GTK CYBER

GET ON THE FAST TRACK

Applied Data Science for Cyber Security Professionals

www.gtkcyber.com

Join the Slack Channel at: http://bit.ly/30IvONs

Course Agenda

- Introduction: What is Data Science?
  - Overview of Machine Learning & Cyber Applications
- Manipulating and Exploring Data
  - Exploratory Data Analysis in 1 Dimension
  - Exploratory Data Analysis in 2 Dimensions
- Overview of Machine Learning
- Hunting with Data Science

Expectations

- Please participate and **ask questions.**
- Please follow along and **TRY OUT** the examples yourself during the class
- All the answers are in the slide decks or GitHub repository, but please try to complete the exercises **without looking at the answers**.
- Join the conversation in slack!
- Have fun!

Introduction

Our Lawyers Make Us Say This

All materials presented in this training and those provided as an adjunct to the program are copyrighted 2019 by GTK Cyber LLC.

They are intended solely for the use of registered program participants and may not be reproduced or redistributed in any manner for any other reason.

Austin Taylor, CISSP

- Director, Cybersecurity R&D @ IronNet Cybersecurity
- Cyber Warfare Operator @ USAF (MDANG)
- Projects: Flare, Bluewall, VulnWhisperer
- Publications: Build A World Class Monitoring
- System for Enterprise, Small Office, or Home
- SANS Instructor - Continuous Monitoring
- GTK Cyber - Instructor

@HuntOperator

www.austintaylor.io

austin.taylor@gtkcyber.com

What is Data Science?

Data Science is the automated extraction of **information** from **raw data**.

Data Science is the art of **turning data into actions**. This is accomplished through the creation of data products, which provide actionable information without exposing decision makers to the underlying data or analytics

Booz Allen Hamilton, Field Guide to Data Science, Pg. 17

Analyst ⟷ Developer

Analyst + Developer

What Data Science is Not

What Data Science is

Research Process

Research Process

"The term "data scientist" will subside and may well sound dated five years from now. **The skills will become more commonplace and commoditized. When that happens, the real boom will begin**, because the technology will become widely adopted and thus more useful. … **Instead, we need self-service tools that empower smart and tenacious business people to perform Big Data analysis themselves.**

–Andrew Brust, "Data scientists don't scale", http://www.zdnet.com/article/data-scientists-dont-scale/

Time to Insight

Time to Insight

Time = $$

Value of Insights over Time

Cost of Insights over Time

Cost of Insights over Time

What skills does a data scientist need?

Computer Science
Math
Domain Expertise

Computer Science
Business Sense
Math
Data Visualization
Domain Expertise
Communication Skills

The Data Scientist Venn Diagram

https://www.kdnuggets.com/2017/03/data-science-data-scientist-do.html

Data Scientists spend
50-90% of their time being…

GTK CYBER

GET ON THE FAST TRACK

gtkcyber.com

Triage
Analysis
Reception
Decision
Action
Hypothesis

By the end of the class you will be able to:

- Quickly and effectively prepare data for analysis
- Apply machine learning techniques to enhance security

I THINK WE SHOULD BUILD AN SQL DATABASE.
UH-OH
E-mail: SCOTTADAMS@AOL.COM
S. Adams
DOES HE UNDERSTAND WHAT HE SAID OR IS IT SOMETHING HE SAW IN A TRADE MAGAZINE AD?
© 1995 United Feature Syndicate, Inc.(NYC)
WHAT COLOR DO YOU WANT THAT DATABASE?
I THINK MAUVE HAS THE MOST RAM.

Buzzword

Recommended Reading

Recommended Reading

Recommended Reading

Recommended Reading

http://datadrivensecurity.info

Machine Learning

What is
Machine Learning (ML)
Artificial Intelligence (AI)

“I have no idea. The latest buzzword? ”

–The guy in the back of the class

“Machine learning is the science of getting computers to **act without being explicitly programmed.** ”

– *https://www.coursera.org/course/ml*

"A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E."

