Anomaly Detection

Outline

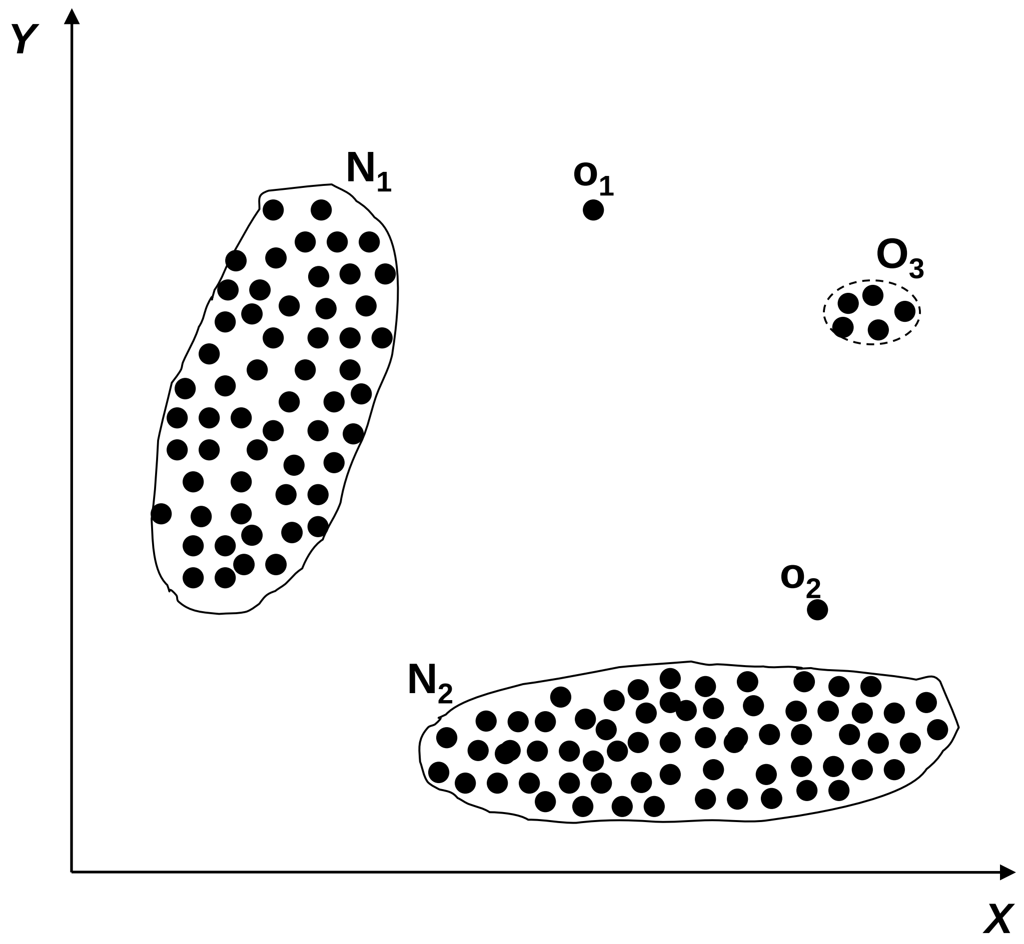
- Introduction
- Aspects of Anomaly Detection Problem
- Applications
- Different Types of Anomaly Detection Techniques
- Case Study
- Discussion and Conclusions

What are Anomalies?

- **Anomaly** is a pattern in the data that does not conform to the expected behavior
  - Also referred to as outliers, exceptions, peculiarities, surprises, etc.
- **Anomalies** translate to significant (often critical) real life entities
  - Cyber intrusions: In the context of network and host security, *"anomaly detection refers to identifying unexpected intruders or breaches"*
  - Credit card fraud
  - Faults in mechanical systems

Simple Examples

- N_1 and N_2 are regions of normal behavior
- Points o_1 and o_2 are anomalies
- Points in region O_3 are also anomalies



Related problems

- Rare Class Mining
- Chance discovery
- Novelty Detection
- Exception Mining
- Noise Removal

Key Challenges

- Defining a representative normal region is challenging
- The boundary between normal and outlying behavior is often not precise
- Availability of labeled data for training/validation
- The exact notion of an outlier is different for different application domains
- Malicious adversaries
- Data might contain noise
- Normal behavior keeps evolving
- Appropriate selection of relevant features

Aspects of Anomaly Detection Problem

- Nature of input data
- Availability of supervision
- Type of anomaly: point, contextual, structural
- Output of anomaly detection
- Evaluation of anomaly detection techniques

Input Data

- Most common form of data handled by anomaly detection techniques is *Record Data*
  - Univariate
  - Multivariate

Engine Temperature
192
195
180
199
19
177
172
285
195
163

Input Data

- Most common form of data handled by anomaly detection techniques is *Record Data*
  - Univariate
  - Multivariate

Tid	SrcIP	Start time	Dest IP	Dest Port	Number of bytes	Attack
1	206.135.38.95	11:07:20	160.94.179.223	139	192	No
2	206.163.37.95	11:13:56	160.94.179.219	139	195	No
3	206.163.37.95	11:14:29	160.94.179.217	139	180	No
4	206.163.37.95	11:14:30	160.94.179.255	139	199	No
5	206.163.37.95	11:14:32	160.94.179.254	139	19	Yes
6	206.163.37.95	11:14:35	160.94.179.253	139	177	No
7	206.163.37.95	11:14:36	160.94.179.252	139	172	No
8	206.163.37.95	11:14:38	160.94.179.251	139	285	Yes
9	206.163.37.95	11:14:41	160.94.179.250	139	195	No
10	206.163.37.95	11:14:44	160.94.179.249	139	163	Yes

Input Data – *Nature of Attributes*

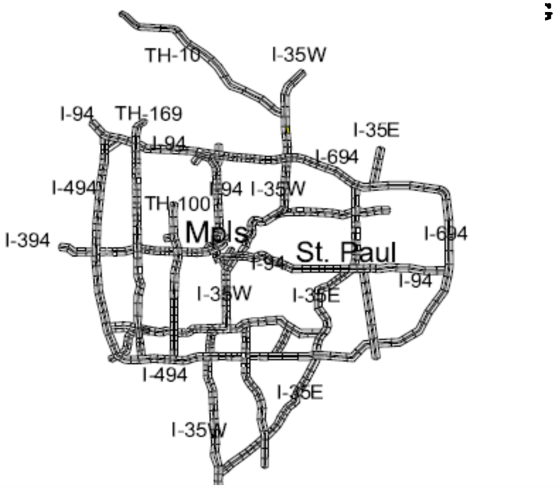
- Nature of attributes
  - Binary
  - Categorical
  - Continuous
  - Hybrid

