Toward Machine Learning Assurance

Matthew Churilla

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Document Markings

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM20-0031

Introduction

The Software Engineering Institute (SEI) is a federally funded research and development center (FFRDC) sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University.

The SEI's mission is to advance the technologies and practices needed to acquire, develop, operate, and sustain software systems that are innovative, affordable, trustworthy, and enduring.

Goals

- Frame the need for machine learning assurance
- Discuss lifecycle, processes and controls

Machine Learning

- The process of using data & math to produce a model that can be used to inform future decisions
- The process of going from data to a model is a software system, it is named the machine learning pipeline
- Useful in software systems where formal rules would be difficult or impossible to implement

Failure Modes in Machine Learning (Microsoft / Harvard)

Data

Model

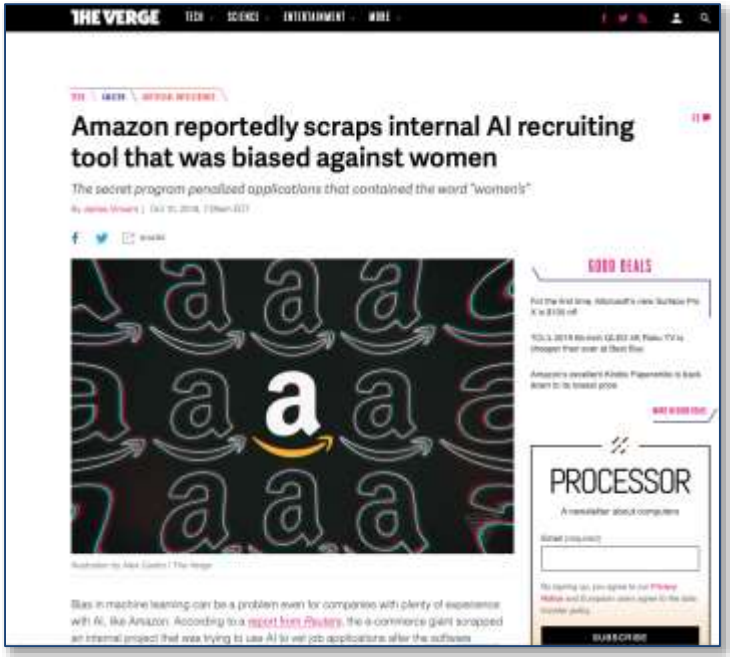
Software System

- Outlines 17 categories of ways machine learning can fail to operate as intended
- Intentional failures (Attacks)
- Unintended failures
- https://docs.microsoft.com/en-us/security/failure-modes-in-machine-learning

Failure Modes Table

Scenario Number	Attack	Overview
1	Perturbation attack	Attacker modifies the query to get appropriate response
2	Poisoning attack	Attacker contaminates the training phase of ML systems to get intended result
3	Model Inversion	Attacker recovers the secret features used in the model by through careful queries
4	Membership Inference	Attacker can infer if a given data record was part of the model's training dataset or not
5	Model Stealing	Attacker is able to recover the model through carefully-crafted queries
6	Reprogramming ML system	Repurpose the ML system to perform an activity it was not programmed for
7	Adversarial Example in Physical Domain	Attacker brings adversarial examples into physical domain to subvertML system e.g
8	Malicious ML provider recovering training data	Malicious ML provider can query the model used by customer and recover customer's training data
9	Attacking the ML supply chain	Attacker compromises the ML models as it is being downloaded for use
10	Backdoor ML	Malicious ML provider backdoors algorithm to activate with a specific trigger
11	Exploit Software Dependencies	Attacker uses traditional software exploits like buffer overflow to confuse/control ML systems
12	Reward Hacking	Reinforcement Learning (RL) systems act in unintended ways because of mismatch between stated reward and true reward
13	Side Effects	RL system disrupts the environment as it tries to attain its goal
14	Distributional shifts	The system is tested in one kind of environment, but is unable to adapt to changes in other kinds of environment
15	Natural Adversarial Examples	Without attacker perturbations, the ML system fails owing to hard negative mining
16	Common Corruption	The system is not able to handle common corruptions and perturbations such as tilting, zooming, or noisy images.
17	Incomplete Testing	The ML system is not tested in the realistic conditions that it is meant to operate in