–Tom Mitchell, Carnegie Mellon University

"Machine learning explores the construction and study of algorithms that can learn from and **make predictions on data.** Such algorithms operate by building a model from example inputs in order to make data-driven predictions or decisions, **rather than following strictly static program instructions.**"

–*https://en.wikipedia.org/wiki/Machine_learning*

SPAM
CHOPPED PORK AND HAM
LICENCED BY:
Hormel
Foods
Serving Suggestion

GTK CYBER

GET ON THE FAST TRACK

gtkcyber.com

- Blacklists
- Simple keyword matching
- Naive Bayesian Classifiers
- Deep Learning

Artificial Intelligence

Any technique which enables computers to mimic human behavior.

Machine Learning

Subset of AI techniques which use statistical methods to enable machines to improve with experiences.

Deep Learning

Subset of ML which make the computation of multi-layer neural networks feasible.

https://www.kdnuggets.com/2017/07/rapidminer-ai-machine-learning-deep-learning.html

@katherinebailey Because marketing? Every time someone calls simple linear regression “AI” Gauss turns over in his grave.

Machine Learning Problems

- **Supervised Learning:** Supervised Learning is a class of Machine Learning in which a model is "trained" using a set of pre-existing labeled data.

- **Unsupervised Learning:** A class of Machine Learning algorithms in which a model is built without the use of labeled data.

Machine Learning Problem Types

- **Classification**: Assigning or predicting a observation's membership in discrete class
- **Regression:** Predicting a continuous value based on the observations' features
- **Clustering:** Identifying groupings within a dataset
- **Dimensionality Reduction:** Reducing the number of variables in a feature set

What Problem am I solving?

What it is not

Path to Artificial Intelligence?

- Hard-coding knowledge failed (e.g. Cyc 1989).
- **Machine Learning: Gathering knowledge from experience** eliminates the need to formally specify all knowledge the computer needs.
- **AI** needs to acquire own knowledge by **extracting pattern from raw data.**
- Success depends heavily on **representation of data, aka features.**

http://www.deeplearningbook.org/

Applications to Security

Regression Example

Server Capacity Prediction: Regression analysis can be used to predict a server's capacity (or CPU usage) based on the server's historical performance.

https://www.researchgate.net/publication/256645877_LiRCUP_Linear_Regression_based_CPU_Usage_Prediction_Algorithm_for_Live_Migration_of_Virtual_Machines_in_Data_Centers

https://jgreenemi.com/predicting-capacity-with-linear-regression-ml/

Clustering Example

Anomaly Detection: Clustering techniques can be used to detect anomalous traffic or loads or anything really.

Network-Based Intrusion Detection

- Derive Features from Network Traffic Captures “pcap” at packet level or NetFlow level (tools: tshark, tcpdump, bro…)
- Example Features based on header information: 2s-windowing of connections > duration, protocol, src and dat bytes, service.
- Get data sets: http://www.netresec.com/?page=PcapFiles, https://maccdc.org/, http://www.westpoint.edu/crc/SitePages/DataSets.aspx http://www.unb.ca/cic/research/datasets/,

Malware Detection/Classification

- Derive Features from Binary Content and metadata manifest (function calls, string obtained from IDA Disassembler)
- Example Features: opcode count (n-grams), segment count, asm pixel intensity, n-gramming of bytes, function name.
- Featureless Deep Learning with word2vec embedding
- Get open source malware samples: Vx Heaven, Virus Share, Maltrieve, Open Malware

Security Applications of Machine Learning

- Domain Generation Algorithm (DGA) Detection (Classification)
- Malicious URL Detection (Classification)
- Network Traffic: Beaconing Detection (Classification/Clustering)
- Detection of new classes of malware (Classification/Clustering)
- General Network Traffic Anomaly Detection (Classification/Clustering)
- Log Analysis - Anomaly Detection (Classification/Clustering)
- Phishing Detection (Classification)
- Identifying SQL Injection (Classification)
- Identifying XSS cross-site scripting (Classification)
- DOS/DDOS Detection (Classification)
- Authentication (Classification)

Data Science Hunting Funnel

http://www.austintaylor.io/network/traffic/threat/data/science/hunting/funnel/machine/learning/domain/expertise/2017/07/11/data-science-hunting-funnel/

The Machine Learning Process

Supervised Machine Learning Process

Unsupervised Machine Learning Process

First, define your analytic question.