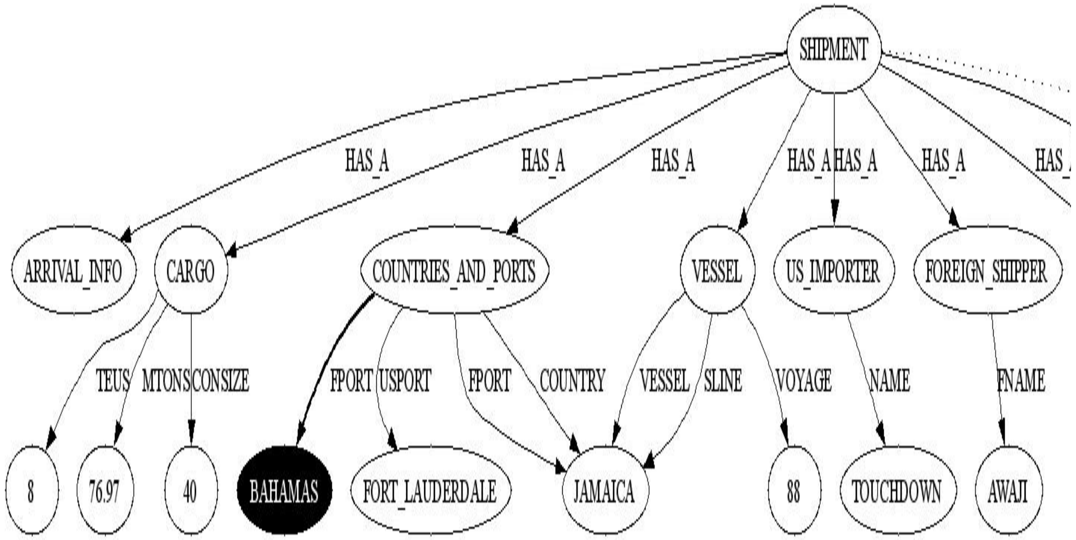
	categorical	continuous	categorical	continuous	binary
Tid	**SrcIP**	**Duration**	**Dest IP**	**Number of bytes**	**Internal**
1	206.163.37.81	0.10	160.94.179.208	150	**No**
2	206.163.37.99	0.27	160.94.179.235	208	**No**
3	160.94.123.45	1.23	160.94.179.221	195	**Yes**
4	206.163.37.37	112.03	160.94.179.253	199	**No**
5	206.163.37.41	0.32	160.94.179.244	181	**No**

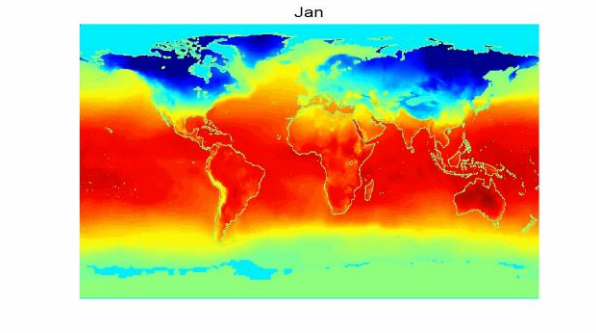
Input Data – *Complex Data Types*

- Relationship among data instances
  - Sequential
    - Temporal
  - Spatial
  - Spatio-temporal
  - Graph

```
GGTTCCGCCTTCAGCCCCGCGCC
CGCAGGGCCCGCCCCGCGCCGTC
GAGAAGGGCCCGCCTGGCGGGCG
GGGGGAGGCGGGGCCGCCCGAGC
CCAACCGAGTCCGACCAGGTGCC
CCCTCTGCTCGGCCTAGACCTGA
GCTCATTAGGCGGCAGCGGACAG
GCCAAGTAGAACACGCGAAGCGC
```







Data Labels

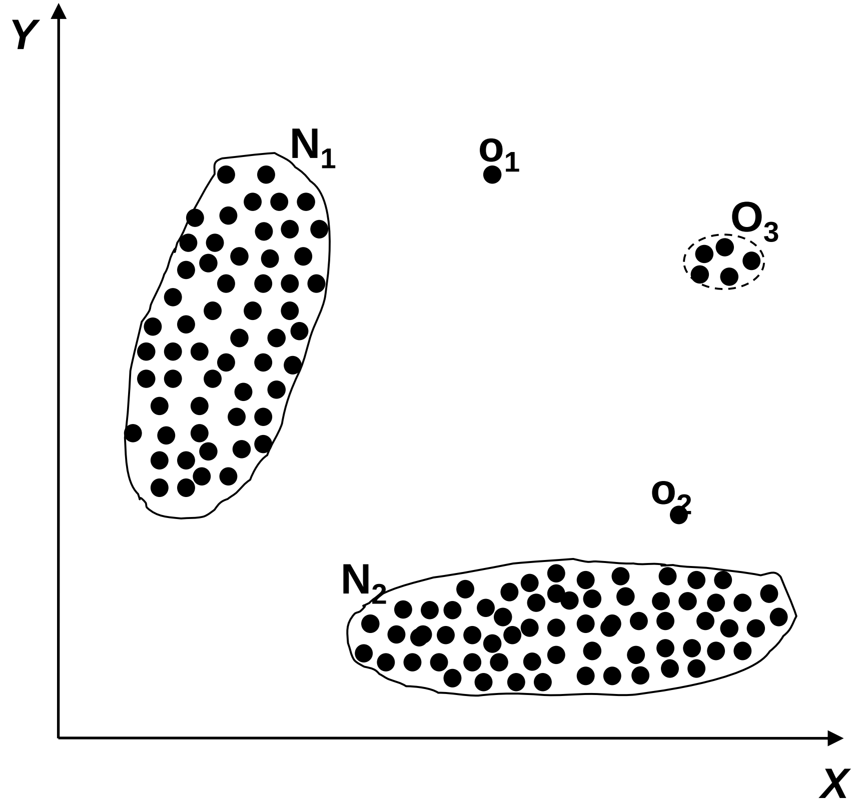
- Supervised Anomaly Detection
  - Labels available for both normal data and anomalies
  - Similar to rare class mining
- Semi-supervised Anomaly Detection
  - Labels available only for normal data
- Unsupervised Anomaly Detection
  - No labels assumed
  - Based on the assumption that anomalies are very rare compared to normal data

