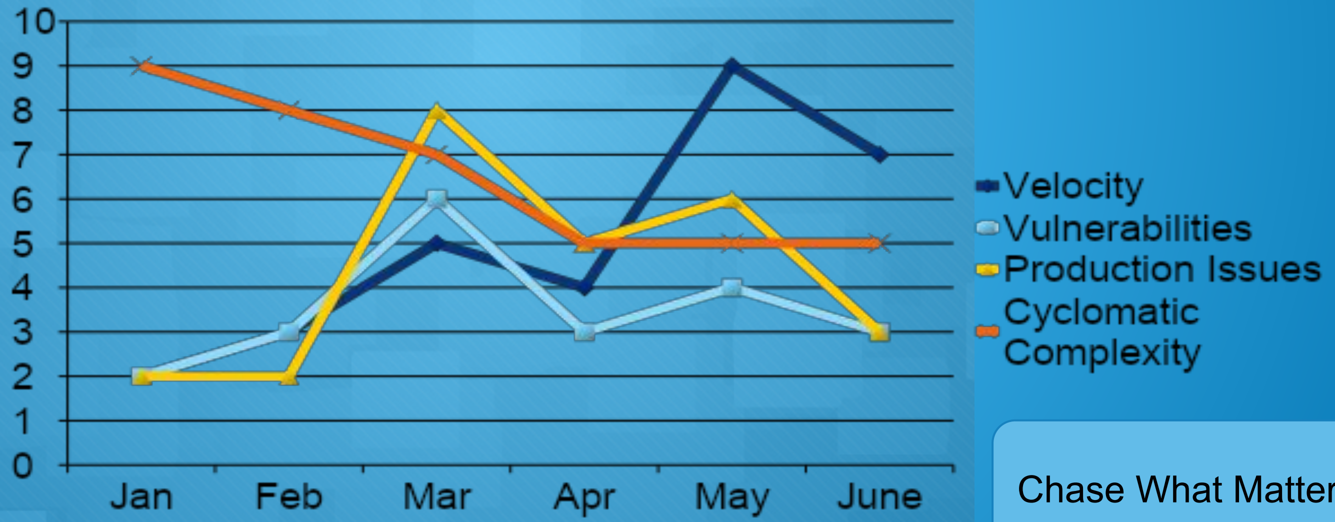
- ❑ (1) test data
- ❑ K-fold cross-validation: K-peat, with a different slice

Simple Software Gates



Simple Gate: Static Thresholds

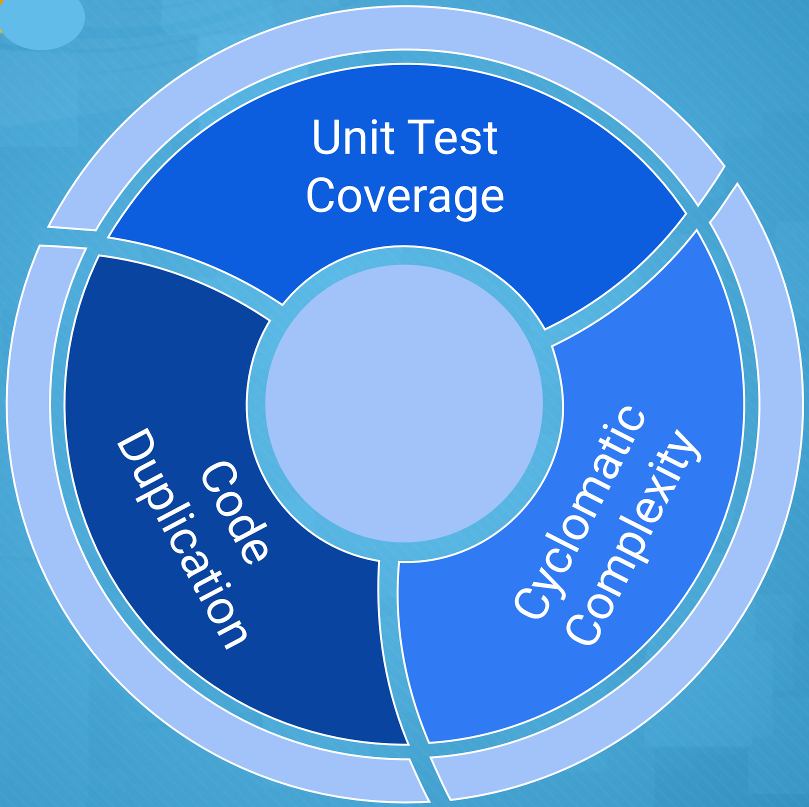
Should we fail pipelines if our code coverage drops below 80%?



Chase What Matters!

Composite Gate: Removes Bias and Skew

Composite Software Gate #1: **Code Quality Index**



How should I gate promotion of versioned artifacts based on “Code Quality”?