What are you trying to do?

How do you define success?
What are you measuring?

Choose data sources

- What is available?
- Is it enough?
- Is the data reliable/clean/consistent?
- What other data could you use?

Other Considerations

- Policies
- Legal contraints
- Biases in Data
- Latency
- Data size

Gather and Explore Your Data

Is the data good enough?

What are the rules governing its use?

Do I have enough?

Do problems or biases exist in the data that could cause problems?

Feature Engineering

- Define what you are trying to measure. These will become the observations or rows of your final dataset

- Define how you will mathematically represent your data. This will be come the features or columns of your final dataset.

Feature Engineering

Feature	Value
Color	Gray
Fins	7
Predator	TRUE

Feature Engineering

Feature	Value
Color	Gray
Fins	7
Predator	TRUE

Feature Engineering

Feature	Value
Color	Gray
Fins	7
Predator	TRUE
Mammal	**TRUE**

Build and Tune your Model

- Believe it or not, this is the easy part.
- Most of this is **done using libraries** like scikit-learn, mllib, tensorflow, caret or keras, and **many steps can be automated**.
- You can even do it in Splunk.

Evaluate Performance

- Use various scoring methods, or write your own to determine model performance.
- Go back to step 1 and repeat! (Do not pass go, do not collect $200)

Discussion

- Consider that you are building a system to identify fraudulent credit card transactions. What are some features that you would want to capture?

- What are the challenges you could foresee in this problem?

The Python Data Science Ecosystem

Machine Learning Ecosystem

- **Data Gathering:** Pandas, Drill, BeautifulSoup, PyDBAPI, PyDAL, Boto3
- **Feature Extraction:** Pandas, NumPy, Featuretools
- **Machine Learning**
  - **"Regular" ML:** Scikit-learn (sklearn), h2o, mllib (PySpark)
  - **Deep Learning:** Tensorflow, Keras, Theano, Caffe, PyTorch
- **Visualization:** Matplotlib, Seaborn, Yellowbrick, LIME, ggplot, plot.ly,

Recommended Reading

Recommended Reading

Recommended Reading

http://www.deeplearningbook.org/

O'REILLY
Machine Learning & Security
PROTECTING SYSTEMS WITH DATA AND ALGORITHMS
Clarence Chio & David Freeman

gtkcyber.com

O'REILLY
Learning
Apache Drill
QUERY AND ANALYZE STRUCTURED DATA
Charles Givre & Paul Rogers

The Virtual Machine: Griffon

```
File  Edit  View  Search  Terminal  Help
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/hbase/hbase-1.1.3/lib/slf4j-log4j12-1
.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4
j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
2016-05-16 13:04:54,887 WARN  [main] util.NativeCodeLoader: Unable to load nativ
e-hadoop library for your platform... using builtin-java classes where applicabl
e
2016-05-16 13:05:11,827 ERROR [main] zookeeper.RecoverableZooKeeper: ZooKeeper exists faile
d after 4 attempts
2016-05-16 13:05:11,828 WARN  [main] zookeeper.ZKUtil: hconnection-0x46a145ba0x0, quorum=lo
calhost:2181, baseZNode=/hbase Unable to set watcher on znode (/hbase/hbaseid)
org.apache.zookeeper.KeeperException$ConnectionLossException: KeeperErrorCode = ConnectionL
oss for /hbase/hbaseid
        at org.apache.zookeeper.KeeperException.create(KeeperException.java:99)
        at org.apache.zookeeper.KeeperException.create(KeeperException.java:51)
        at org.apache.zookeeper.ZooKeeper.exists(ZooKeeper.java:1045)
        at org.apache.hadoop.hbase.zookeeper.RecoverableZooKeeper.exists(RecoverableZooKeep
er.java:221)
        at org.apache.hadoop.hbase.zookeeper.ZKUtil.checkExists(ZKUtil.java:541)
        at org.apache.hadoop.hbase.zookeeper.ZKClusterId.readClusterIdZNode(ZKClusterId.jav
a:65)
        at org.apache.hadoop.hbase.client.ZooKeeperRegistry.getClusterId(ZooKeeperRegistry.
java:105)
```

