

# Hands-on Machine Learning for Cybersecurity

James Walden<sup>1</sup>

<sup>1</sup>Center for Information Security  
Northern Kentucky University

11th Annual NKU Cybersecurity Symposium  
Highland Heights, KY  
October 11, 2018

# Topics

Introduction

Building a Model

A Machine Learning Algorithms

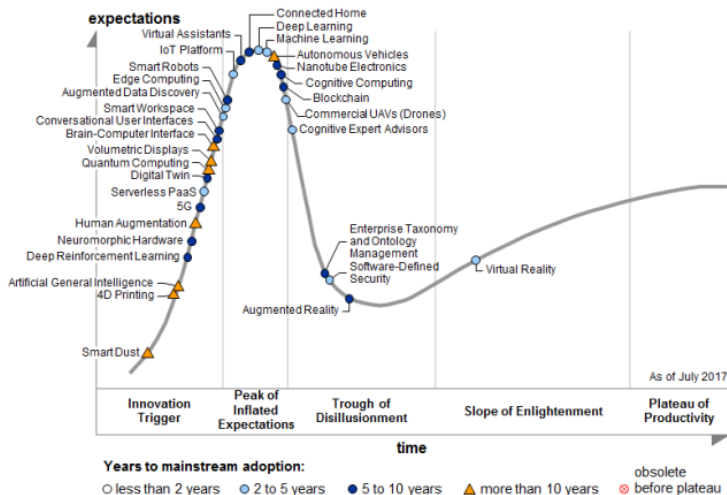
Machine Learning with Python

Using scikit-learn

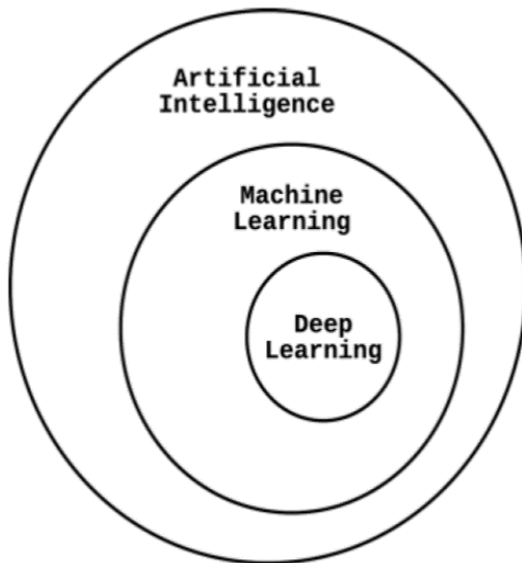
What's Next

References

# The Hype Cycle



# AI vs ML vs Deep Learning



# AI and ML Definitions

## Artificial Intelligence

Artificial intelligence is a term used to describe a system which perceives its environment and takes actions to maximize its chances of achieving its goals.

## Machine Learning

Machine learning is a set of techniques that enable computers to perform tasks without being explicitly programmed. ML systems generalize from past data to make predictions about future data.

# Machine Learning Formal Definition

## Machine Learning (Tom Mitchell, 1997)

“A computer program is said to learn from experience  $E$  with respect to some class of tasks  $T$  and performance measure  $P$  if its performance at tasks in  $T$ , as measured by  $P$ , improves with experience  $E$ .”

Experience	Task	Performance
E-mail message.	Identify phishing attempt	% correctly classified
Malware.	Categorize by threat actor	Coherent groupings
Login records.	Identify credential misuse	% verified misuse
Attack data.	Predict #attacks next year	Accurate #attacks

# Machine Learning Tasks

## Supervised Learning

Supervised learning focuses on models that predict the probabilities of new events based on the probabilities of previously observed events.

Example task: determine if a file is malware or not.

## Unsupervised Learning

Unsupervised learning models attempt to find patterns in data.

Example task: determine how many families of malware exist in dataset and which files belong to each family.

# Supervised Learning

## Classification

Classification algorithms predict which category an input belongs to based on probabilities learned from previously observed inputs. Example task: determine if a file is malware or not.

