Production Systems

Intro

- We have focused our discussion on implementing machine learning algorithms for security in isolated lab environments.
- After you have proven that the algorithm works, the next step will likely be to get the software ready for production.
- Deploying machine learning systems in production comes with an entirely different set of challenges and concerns
- Let’s begin by concretely defining what it means for such systems to be production ready, deployable, and scalable.

Defining Machine Learning System Maturity and Scalability

- Data Quality
  - Unbiased data
  - Verifiable ground truth
  - Sound treatment of missing data
- Model Quality
  - Efficient hyper-parameter optimization
  - A/B testing of models
  - Timely feedback loops
  - Repeatable results
  - Explainable results
- Performance
  - Low-latency training and predictions
  - Scalability
  - Automated and efficient data collection

Defining Machine Learning System Maturity and Scalability

- Maintainability
  - Checkpointing and versioning of models
  - Smooth model deployment process
  - Graceful degradation
  - Easily tunable and configurable
  - Well documented
- Monitoring and Alerting
  - System health/performance monitoring (i.e., is it running?)
  - System efficacy monitoring (i.e., precision/recall)
  - Monitoring data distributions (e.g., user behavior changing or adversaries adapting)
- Security and Reliability
  - Robust in adversarial contexts
  - Data privacy safeguards and guarantees

Data Quality

- The quality of a machine learning system's input data dictates its success or failure.
- When training an email spam classifier using supervised learning,
  - If feeding an algorithm training data that contains only health and medicine advertising spam emails will not result in a balanced and generalizable model.
  - The resulting system might be great at recognizing unsolicited emails promoting weight-loss medication, but it will likely be unable to detect adult content spam.

Data Quality (Problem: Bias in Datasets)

- Well-balanced datasets are scarce, and using unbalanced datasets can result in bias that is difficult to detect.
- Because it is typically impossible to collect samples from the universe of data (the entire population),
  - datasets are created by drawing from sources that produce samples belonging to the population.

Data Quality (Problem: Bias in Datasets)

- *Selection bias* and *exclusion bias* are common forms of bias that can be caused by flawed data collection flows.
- These forms of bias are introduced by the systematic and improper selection or exclusion of data from the population intended to be analyzed.

Data Quality (Problem: Bias in Datasets)

- *Observer bias*, or the observer-expectancy effect, is another common type of bias caused by errors in human judgment or human - designed processes.
- Software binary feature extraction processes might be biased toward certain behavior exhibited by the binaries that human analysts have been trained to look out for

Data Quality (Problem: Label Inaccuracy)

- When doing supervised learning, mislabeled data will cause machine learning algorithms to lose accuracy
- Checking the correctness of labels in a dataset often requires expensive human expert resources.
- Even doing random subset validation on datasets can still be expensive.

Data Quality (Problem: Missing Data)

- Missing data is one of the most common problems that you will face when working with machine learning.
- It is very common for datasets to have rows with missing values.

Table 7-1. Sample rows drawn from employee attrition dataset

	TotalWorkingYears	MonthlyIncome	Overtime	DailyRate	Label
0	NaN	6725	0	498.0	0
1	12.0	2782	0	NaN	0
2	9.0	2468	0	NaN	0
3	8.0	5003	0	549.0	0
4	12.0	8578	0	NaN	0

Model Quality

- *Trained models* form the core intelligence of machine learning systems.
- But without safeguards in place to ensure the quality of these models, the results they produce will be suboptimal.
- Models can take on different forms depending on the machine learning algorithm used,
  - They are essentially data structures containing the parameters learned during the algorithm's training phase.
- Regular maintenance and reevaluation will ensure that it does not degrade over time.

Model Quality (Problem: Hyperparameter Optimization)

- Hyper-parameters are machine learning algorithm parameters that are not learned during the regular training process

```
from sklearn import tree
classifier = tree.DecisionTreeClassifier(max_depth=12,
                                         min_samples_leaf=3,
                                         max_features='log2')
```

Model Quality (Problem: Hyperparameter Optimization)

- Hyperparameters typically need to be chosen before commencing the training phase. Some questions:
- How do you know what to set the learning rate to? Or
- How many hidden layers in a deep neural network will give the best results? Or
- What value of k to use in k-means clustering?