Do Data Science, Not Sysadmin

Built on Ubuntu MATE

Easy to Use

Programming Languages

- Languages
- Libraries
- Editors and Notebooks
- Databases + Administrative tools
- Big Data Tools
- Machine Learning Libraries
- Data Visualization

Scripting Languages

Griffon RC3 [Running]
Applications
Places
System
Accessories
Big Data
Databases
Data Visualization
Education
Graphics
Internet
Office
Other
Programming
Sound & Video
System Tools
Universal Access
passwords.txt
Shells
Atom
DBeaver CE
H2O.ai
IntelliJ IDEA Community Editior
Jupyter Notebook
PyCharm Community Edition
Rodeo
RStudio
Spyder
Julia
NodeJS
PHP Shell
Pig Shell
Python (v2.7)
Python (v3.5)
R
Ruby
Scala

Editors & Notebooks

GTK CYBER

GET ON THE FAST TRACK

gtkcyber.com

Atom

Project Merlin Beta 3 (Small updates) [Running]
Applications Places System
Mon Aug 15, 12:26
python_demo.py — /usr/share/atom/resources — Atom
File Edit View Selection Find Packages Help
resources
python_demo.py
import pandas as pd
df = pd.DataFrame([2,3,4,5,6,7,8])

GTK CYBER
GET ON THE FAST TRACK

Jupyter
Nb

gtkcyber.com

Jupyter Notebook

- Python 2 & 3
- R
- Ruby
- Scala
- PySpark

GTK CYBER

GET ON THE FAST TRACK

MySQL®

mongoDB®

PostgreSQL

neo4j

SQLite

gtkcyber.com

GTK CYBER
GET ON THE FAST TRACK

MySQL®

gtkcyber.com

GTK CYBER

GET ON THE FAST TRACK

mongoDB®

Applications Places System
Fri Aug 19, 01:27
db - Mongo Express - Mozilla Firefox
File Edit View History Bookmarks Tools Help
db - Mongo Express
localhost:8081/db/db
Search
Mongo Express
Database: db
Viewing Database: db
Collection "test" deleted!
Collections
Collection Name
Create collection
View
Export
[JSON]
system.indexes
Del

Applications
Places
System
Accessories
Big Data
Databases
Data Visualization
Education
Graphics
Internet
Office
Other
Programming
Sound & Video
System Tools
Universal Access
DBeaver CE
MongoDB Shell
MongoDB Web Interface
MySQL
MySQL Web Interface
MySQL Workbench
Neo4j Web Interface
PostgreSQL
PostgreSQL Web Interface
Sqlite
passwords.txt

GTK CYBER
GET ON THE FAST TRACK

merlin projectmerlin: /usr/local/hadoop/sbin

Edit View Terminal Help

```
erlinuser@project n:/usr/local/hadoop/sbin$ ./start-dfs.sh
16/05/11 23:23:39 WA il.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... ng builtin-java classes where applicable
Starting namenodes on [lo ost]
merlinuser@localhost's pas d:
merlinuser@localhost's passw localhost: Permission denied, please try again.

merlinuser@localhost's password: lhost: Permission denied, please try again.

localhost: Permission denied (public assword).
merlinuser@localhost's password:
merlinuser@localhost's password: localho ermission denied, please try again.
```

gtkcyber.com

Questions?

Using Jupyter Notebook

Using Jupyter Notebook

Open a notebook

Using Jupyter Notebook

Run a cell

Using Jupyter Notebook

Questions?

Module 1: Exploratory Data Analysis in One Dimension

Vectorized Data Structures

What are they?