Type of Anomalies

- Point Anomalies
- Contextual Anomalies
- Collective Anomalies

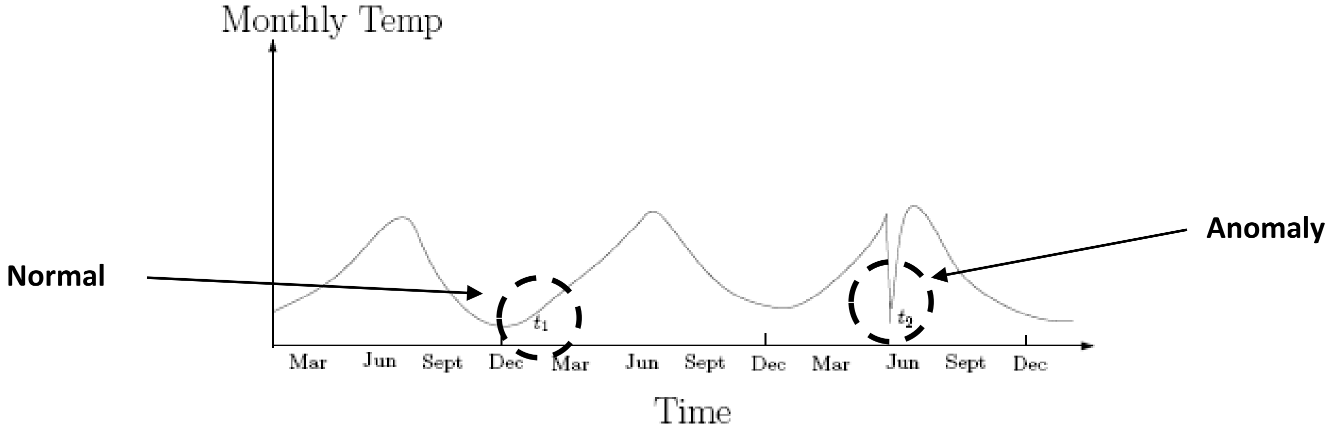
Point Anomalies

- An individual data instance is anomalous w.r.t. the data



Contextual Anomalies

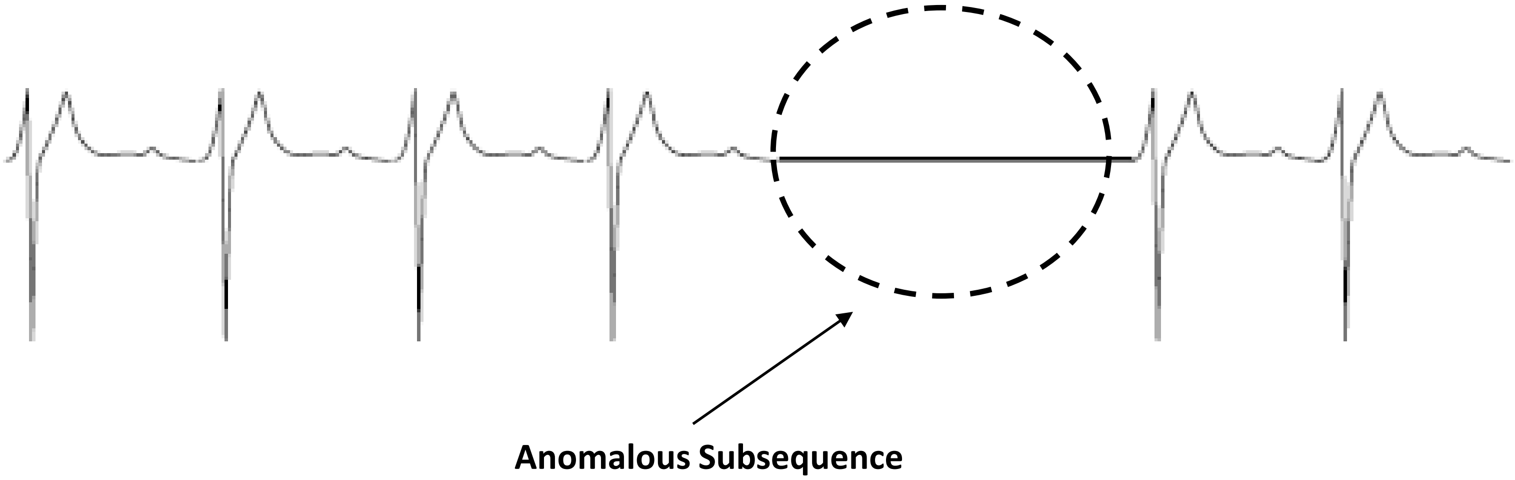
- An individual data instance is anomalous within a context
- Requires a notion of context
- Also referred to as conditional anomalies*



* Xiuyao Song, Mingxi Wu, Christopher Jermaine, Sanjay Ranka, Conditional Anomaly Detection, IEEE Transactions on Data and Knowledge Engineering, 2006.