Model Quality (Problem: Hyperparameter Optimization)

- The problem can be naively approached in a "brute-force" fashion:
  - Training different models using all different combinations of the algorithm's hyper-parameters, and
  - then selecting the set of hyper-parameters that results in the best-performing model.
- Hyper-parameter optimization is most commonly done using a technique called *grid search*, an exhaustive sweep through the hyper-parameter space of a machine learning algorithm.

Model Quality (Feature: Feedback Loops)

- Because security machine learning systems have such a low tolerance for inaccuracies,
  - every bit of user feedback needs to be taken into account to improve system efficacy as much as possible.
- For example, an anomaly detection system that raises too many *false positive* alerts to security operations personnel should take advantage of the correct labels given by human experts during the alert triaging phase to retrain and improve the model.

Model Quality (Feature: Feedback Loops)

- Feedback loops are a good way to not only detect when the model is deteriorating but also gather labeled training data for continuously improving the system

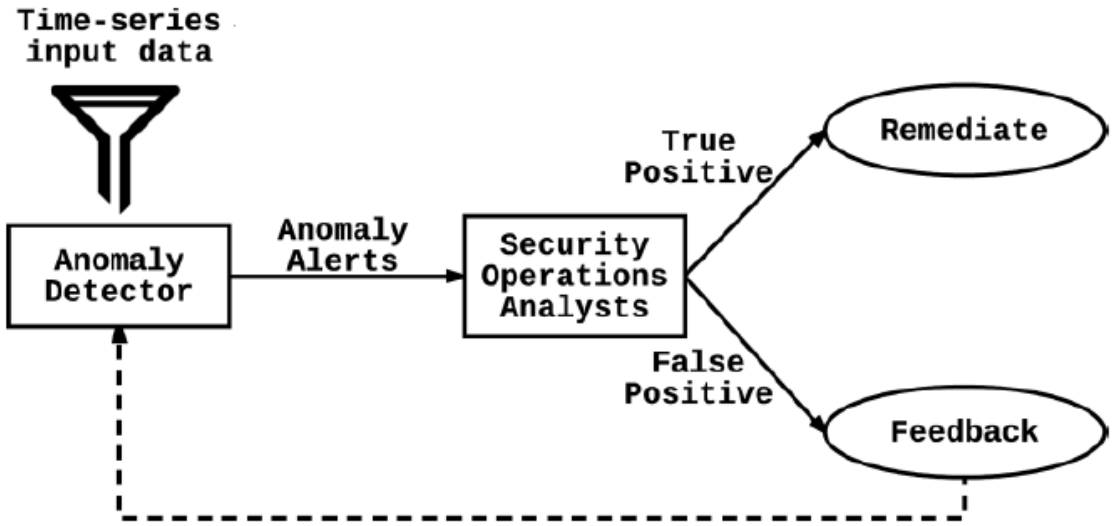

Figure 7-2. Anomaly detection system with feedback loop

Model Quality (Feature: Repeatable and Explainable Results)

- Repeatability of machine learning predictions is a simple concept:
  - Assuming constantly changing priors in a statistical system (due to continuous adaptation, manual evolution, etc.), we should be able to reproduce any decision made by the system at any reasonable point in its history.

Model Quality (Feature: Repeatable and Explainable Results)

- the goal of Explainable AI (XAI) is to create a suite of new or modified machine learning techniques that produce *explainable models* that,
  - when combined with effective explanation techniques, enable end users to understand, appropriately trust, and
  - effectively manage the emerging generation of AI systems.