Vectorized Data Structures let you **perform operations** on your data **all at once**

Advantages of Vectorized Data Structures

- More readable and simpler code
- Allows the module to optimize for CPU and Memory usage
- Pass serialized objects from one system to another

```
for x in range(0, len( data )):
    data[x] = data[x] + 1
```

```
for x in range(0, len( data )):
    data[x] = data[x] + 1
```

```
odds = evens + 1
```

F = M * A

Each **variable** is saved in its own **column**

&

Each **observation** is saved in its own **row**

No loops

Dimensions	Name	Description
1	Series	Indexed 1 dimensional data structure
1	Timeseries	Series using timestamps as an index
2	DataFrame	A two dimensional table
3	Panel	A three dimensional mutable data structure

Columns

rows

Regd. No	Name	Marks%
1000	Steve	86.29
1001	Mathew	91.63
1002	Jose	72.90
1003	Patty	69.23
1004	Vin	88.30

Image Source: https://www.tutorialspoint.com/python_pandas/python_pandas_dataframe.htm

The Series Object

A **Series** is like a list

A **Series** is like a list or a dictionary

A **Series** is like a list or a dictionary **but BETTER!!**

The **Series** is the **primary building block** for all the other data structures we will discuss

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
myData  = pd.Series( <data>, index=<index> )
```

Filtering Data in a Series

```
randomNumbers [ <boolean condition> ]
randomNumbers [ randomNumbers < 10 ]
```

```
12    5
21    2
24    1
27    1
29    1
dtype: int64
```

```
record.str.contains('Cha')
firstname     True
lastname     False
middle       False
dtype: bool

record[ record.str.contains('Cha') ]
firstname    Charles
dtype: object
```

String Functions

Function	Explanation
Series.str.contains(<pattern>)	Returns true/false if text matches a pattern
Series.str.count(<pattern>)	Returns number of occurrences of a pattern in a string
Series.str.extract(<pattern>)	Returns matching groups from a string
Series.str.find(<string>)	Returns index of first occurrences of a substring (Note: not regex)
Series.str.findall(<pattern>)	Returns all occurrences of a regex
Series.str.len()	Returns the length of text
Series.str.replace(<pat>, <replace>)	Replaces matches with a replacement string

Date/Time Operations

```
pd.to_datetime(<times>, format='<format string>')
```

Selected Date/Time Operations

Function	Explanation
`series.dt.year`	Returns the year of the date time
`series.dt.month`	Returns the month of the date time
`series.dt.day`	Returns the days of the date time
`series.dt.weekday`	Returns the day of the week
`series.dt.is_month_start`	Returns true if it is the first day of the month
`series.dt.strftime()`	Returns an array of formatted strings specified by a **`date_format`**

https://pandas.pydata.org/pandas-docs/stable/api.html#datetimelike-properties

```
def addTwo( n ):
    return n + 2

odds.apply( addTwo )
```

```
odds.apply( lambda x: x + 1 )
```

Explore Your Data

Tukey 5 Number Summary

- **Minimum:** The smallest value in the dataset
- **Lower Quartile:** Smallest 25% of the dataset
- **The Median:** The middle value of the dataset
- **Upper Quartile:** The largest 25% of the dataset
- **Maximum:** The largest value in the dataset

`Series.abs()`	Absolute Value of the Series
`Series.count()`	Returns number of non-empty values in the series
`Series.max()`	Returns maximum value in the Series
`Series.mean()`	Returns the mean of a Series
`Series.median()`	Returns the median of a Series
`Series.min()`	Returns the minimum value in a Series
`Series.mode()`	Take a guess..
`Series.quantile([q])`	Returns the quantiles of a Series
`Series.sum`	Returns the sum of a series
`Series.std`	Returns the standard deviation of a Series

```
Series.describe()
```

Series.describe()

```
>>> random_numbers.describe()
count    50.000000
mean     50.620000
std      30.102471
min       1.000000
25%      25.500000
50%      54.000000
75%      73.000000
max      99.000000
dtype: float64
```

```
names = pd.Series(
 ['Jim','Bob',
 'Bob','Steve','Jim,'Jane','Steph','Emma','Rachel
 '])

names.describe()
```

```
names = pd.Series(
 ['Jim','Bob',
 'Bob','Steve','Jim,'Jane','Steph','Emma','Rachel
 '])

names.describe()
```

```
count        9
unique       7
top        Jim
freq         2
dtype: object
```

Finding Unique Values

```
unique_nums = []
for key, value in random_numbers.iteritems():
    if value not in unique_nums:
        unique_nums.append(value)
```

NO!!!