Collective Anomalies

- A collection of related data instances is anomalous
- Requires a relationship among data instances
  - Sequential Data
  - Spatial Data
  - Graph Data
- The individual instances within a collective anomaly are not anomalous by themselves



Output of Anomaly Detection

- Label
  - Each test instance is given a *normal* or *anomaly* label
  - This is especially true of classification-based approaches
- Score
  - Each test instance is assigned an anomaly score
    - Allows the output to be ranked
    - Requires an additional threshold parameter

Applications of Anomaly Detection

- Network intrusion detection
- Insurance / Credit card fraud detection
- Healthcare Informatics / Medical diagnostics
- Industrial Damage Detection
- Image Processing / Video surveillance
- Novel Topic Detection in Text Mining
- …

Real World Anomalies

- Credit Card Fraud
  - An abnormally high purchase made on a credit card

- Cyber Intrusions
  - A web server involved in *ftp* traffic

Intrusion Detection

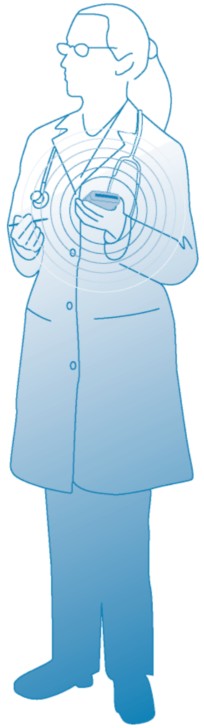
- Intrusion Detection:
  - Process of monitoring the events occurring in a computer system or network and analyzing them for intrusions
  - Intrusions are defined as attempts to bypass the security mechanisms of a computer or network
- Challenges
  - Traditional signature-based intrusion detection systems are based on signatures of known attacks and cannot detect emerging cyber threats
  - Substantial latency in deployment of newly created signatures across the computer system
- Anomaly detection can alleviate these limitations

Fraud Detection

- Fraud detection refers to detection of criminal activities occurring in commercial organizations
  - Malicious users might be the actual customers of the organization or might be posing as a customer (also known as identity theft).
- Types of fraud
  - Credit card fraud
  - Insurance claim fraud
  - Mobile / cell phone fraud
  - Insider trading
- Challenges
  - Fast and accurate real-time detection
  - Misclassification cost is very high

Healthcare Informatics

- Detect anomalous patient records
  - Indicate disease outbreaks, instrumentation errors, etc.
- Key Challenges
  - Only normal labels available
  - Misclassification cost is very high
  - Data can be complex: spatio-temporal



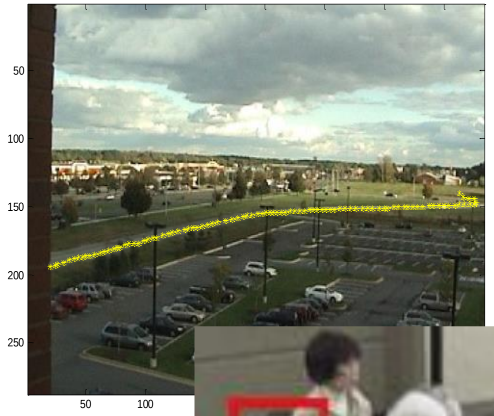
Industrial Damage Detection

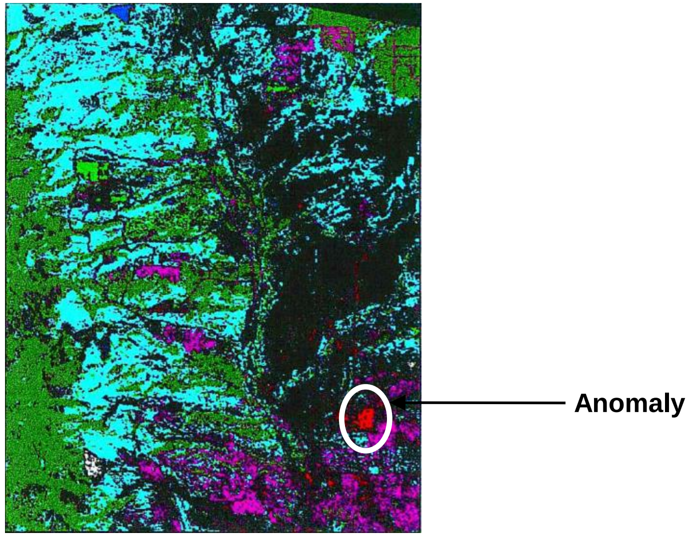
- Industrial damage detection refers to detection of different faults and failures in complex industrial systems, structural damages, intrusions in electronic security systems, abnormal energy consumption, etc.
  - Example: Aircraft Safety
    - Anomalous Aircraft (Engine) / Fleet Usage
    - Anomalies in engine combustion data
    - Total aircraft health and usage management

- Key Challenges
  - Data is extremely huge, noisy and unlabelled
  - Most of applications exhibit temporal behavior
  - Detecting anomalous events typically require immediate intervention

Image Processing

- Detecting outliers in a image or video monitored over time
- Detecting anomalous regions within an image
- Used in
  - mammography image analysis
  - video surveillance
  - satellite image analysis
- Key Challenges
  - Detecting collective anomalies
  - Data sets are very large





1.When to Use Anomaly Detection Versus Supervised Learning

- Anomaly detection is often combined with pattern recognition—for example, using supervised learning
- However, it is sometimes unclear which approach to take when looking to develop a solution for a problem.
  - In many cases, it can be difficult to find a representative pool of positive examples that is sufficient for the algorithm to get a sense of what positive events are like.
  - Server breaches are sometimes caused by zero-day attacks or newly released vulnerabilities in software

When to Use Anomaly Detection Versus Supervised Learning

- By definition, the method of intrusion cannot be predicted in advance, and it is difficult to build a profile of every possible method of intrusion in a system.
- Because these events are relatively rare, this also contributes to the class imbalance problem that makes for difficult application of supervised learning.
- Anomaly detection is perfect for such problems.

2.Intrusion Detection with Heuristics

- Intrusion detection systems (IDSs) have been around since 1986 and are commonplace in security-constrained environments.
- Even today, using thresholds, heuristics, and simple statistical profiles remains a reliable way of detecting intrusions and anomalies.
  - For example, suppose that we define 10 queries per hour to be the upper limit of normal use for a certain database.
  - Each time the database is queried, we invoke a function *is_anomaly(user)* with the user's ID as an argument.
  - If the user queries the database for an 11th time within an hour, the function will indicate that access as an anomaly

How do we set the threshold?

https://github.com/oreilly-mlsec/book-resources/blob/master/chapter3/ids_heuristics_a.py

3.Objectives for an optimal anomaly detection system

- Low false positives and false negatives
- Easy to configure, tune, and maintain
- Adapts to changing trends in the data
  - Seasonality is the tendency of data to show regular patterns due to natural cycles of user activity (e.g., low activity on weekends)
- Works well across datasets of different nature
- Resource - efficient and suitable for real-time application
- Explainable alerts