Examples of failures-1

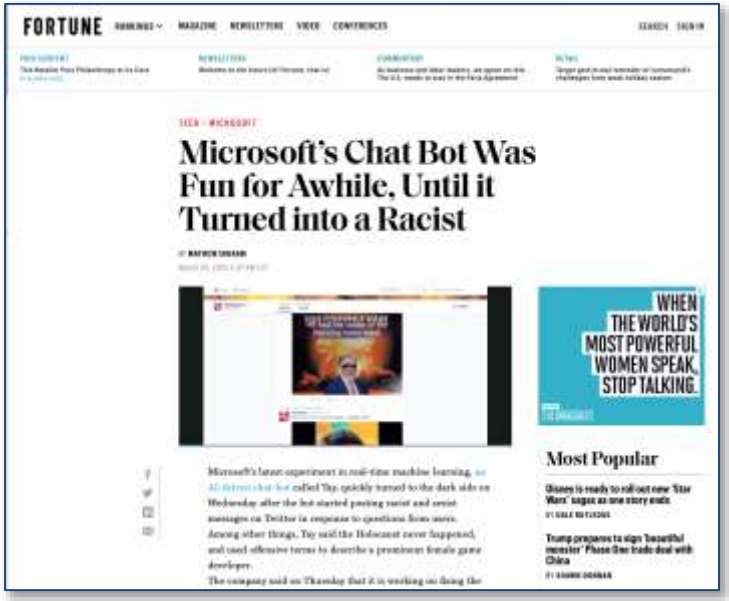


https://www.theverge.com/2018/10/10/17958784/ai-recruiting-tool-bias-amazon-report

Amazon Recruiting Tool's Gender Bias

- https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G

Examples of failures-2

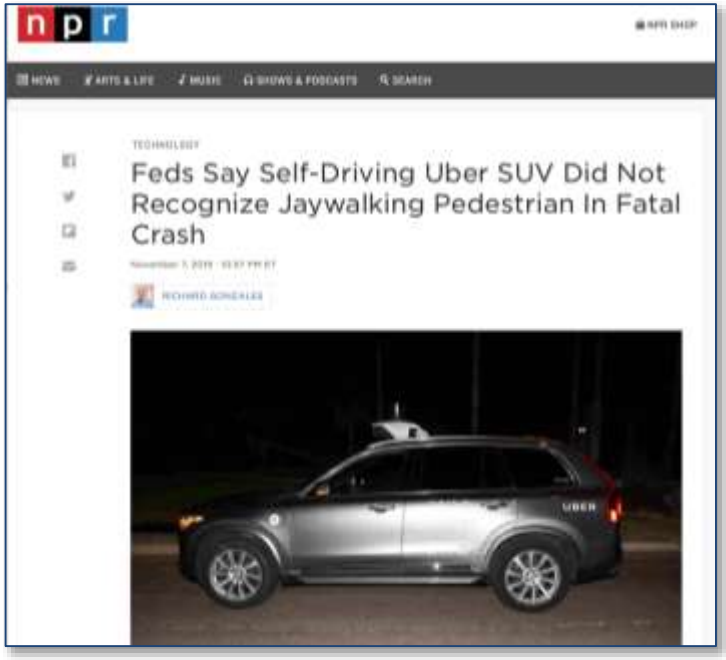


https://fortune.com/2016/03/24/chat-bot-racism/

Tay a Microsoft Chat Bot learns offensive messaging

- https://en.wikipedia.org/wiki/Tay_(bot)