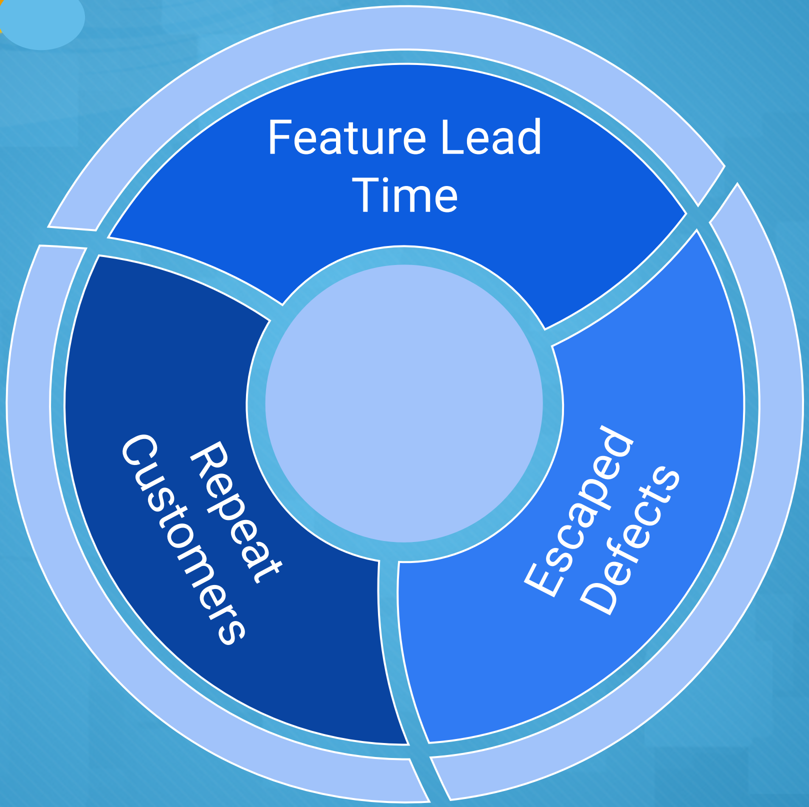
How do I define “Code Quality”?

Diversify your portfolio of KPIs

Smart Gate Monitors Code Quality

Example: Sum(weight[i] * feature[i]) => “Code Quality” Index

Composite Software Gate #2: Stability Index



How do I define a stability index that satisfies Dev, Ops and Business?

How do I automate Segregation of Duties (SoD)?

SoD: Don’t put all your eggs in the same basket

Pipeline Design Patterns

Anti-corruption Layer

Acts like a facade between different subsystems that don't share the same semantics.

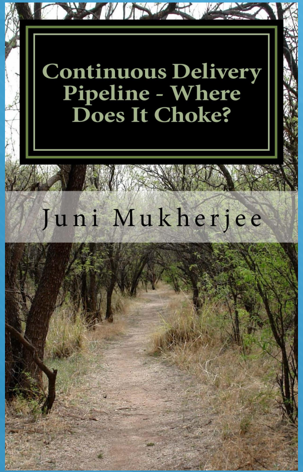
Circuit Breaker

Unstable models should not break fast-feedback pipelines.

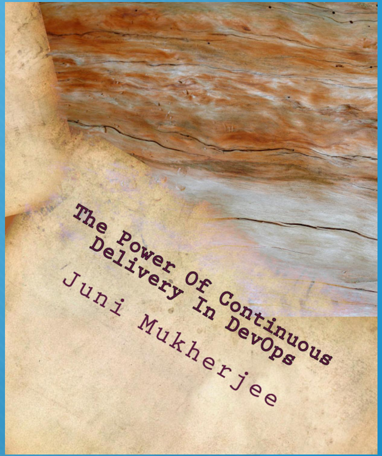
Protect with a circuit breaker.

eBook offer on Amazon

Borrow It For Free

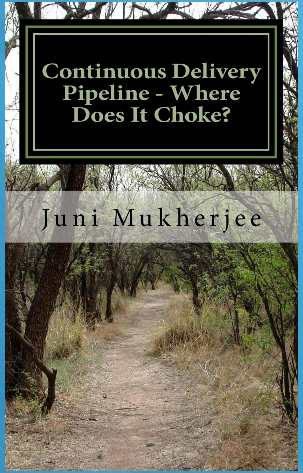


Own It For Free (4 days)

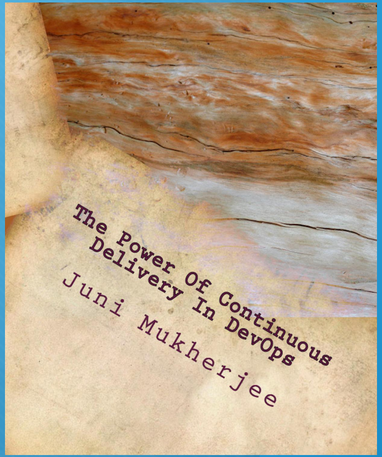


5-pack Paperback 20% Discount: 1st 10, 4 days

Discount code “book1”



Discount code “book2”



Machine Learning Use Cases for DevOps and Continuous Everything

Thank you!

continuity.world

T: @JuniTweets

L: https://www.linkedin.com/in/junim/

E: powerofcontinuity@gmail.com