4.Feature Engineering for Anomaly Detection

- **As with any other task in machine learning, selecting good features for anomaly detection is of paramount importance**
- **We focus our feature engineering discussions on three domains:**
    - host intrusion detection,
    - network intrusion detection, and
    - web application intrusion detection.

a.Host Intrusion Detection

- Developing an intrusion detection agent for hosts (*e.g., servers, desktops, laptops, embedded systems*),
  - you will likely need to generate your own metrics and might even want to perform correlations of signals collected from different sources.
- Basic system- and network-level statistics make for a good starting point

Some common signals of malwares that you can collect for features:

- *Running processes*
- *Active/new user accounts*
- *Kernel modules loaded*
- *DNS lookups*
- *Network connections*
- *System scheduler changes*
- *Daemon/background/persistent processes*
- *Startup operations, launchd entries*
- *OS registry databases, .plist files*
- *Temporary file directories*
- *Browser extensions*

Host Intrusion Detection: query

- We'll take a look at *osquery*, a popular OS instrumentation framework that collects and exposes low-level OS metrics,
  - making them available for querying through a SQL-based interface.
- Making scheduled queries through *osquery* can allow you to establish a baseline of host and application behavior,
  - thereby allowing the intrusion detector to identify suspicious events that occur unexpectedly

```sql
SELECT * FROM processes WHERE on_disk = 0;
```

Suppose that this query generates some data that looks like this:

```
2017-06-04T18:24:17+00:00        []
2017-06-04T18:54:17+00:00        []
2017-06-04T19:24:17+00:00        ["/tmp/YBBHNCA8J0"]
2017-06-04T19:54:17+00:00        []
```

b.Network Intrusion Detection

- Almost all forms of host intrusion instigate communication with the outside world.
- Most breaches are carried out with the objective of stealing some valuable data from the target,
  - *so it makes sense to detect intrusions by focusing on the network.*

Network Intrusion Detection

- For *botnets*,
  - remote command-and-control servers communicate with the compromised "zombie" machines to give instructions on operations to execute.
- For *APTs*,
  - hackers can remotely access the machines through a vulnerable or misconfigured service, allowing them shell and/or root access.
- For *adware*,
  - communication with external servers is required for downloading unsolicited ad content.
- For *spyware*,
  - results of the covert monitoring are often transmitted over the network to an external receiving server.

Network Intrusion Detection

- Network intrusion detection tools operate on the basic concept of inspecting traffic that passes between hosts.
- *Snort* is a popular open source IDS that sniffs packets and network traffic for realtime anomaly detection
  - It is the de facto choice for intrusion-detection monitoring, providing a good balance of usability and functionality

Network Intrusion Detection

- In extracting features for network intrusion detection, there is a noteworthy difference between *extracting* network traffic metadata and *inspecting* network traffic content.
  - The former is used in stateful packet inspection (SPI), working at the network and transport layers—OSI layers 3 and 4—and examining each network packet's header and footer without touching the packet context

Network Intrusion Detection: Deep packet inspection

- *Deep packet inspection (DPI)* is the process of examining the data encapsulated in network packets, in addition to the headers and footers

- This allows for the collection of signals and statistics about the network correspondence originating from the application layer

    - DPI is capable of collecting signals that can help detect spam, malware, intrusions, and subtle anomalies

Network Intrusion Detection: Deep packet inspection

- Bro: the earliest systems that implemented a passive network monitoring framework for network intrusion detection

- You can use *Bro* to detect suspicious activity in web applications by inspecting the strings present in the POST body of HTTP requests.

- For example, you can detect *SQL injections* and *cross-site scripting (XSS)* reflection attacks by creating a profile of the POST body content for a particular web application entry point

Features for network intrusion detection

- The Knowledge Discovery and Data Mining Special Interest Group (SIGKDD) from ACM
  - It holds the KDD Cup every year, posing a different challenge to participants.
- In 1999, the topic was “*computer network intrusion detection*”
  - the task was to “learn a predictive model capable of distinguishing between legitimate and illegitimate connections in a computer network.”
- This artificial dataset is very old and has been shown to have significant flaws,
  - but the list of derived features provided by the dataset is a good source of example features to extract for network intrusion detection in your own environment

c. Web Application Intrusion Detection

- Inspecting *HTTP server logs*
  - can provide you with a similar level of information and
  - is a more direct way of obtaining features derived from web application user interactions.
- Standard web servers like *Apache, IIS, and Nginx* generate logs in the NCSA Common Log Format, also called access logs.

Web Application Intrusion Detection

- NCSA combined logs and error logs also record information about
  - the client's user agent, referral URL, and any server errors generated by requests
- Here is an example:
  - a record in the combined log format that includes the requestor's user agent and referral URL

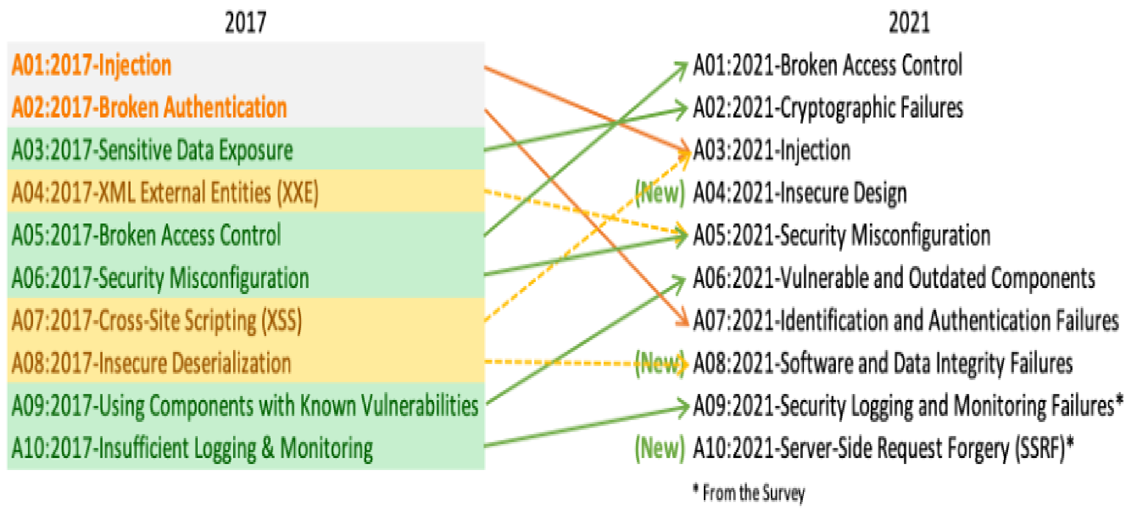
```
123.123.123.123 - jsmith [17/Dec/2016:18:55:05 +0800] "GET /index.html HTTP/1.0"
200 2046 "http://referer.com/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.17.3)
AppleWebKit/536.27.14 (KHTML, like Gecko) Chrome/55.0.2734.24 Safari/536.27.14"
```