## Regression





Regression models predict a continuous output value for a given input based on the output values associated with previous inputs. Example task: predict how many malware samples will be seen next month.

We will focus on classification models.

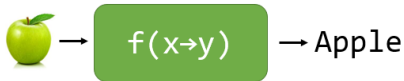


# Classification

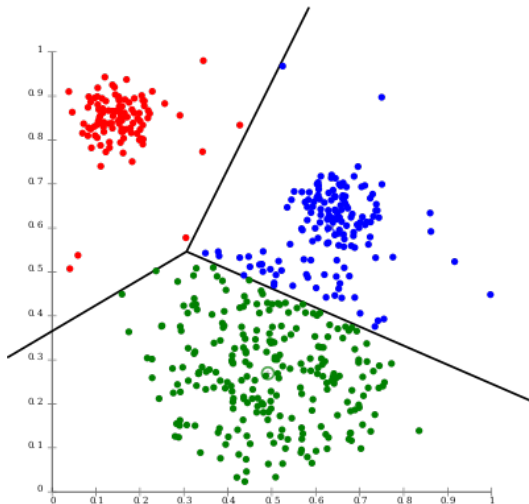
## Training Data

Sample (X)	Label (Y)
	Apple
	Orange
	Apple
	Orange

## Resulting Model



# Unsupervised Learning



# Machine Learning in Security

What questions can machine learning answer for us in security?

- Is this credit card transaction fraudulent?
- Is this e-mail message spam or not?
- Is this e-mail message a phishing attempt?
- Does this file contain malware?
- Does this login attempt result from a compromised password?
- Is this inbound network packet of a DoS attack?
- Is this outbound packet calling back to a C2 server?

# Hands-on Machine Learning for Cybersecurity?

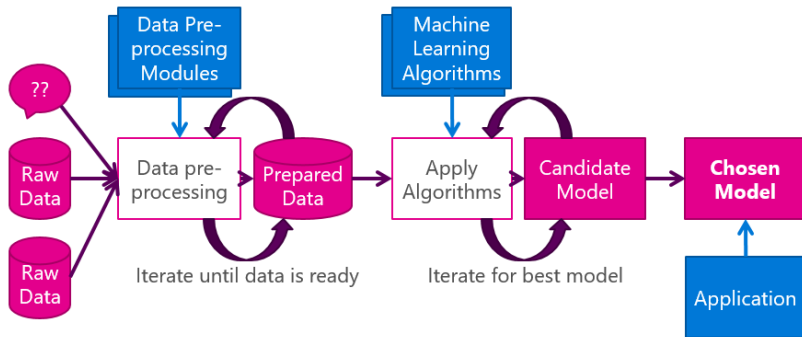
What does this workshop cover?

- Basic ML concepts
- ML workflow
- How to apply ML algorithms in Python

What doesn't it cover?

- Theory and mathematics behind ML algorithms
- Unsupervised learning
- Deep learning and neural networks
- Data-specific techniques like text mining or network analysis

# Machine Learning Process



## Building a Model

1. **Collect** samples of data from both classifications to train the machine learning model.
2. **Extract** features from each training example to represent the example numerically.
3. **Train** the machine learning system to identify bad items using the features.
4. **Test** the system on data that was not used when training to evaluate its performance.

## Collecting Data

Machine learning systems are only as good as their training data.

1. Training data should be as close to the data being test as possible.
  - Good examples should be taken from actual files, logs, etc. on your systems.
  - Bad examples should be taken from actual malware, indicators of compromise, etc.
  - If a specific threat actor is of interest, then taking bad examples from their activities is best.
2. Having close to equal numbers of bad and good items is better.
3. More training data is better.
4. Systems need to be retrained as the threats they face change.

# Extracting Features

## E-mail

Hi James

Do you need to go to Walmart ? Take that  
gift card below now:

Download it now

Walmart Team



Feature	Count
do	1
gift	1
go	1
card	1
James	1
Walmart	2
?	1

## URL

[http://admin.jablum.cz/files/2914d7d2a19  
d2EyWE=/customer\\_center/customer-  
IDPP00C741/myaccount/signin/](http://admin.jablum.cz/files/2914d7d2a19d2EyWE=/customer_center/customer-IDPP00C741/myaccount/signin/)



Feature	Count
URL Length	102
/ count	6
= count	1
? count	0
& count	0



# Extracting Features

- Feature selection is guided by expert knowledge.
- There should be more samples than features.
- Feature values should not be close to constant.
- Strongly correlated features can cause problems for some algorithms.