Performance

- By nature, many security machine learning systems are in the direct line of traffic, where they are forced to make rapid-fire decisions or risk falling behind.
  - Detecting an anomaly 15 minutes after the event is often too late.
- Systems that have real-time adaptability requirements must also meet a high bar for efficiently implementing continuous incremental retraining

Performance (Goal: Low Latency, High Scalability)

- **For security machine learning systems in critical decision paths, the end user's tolerance for high-latency responses might be limited.**
  - A good design choice to take machine learning systems out of the main line of interaction between users and a system.
  - Your security system should make its decisions asynchronously wherever possible, and it should be able to remediate or mitigate threats in a separate and independent path.
- **In even more strict systems, it is worthwhile to invest resources to optimizing the machine learning system to minimize latency, especially when under heavy load.**

Performance (Performance Optimization)

- **To speed up ML applications, we can search for performance bottlenecks in the program execution framework, find more efficient algorithms, or use parallelism.**
- **Approach 1: Profiling and framework optimization**
  - The profiler typically inserts hooks into components, functions, events, code, or instructions being executed, and does a deep analysis of the time it takes for each individual component to run.
  - Core algorithms in *scikit-learn* are frequently *Cython* wrappers around other popular and well-maintained machine learning libraries written in native C or C++ code.

Performance (Performance Optimization)

- **Approach 2: Algorithmic optimization**
  - Having fewer features means having to do fewer arithmetic operations, which can improve performance
  - Tree-based models (e.g., *decision trees, random forests*) tend to have very good prediction performance because every query interacts only with a small portion of the model space
  - Linear models are fast to train and evaluate

Performance (Performance Optimization)

- **Approach 2: Algorithmic optimization (cont)**
  - SVMs suffer from widely known scalability problems
  - Deep learning algorithms (deep neural nets) are slow to train and quite resource intensive (*i.e millions of matrix multiplications involved*), but can easily be parallelized with the appropriate hardware—e.g., graphics processing units (*GPUs*)—and modern frameworks such as *Tensor-Flow, Torch, or Caffe*.
  - Approximate nearest neighbor search algorithms such as *k-d trees* can significantly speed up close-proximity searches in large datasets

Performance (Horizontal Scaling with Distributed Computing Frameworks)

- Many steps of the machine learning process can benefit from parallelism, but many datasets and algorithms cannot be “blindly distributed” because each unit of operation might not be independent.
- *Scikit-learn* is designed for single-node execution, but there are some types of tasks that are better suited for the distributed computing.
- *Spark-sklearn* is a Python package that integrates the *Spark* computing framework with *scikit-learn*, focused on hyper-parameter optimization

Performance (Using Cloud Services)

- All of the popular public cloud providers have several machine learning and data infrastructure offerings that you can use to quickly and economically scale your operations.
- These services relieve organizations of the operational overhead of managing a *Spark* cluster or *TensorFlow* deployment that requires significant effort to configure and maintain.
- The largest players in the public cloud arena, such as *Amazon Web Services (AWS) and Google Cloud Platform (GCP),* provide powerful APIs for video, speech, and image analysis using *pre-trained* machine learning models

Maintainability

- Problem: Checkpointing, Versioning, and Deploying Models
- Goal: Graceful Degradation
  - How to response to degradation of the system?
  - Should security systems *fail open* (allow requests through if the system fails to respond) or *fail closed* (block all requests if the system fails to respond)?
- Goal: Easily Tunable and Configurable

Monitoring and Alert

- A monitoring framework is a system that aggregates metrics from different sources in a central place for manual monitoring and performing anomaly detection.
- Such systems are often made up of five distinct components:
  - Metrics collectors
  - Time series database
  - Detection engine
  - Visualization layer
  - Alerting mechanism

Security and Reliability

- Wherever security solutions are deployed, malicious activity should be expected.
- Let's look at the security and privacy guarantees that security machine learning systems should provide.
  - Feature: Robustness in Adversarial Contexts
  - Feature: Data Privacy Safeguards and Guarantees

Conclusion

- Security machine learning systems must be one of the strongest links in a modern application environment.
  - As such, these systems need to meet quality, scalability, and maintainability standards that surpass most other components in an operation.
- This lecture provided a framework for evaluating a system's production readiness.