```
series.drop_duplicates()
series.unique()
```

Counting Unique Occurrences

```
series.value_counts()
```

```
series.value_counts(bins=5)
```

```
series.value_counts(bins=5,
normalize=True)
```

```
series.value_counts(bins=5,
normalize=True,
dropna=False)
```

```
In [7]: area_codes2.value_counts()

Out[7]: 833    7
        811    5
        822    5
        899    3
        844    3
        855    2
        dtype: int64
```

```
series.hist()
```

series.hist()

```
series.hist(bins=40)
```

```
series.hist(bins=5)
```

```
series.plot()
```

series.plot()

series.plot(kind='box')

How long will a service go undiscovered by opportunistic attackers?

By measuring the time between packets received, it is possible to estimate how long a service may be on the Internet before it is discovered by opportunistic attackers.

Questions?

Module 2: Exploratory Data Analysis

Part 2: Two Dimensional Data

STK CYBER
GET ON THE FAST TRACK

The Data Frame

```
df = pd.DataFrame( <data>, <index>, <column_names> )
```

CSV

```
df = pd.read_csv( <file> )

df = pd.read_csv( <url> )
```

Excel

```
df = pd.read_excel( <file>, sheetname=<sheetname> )
```

```
[
  {
   “changed":"2015-04-0300:00:00",
   "created":"2014-04-10 00:00:00”,
   “dnssec":"False",
   "expires":"2016-04-10 00:00:00”,
   “isMalicious":0,
   “url":"nuteczki.com"
  },
…
]
```

JSON

```
df = pd.read_json( <file> )
or
df = pd.read_json( <url> )
```

From a Database

```
df = pd.read_sql( <query>, <connection> )
```

Parquet

```
df = pd.read_parquet( <file> )
```

HTML

```
df = pd.read_html( <source> )
```

XML

```
import requests

user_agent_url = 'http://www.user-agents.org/allagents.xml'
xml_data = requests.get(user_agent_url).content
```

http://www.austintaylor.io/lxml/python/pandas/xml/dataframe/2016/07/08/convert-xml-to-pandas-dataframe/

XML

```
import xml.etree.ElementTree as ET

class XML2DataFrame:

    def __init__(self, xml_data):
        self.root = ET.XML(xml_data)

    def parse_root(self, root):
        return [self.parse_element(child) for child in iter(root)]

    def parse_element(self, element, parsed=None):
        if parsed is None:
            parsed = dict()
        for key in element.keys():
            parsed[key] = element.attrib.get(key)
        if element.text:
            parsed[element.tag] = element.text
        for child in list(element):
            self.parse_element(child, parsed)
        return parsed

    def process_data(self):
        structure_data = self.parse_root(self.root)
        return pd.DataFrame(structure_data)

xml2df = XML2DataFrame(xml_data)
xml_dataframe = xml2df.process_data()
```

http://www.austintaylor.io/lxml/python/pandas/xml/dataframe/2016/07/08/convert-xml-to-pandas-dataframe/

Log Files…

```
070823 21:00:32        1 Connect     root@localhost on test1
070823 21:00:48        1 Query       show tables
070823 21:00:56        1 Query       select * from category
070917 16:29:01       21 Query       select * from location
070917 16:29:12       21 Query       select * from location where id = 1 LIMIT 1
```

```
070823 21:00:32        1 Connect     root@localhost on test1
070823 21:00:48        1 Query       show tables
070823 21:00:56        1 Query       select * from category
070917 16:29:01       21 Query       select * from location
070917 16:29:12       21 Query       select * from location where id = 1 LIMIT 1
```

```
logdf = pd.read_table('../data/mysql.log', names=['raw'])
```

	raw
0	070823 21:00:32 1 Connect root@local...
1	070823 21:00:48 1 Query show tables
2	070823 21:00:56 1 Query select * f...
3	070917 16:29:01 21 Query select * f...
4	070917 16:29:12 21 Query select * f...