# Training

For each sample, provide training interface with

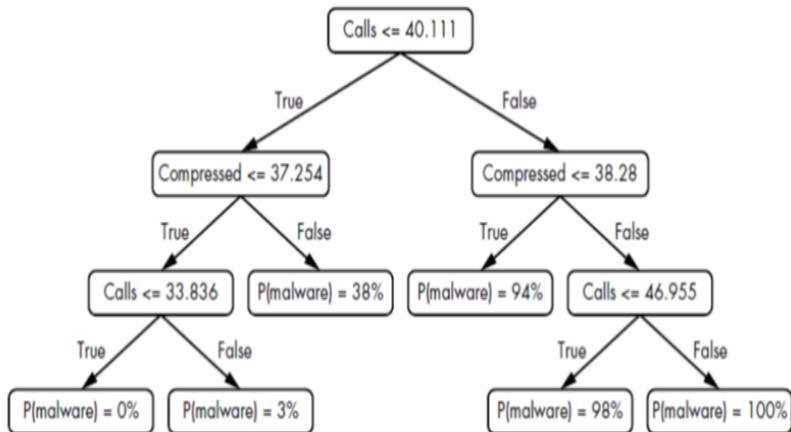
- Feature values for sample.
- Classification of sample as good or bad.

# Testing

Classify data not used in training with model.

		Predicted class	
		$P$	$N$
Actual Class	$P$	True Positives (TP)	False Negatives (FN)
	$N$	False Positives (FP)	True Negatives (TN)

# Decision Trees



## Comparing with other Algorithms

### Advantages

- Decision trees can be interpreted by humans.
- Can be combined with other techniques.

### Disadvantages

- Relatively inaccurate compared to other algorithms.
- A small input change can result in a big change in the tree.

# scikit-learn



<http://scikit-learn.org>

- Efficient user-friendly machine learning toolkit
- Built on NumPy, SciPy, and matplotlib
- Open source with BSD license

# SciPy



<https://www.scipy.org>

- Scientific computing library build on NumPy
- Sparse matrices and graphs
- Optimization and interpolation
- Signal processing and Fourier transforms

# NumPy



<http://www.numpy.org/>

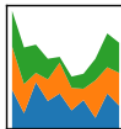
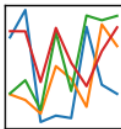
- Space-efficient n-dimensional arrays
- Fast vector operations
- Tools for integrating C/C++ and Fortran code
- Linear algebra functions



# Pandas

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



<https://pandas.pydata.org/>

- Python data science library built on NumPy
- Provides user friendly Data Frames like R
- Statistical and data visualization functions

## Matplotlib



<https://matplotlib.org/>

- Python 2D plotting library for publication quality graphics.
- The `pyplot` module provides a MATLAB-like interface for simple plots.
- User has full control of all plotting details.
- Used as basis of Pandas plotting abilities.

# IPython

**IP[y]:** IPython  
Interactive Computing

<https://ipython.org/>

- A powerful, interactive Python shell.
- Use shell commands and Python code in same interface.
- Used as computation kernel by Jupyter.

# Jupyter



<https://jupyter.org/>

- Interactive notebooks for data science in many languages.
- Combine Markdown text, computation results, and graphics in a single document.
- Similar to Mathematica notebooks or RStudio documents.
- Uses a web interface.

# Anaconda



<https://www.anaconda.com/>

- Most popular Python data science distribution.
- Comes with scikit-learn, pandas, scipy, numpy, etc.
- Uses conda package management tool.
- Create environments with different versions of libraries.