Web Application Intrusion Detection

- Features extracted form log files:
  - IP-level access statistics,
  - URL string aberrations,
  - Decoded URL and HTML entities,
  - escaped characters,
  - null-byte string termination,
  - Unusual referrer patterns,
  - Sequence of accesses to endpoints,
  - User agent patterns.

Web Application Intrusion Detection

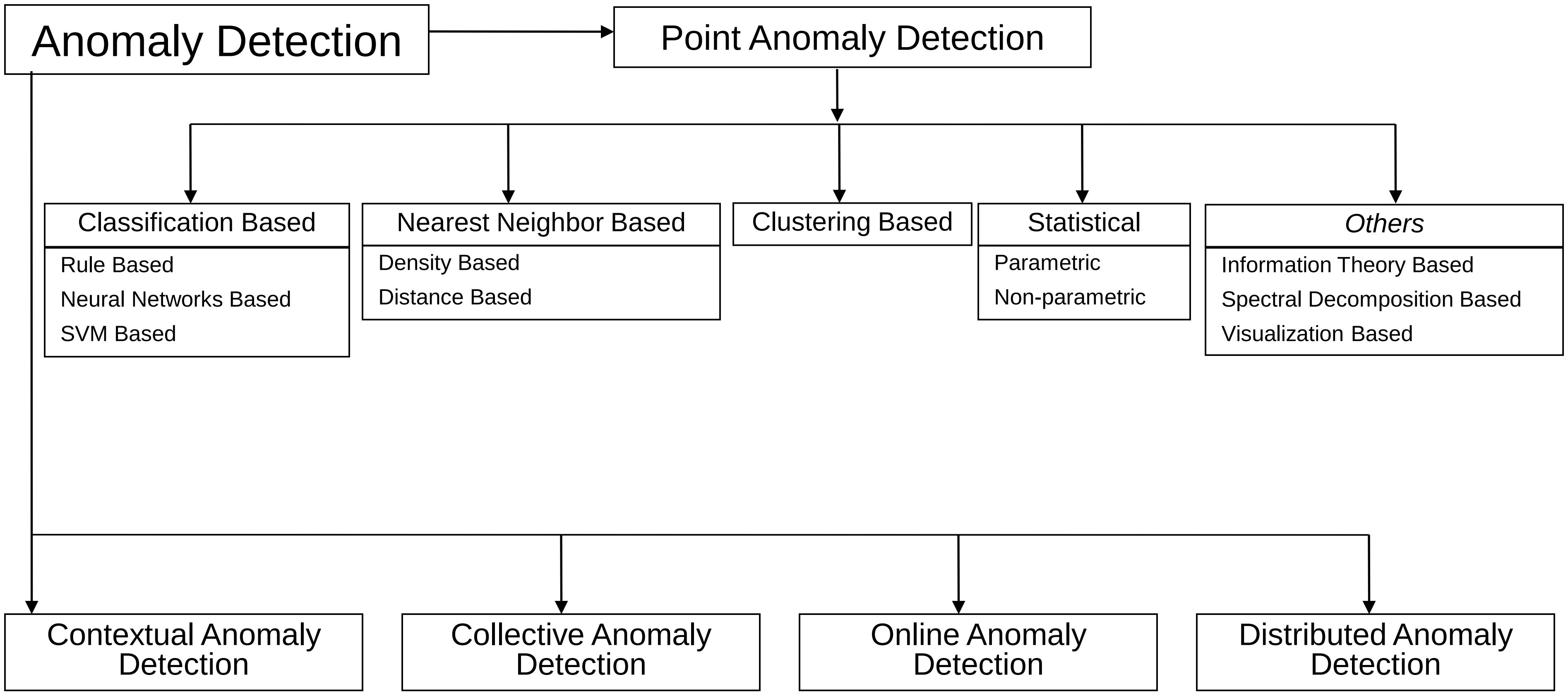
- Web logs provide enough information to detect different kinds of attacks on web applications, including, but not limited to,
  - XSS,
  - Injection,
  - CSRF,
  - Insecure Direct Object References,
  - etc



5.Anomaly Detection with Data and Algorithms

- Forecasting (supervised machine learning)
- Statistical metrics
- Unsupervised machine learning
- Goodness-of-fit tests
- Density-based methods

Taxonomy*



* Anomaly Detection – A Survey, Varun Chandola, Arindam Banerjee, and Vipin Kumar, To Appear in ACM Computing Surveys 2008.

Forecasting (Supervised Machine Learning)

- Forecasting is a highly intuitive way of performing anomaly detection:
  - we learn from prior data and make a prediction about the future
- We can consider any substantial deviations between the forecasts and observations as anomalous
- This class of anomaly detection algorithms uses *past data* to predict current data, and measures how different the currently observed data is from the prediction

https://github.com/oreilly-mlsec/book-resources/blob/master/chapter3/arima-forecasting.ipynb

Forecasting (Supervised Machine Learning)

- In forecasting, it is important to define the following descriptors of time series
  - Trends
  - Seasons
  - Cycles

https://github.com/oreilly-mlsec/book-resources/blob/master/chapter3/arima-forecasting.ipynb

Forecasting (Supervised Machine Learning): ARIMA

- Using the ARIMA (autoregressive integrated moving average) family of functions is a powerful and flexible way to perform forecasting on time series.
- Autoregressive models are a class of statistical models that have outputs that are linearly dependent on their own previous values in combination with a stochastic factor.