Examples of failures-3

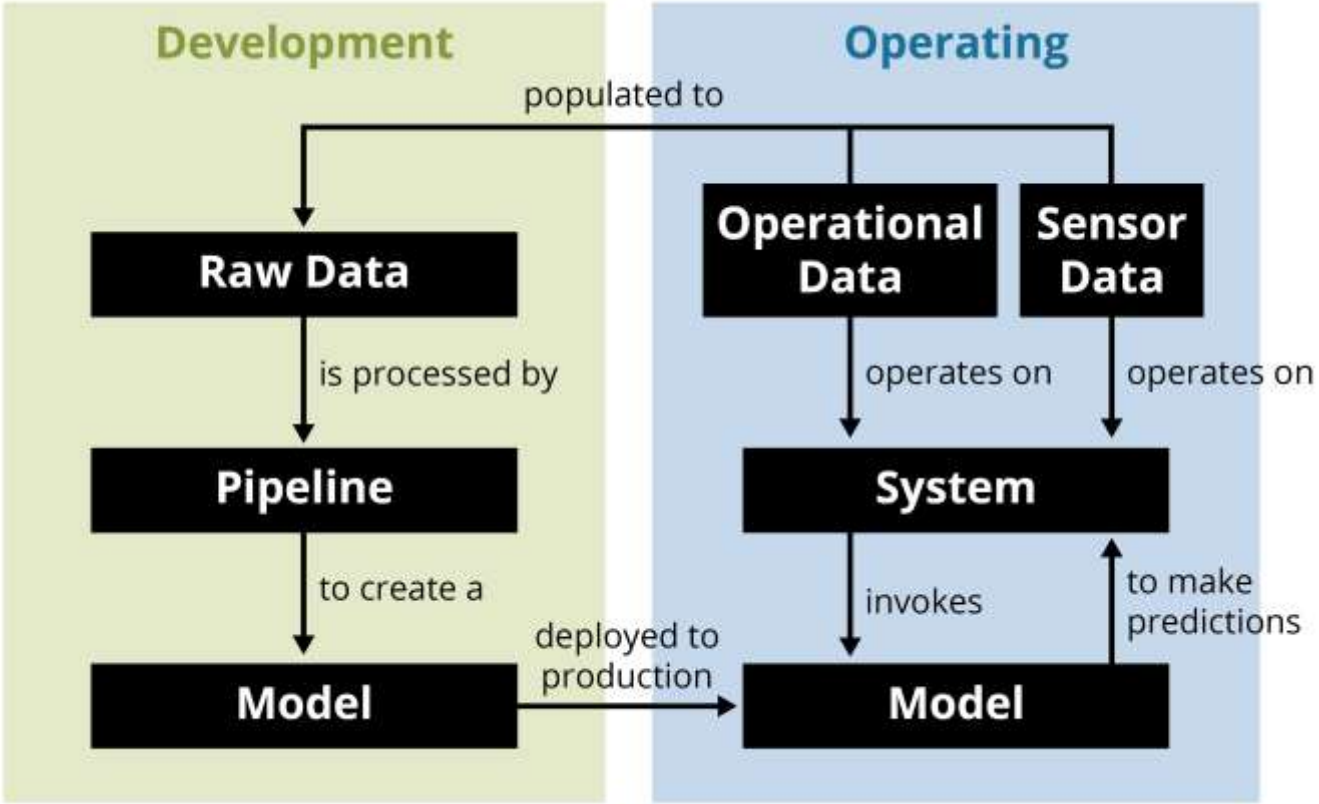


https://www.npr.org/2019/11/07/777438412/feds-say-self-driving-uber-suv-did-not-recognize-jaywalking-pedestrian-in-fatal-

Uber Autonomous Vehicle's Fatal Collision

- https://www.ntsb.gov/investigations/AccidentReports/Reports/HAR1903.pdf

Machine Learning is a Cycle



Machine Learning Model

Think of it as a black box

- Come in many formats
- Data Inputs go in
- Output predictions come out
- Due to non-deterministic nature it is not always correct!

How do you assure or do system analysis on a black box?

- Why was a given output produced?
- What is the correct output for a given input?
- How do you determine if two models are similar?
- Does ground truth exist?
- Can you even assure the model?

Assuring the Software System(s)

- Accept and account for a tight coupling between data, model, and software systems that use the model
- Need to assure that the machine learning component is used correctly by the system. (no bad inputs, proper purpose)
- Need to assure that the system can not be put into a bad state by a machine learning output
  - Remember that the model might not always be correct
- The Machine Learning Pipeline is a software system
- Continuous Evaluation should be the norm

Assuring the Model – Things to remember

- Changes to the model will result in changes in software behavior
- The model might perform well when tested but not well when deployed
- The model is valuable intellectual property

Assuring the Model

Ideal: Common specification and packaging for models

What to do now:

- Checksum model to detect tampering
- Encrypt model at rest
- Document and communicate, Inputs, and Outputs, and behavior

Assuring the Model - Validation

Testing and Quality Validation for AI Software–Perspectives, Issues, and Practices

- https://ieeexplore.ieee.org/abstract/document/8811507

Proposes validation methods

- Classification based testing
- Learning based testing
- Rule based testing
- Metamorphic testing
- Model based testing

Assuring The Data – Things to remember

- If training data changes model behavior will change
- Training data may contain sensitive information
- Two types of data Raw and Prepared data
- Reality is not static

Assuring The Data

Ideal: Static Analysis for data toolchain

What to do now:

- Version and secure data
- Manual analysis before model generation
- Communicate dependencies to system owners

Final thoughts

Machine Learning components (data, pipeline, model) need to be assured

A machine learning component in a software system creates a tight coupling between the data, model, and the software system

Assurance teams will need access to Statistics and Machine Learning experts

Until tools and automation can catchup documentation and communication will be very important

Reality is not static so Machine Learning Assurance needs to be a continuous process

Informational Sources

Failure Categorization Papers

NIST Adversarial Machine Learning Draft - https://nvlpubs.nist.gov/nistpubs/ir/2019/NIST.IR.8269-draft.pdf
Failure Modes in Machine Learning - https://docs.microsoft.com/en-us/security/failure-modes-in-machine-learning
Machine Learning: The High Interest Credit Card of Technical Debt - https://research.google/pubs/pub43146/

Testing Machine Learning References

A validation methodology for AI simulation models - https://ieeexplore.ieee.org/abstract/document/1527652
Testing and Quality Validation for AI Software Perspectives, Issues, and Practices - https://ieeexplore.ieee.org/abstract/document/8811507

Failure Examples

Amazon - https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G
Tay - https://en.wikipedia.org/wiki/Tay_(bot)
Uber - https://www.ntsb.gov/investigations/AccidentReports/Reports/HAR1903.pdf

Questions?

XKCD Machine Learning: https://xkcd.com/1838/

Contact information:

Matthew Churilla
churilla@cert.org

Software Engineering Institute

info@sei.cmu.edu