## conda commands

```
conda list                # list installed pkgs
conda install pkgname     # install package
conda update pkgname      # upgrade package
conda update -n base conda # update conda
conda update python       # update python
```

# Import Libraries

These are libraries that we will need regardless of ML algorithm.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import
    train_test_split
from sklearn.metrics import accuracy_score,
    confusion_matrix
```

## Load Data

Read the CSV data as a Pandas data frame.

```
In [5]: df = pd.read_csv('data.csv')
```

```
In [6]: df.shape
```

```
Out[6]: (1372, 5)
```

```
In [7]: df.head(3)
```

```
Out[7]:
```

	Variance	Skewness	Kurtosis	Entropy	Forgery
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0

Data frames are preferred for exploring the data.



## Convert Data Frame to Numpy Array

Scikit-learn expects labels as a vector, features as an array.

```
In [5]: y = df['Forgery'].values
```

```
In [6]: y.shape
```

```
Out [6]: (1372,)
```

```
In [7]: X = df.drop('Forgery', axis=1).values
```

```
In [8]: X.shape
```

```
Out [8]: (1372, 4)
```

## Split the Data

Choose 80% of the data to train the model and 20% to test it.

- Samples (rows) are chosen randomly.
- Set `random_state` to make split always the same.

```
In [14]: X_train, X_test, y_train, y_test =  
         train_test_split(X, y, test_size=0.2,  
                         random_state=1)
```

```
In [15]: X_train.shape
```

```
Out[15]: (1097, 4)
```

```
In [16]: X_test.shape
```

```
Out[16]: (275, 4)
```

```
In [17]: y_train.shape
```

```
Out[17]: (1097,)
```

```
In [18]: y_test.shape
```

```
Out[18]: (275,)
```

## Train the Model

Create a classifier object, then fit it.

```
In [19]: from sklearn.tree import  
         DecisionTreeClassifier  
  
In [21]: model = DecisionTreeClassifier()  
  
In [22]: model.fit(X_train, y_train);
```

The class names and model creation method names change, but we always use the `fit` method with the training features + labels.

## Evaluate the Model

We make predictions using the `predict()` method.

```
In [23]: y_pred = model.predict(X_test)
```

then compare the predicted labels with the actual labels to measure accuracy.

```
In [24]: accuracy_score(y_pred, y_test)  
Out [24]: 0.9745454545454545
```

Our model predicts forged bank notes with 97.5% accuracy.

## Confusion Matrix

For more detailed model performance, we use the confusion matrix.

		Predicted class	
		<i>P</i>	<i>N</i>
Actual Class	<i>P</i>	True Positives (TP)	False Negatives (FN)
	<i>N</i>	False Positives (FP)	True Negatives (TN)

```
In [27]:
    confusion_matrix(
        y_pred, y_test)
Out[27]:
array([[153,    3],
       [  4, 115]])
```

Decision tree had 3 false negatives, 4 false positives.

# What's Next

We have a hands-on activity next, in which we

- will log into a Linux VM with Anaconda installed,
- start a Jupyter notebook server,
- use a notebook to solve the bank note problem, and
- experiment with a few machine learning algorithms.

## Accessing the Data Science VM

- Go to `view.nku.edu` in your browser.
- Login with
  - Username: nku  
username from hardcopy
  - Password: password from hardcopy
- Open the COI Labs VM pool.
- Select a Windows 10 desktop.
- Open VMware Workstation.
- Go to `coivcenter.hh.nku.edu` in browser.
- Login to VCenter with
  - Username: nku  
username from hardcopy
  - Password: password from hardcopy
- Select the Data Science VM.
- Login with
  - Username: dsc
  - Password: securityml

## References

1. Clarence Chio and David Freeman, *Machine Learning and Security: Protecting Systems with Data and Algorithms*, O'Reilly Media, 2018.
2. Aurélien Géron, *Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, O'Reilly Media, 2017.
3. Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, *An Introduction to Statistical Learning with Applications in R*, Springer. 2014.
4. Andreas C Müller, Sarah Guido, et. Al, *Introduction to Machine Learning with Python: a Guide for Data Scientists*, O'Reilly Media, 2016.
5. Joshua Saxe and Hillary Sanders, *Malware Data Science: Attack Detection and Attribution*, No Starch Press, 2018.