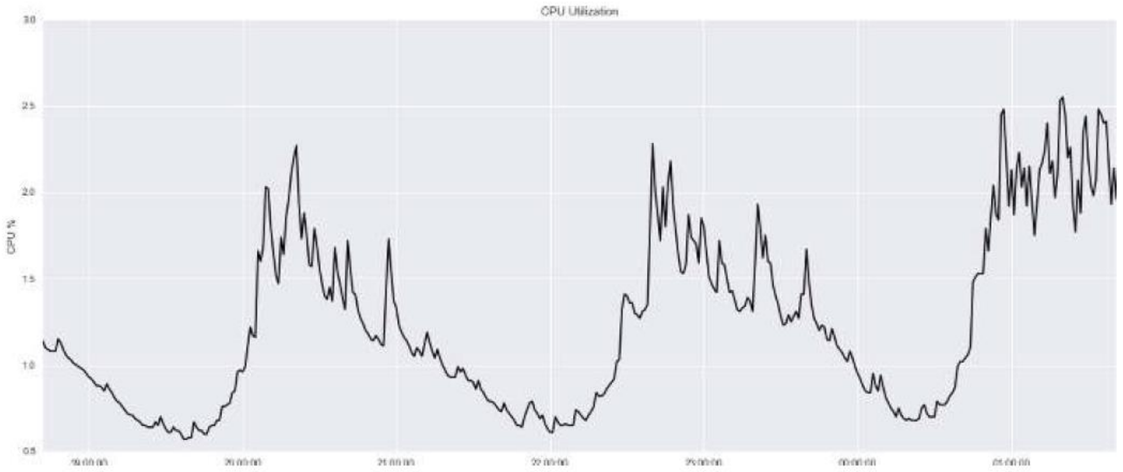


Figure 3-2. CPU utilization over time

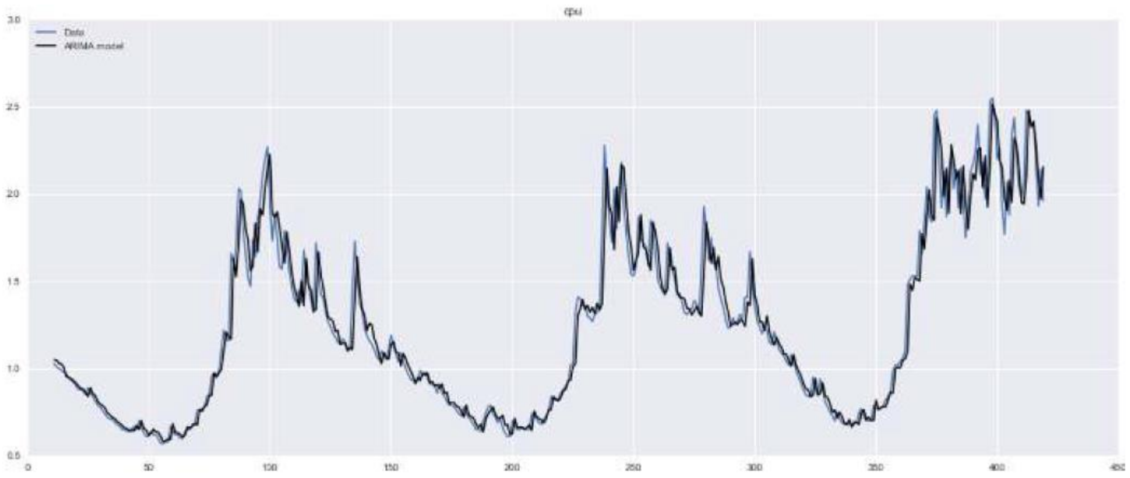


Figure 3-3. CPU utilization over time fitted with ARIMA model prediction

https://github.com/oreilly-mlsec/book-resources/blob/master/chapter3/arima-forecasting.ipynb

Forecasting (Supervised Machine Learning): ARIMA

- ARIMA(p,d,q)
  - p = the number of autoregressive terms
  - d = the number of nonseasonal differences
  - q = the number of moving-average terms

The differencing (if any) must be *reversed* to obtain a forecast for the original series:

$$\text{If } d = 0: \quad \hat{Y}_t = \hat{y}_t$$

$$\text{If } d = 1: \quad \hat{Y}_t = \hat{y}_t + Y_{t-1}$$

$$\text{If } d = 2: \quad \hat{Y}_t = \hat{y}_t + 2Y_{t-1} - Y_{t-2}$$

- Let Y denote the *original* series
- Let y denote the *differenced* (stationarized) series

No difference $(d=0)$: $y_t = Y_t$

First difference $(d=1)$: $y_t = Y_t - Y_{t-1}$

Second difference $(d=2)$: $y_t = (Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2})$

$= Y_t - 2Y_{t-1} + Y_{t-2}$

$$\hat{y}_t = \overbrace{\mu}^{\text{constant}} + \overbrace{\phi_1 y_{t-1} + \ldots + \phi_p y_{t-p}}^{\text{AR terms (lagged values of } y)}$$

$$\underbrace{-\theta_1 e_{t-1} \ldots - \theta_q e_{t-q}}_{\text{MA terms (lagged errors)}}$$

By convention, the AR terms are + and the MA terms are –

Not as bad as it looks! Usually $p+q \leq 2$ and either p=0 or q=0 (pure AR or pure MA model)

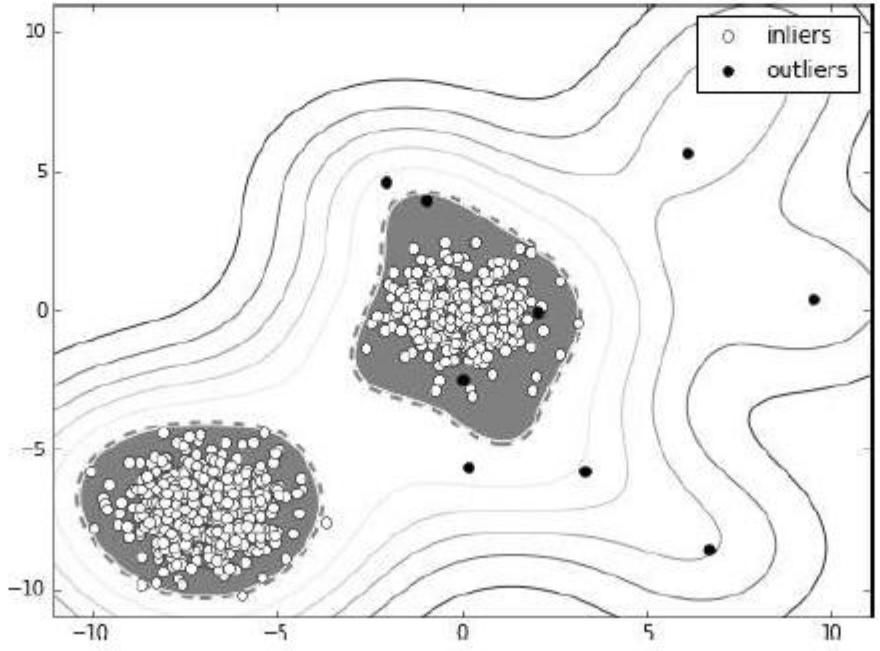
```python
import pandas as pd
import pyflux as pf
from datetime import datetime

# Read in the training and testing dataset files
data_train_a = pd.read_csv('cpu-train-a.csv',
    parse_dates=[0], infer_datetime_format=True)
data_test_a = pd.read_csv('cpu-test-a.csv',
    parse_dates=[0], infer_datetime_format=True)

# Define the model
model_a = pf.ARIMA(data=data_train_a,
                   ar=11, ma=11, integ=0, target='cpu')

# Estimate latent variables for the model using the
# Metropolis-Hastings algorithm as the inference method
x = model_a.fit("M-H")

# Plot the fit of the ARIMA model against the data
model_a.plot_fit()
```

Unsupervised Machine Learning Algorithms

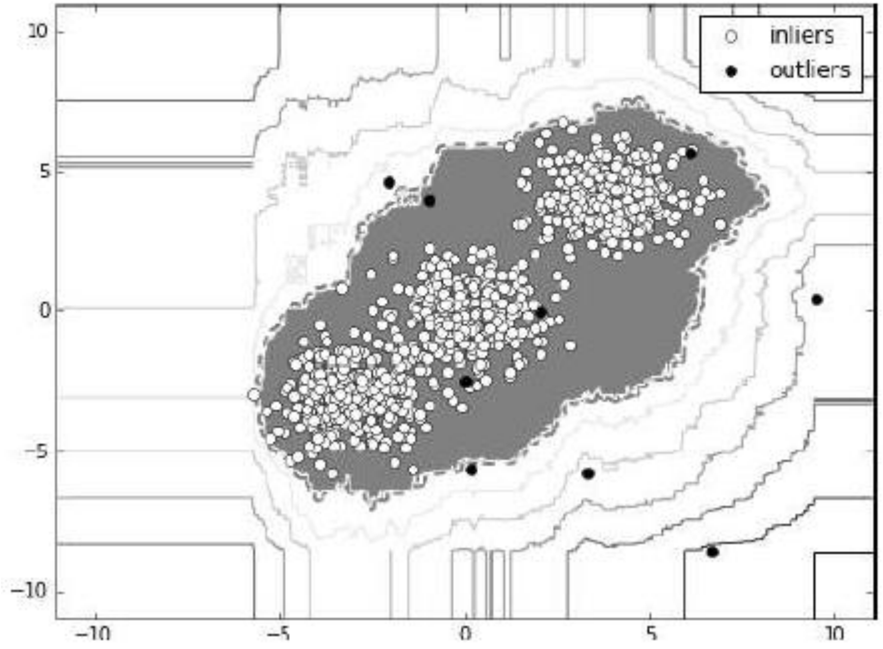
- Supervised machine learning classifiers are typically used to solve problems that involve two or more classes.
- However, when used for anomaly detection, the modifications of these algorithms give them characteristics of unsupervised learning
- One-class SVM is an unsupervised algorithm that learns a decision function for novelty detection: *classifying new data as similar or different to the training set*

https://github.com/oreilly-mlsec/book-resources/blob/master/chapter3/one-class-svm.ipynb

```python
from sklearn import svm

classifier = svm.OneClassSVM(nu=0.99 * outlier_ratio + 0.01,
                             kernel="rbf",
                             gamma=0.1)
classifier.fit(x)
y_pred = classifier.predict(x)
num_errors = sum(y_pred != labels)
print('Number of errors: {}'.format(num_errors))
```




```python
from sklearn.ensemble import IsolationForest

rng = np.random.RandomState(99)

classifier = IsolationForest(max_samples=num_samples,
                             contamination=outlier_ratio,
                             random_state=rng)
classifier.fit(x)
y_pred = classifier.predict(x)
num_errors = sum(y_pred != labels)
print('Number of errors: {}'.format(num_errors))
```



Density-Based Methods

- Density-based methods are well suited for high-dimensional datasets,
  - which can be difficult to deal with using the other classes of anomaly detection methods.
- The main idea behind all of them is to form a cluster representation of the training data, under the hypothesis that: *outliers or anomalies will be located in low-density regions of this cluster representation*.

https://github.com/oreilly-mlsec/book-resources/blob/master/chapter3/local-outlier-factor.ipynb

Density-Based Methods

- *k-NN* is commonly considered a density-based method and is actually quite a popular way to measure the probability that a data point is an outlier.
- We can also use k-means clustering for anomaly detection in a similar way, using distances between the point and centroids as a measure of sample density.

https://github.com/oreilly-mlsec/book-resources/blob/master/chapter3/local-outlier-factor.ipynb

6.Challenges of Using Machine Learning in Anomaly Detection

- Because of the high cost of classification errors, fully automated, end-to-end anomaly detection systems that are powered purely by machine learning are very rare
  - there is almost always a human in the loop to verify that alerts are relevant before any action is taken on them
- The semantic gap is a real problem with machine learning in many environments

7. Practical System Design Concerns

- Optimizing for Explainability
  - the semantic gap of alert explainability. **Real issue!!!**
  - More complex machine learning models can fit real-world data better, but they are very black-box—the decision-making processes are completely opaque to an external observer.
- Performance and scalability in real-time streaming applications
  - Using distributed machine learning libraries such as Apache Spark Mllib can help