```
logdf = pd.read_table('../data/mysql.log', names=['raw'])
logdf['raw'].str.extract('(\d{6}\s\d{2}:\d{2}:\d{2})\s+(\d+)\s(\S+)
\s(.+)', expand=False)
```

	0	1	2	3
0	070823 21:00:32	1	Connect	root@localhost on test1
1	070823 21:00:48	1	Query	show tables
2	070823 21:00:56	1	Query	select * from category
3	070917 16:29:01	21	Query	select * from location
4	070917 16:29:12	21	Query	select * from location where id = 1 LIMIT 1

```
logdf = pd.read_table('../data/mysql.log', names=['raw'])
logdf['raw'].str.extract('(?P<date>\d{6}\s\d{2}:\d{2}:\d{2})\s+(?
P<PID>\d+)\s(?P<Action>\S+)\s(?P<Query>.+)', expand=False)
```

	Date	PID	Action	Query
0	070823 21:00:32	1	Connect	root@localhost on test1
1	070823 21:00:48	1	Query	show tables
2	070823 21:00:56	1	Query	select * from category
3	070917 16:29:01	21	Query	select * from location
4	070917 16:29:12	21	Query	select * from location where id = 1 LIMIT 1

Web Server Logs

Web Server Logs

```
195.154.46.135 - - [25/Oct/2015:04:11:25 +0100] "GET /linux/
doing-pxe-without-dhcp-control HTTP/1.1" 200 24323 "http://
howto.basjes.nl/" "Mozilla/5.0 (Windows NT 5.1; rv:35.0)
Gecko/20100101 Firefox/35.0"
```

Web Server Logs

```
195.154.46.135 - - [25/Oct/2015:04:11:25 +0100] "GET /linux/
doing-pxe-without-dhcp-control HTTP/1.1" 200 24323 "http://
howto.basjes.nl/" "Mozilla/5.0 (Windows NT 5.1; rv:35.0)
Gecko/20100101 Firefox/35.0"
```

```
import apache_log_parser
line_parser = apache_log_parser.make_parser("%h %l %u %t
\"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\"")
```

Web Server Logs

```
import apache_log_parser
line_parser = apache_log_parser.make_parser("%h %l %u %t
\"%r\" %>s %b \"%{Referer}i\" \”%{User-agent}i\”")

server_log = open(“../data/hackers-access.httpd”, "r")
parsed_server_data = []
for line in server_log:
    data = {}
    data = line_parser(line)
    parsed_server_data.append( data )

server_df = pd.DataFrame( parsed_server_data  )
```

Web Server Logs

er	request_first_line	request_header_referer	request_header_user_agent	request_http_ver	request_method	request_url
	GET /linux/doing-pxe-without-dhcp-control HTTP...	http://howto.basjes.nl/	Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20...	1.1	GET	/linux/doing-pxe-v
	GET /join_form HTTP/1.0	http://howto.basjes.nl/	Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20...	1.0	GET	/join_form
	POST /join_form HTTP/1.1	http://howto.basjes.nl/join_form	Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20...	1.1	POST	/join_form
	GET /join_form HTTP/1.0	http://howto.basjes.nl/	Mozilla/5.0 (Windows NT 6.3; WOW64; rv:34.0) G...	1.0	GET	/join_form
	POST /join_form HTTP/1.1	http://howto.basjes.nl/join_form	Mozilla/5.0 (Windows NT 6.3; WOW64; rv:34.0) G...	1.1	POST	/join_form
	GET /acl_users/credentials_cookie_auth/require...	http://howto.basjes.nl/join_form	Mozilla/5.0 (Windows NT 6.3; WOW64; rv:34.0) G...	1.1	GET	/acl_users/creden

Windows Event Logs

```
pip install python-evtx
```

```
import Evtx.Evtx as evtx

xml = "<Events>"
with evtx.Evtx("window_event_log.evtx") as log:
    for record in log.records():
        xml += record.xml()
xml += "</Events>"
```

Nested Data?

Nested Data?


```
{"time": 1084443427.311224,
 "timestamp": "2004-05-13T10:17:07.311224",
"IP": {
   "version": 4,
   "ttl": 128,
   "proto": 6,
   "options": [],
   "len": 48,
   "dst": "65.208.228.223",
   "frag": 0,
   "flags": 2, "src": “145.254.160.237",
   "chksum": 37355
   },
   "Ethernet": {"src": "00:00:01:00:00:00", "type": 2048, "dst": “fe:ff:20:00:01:00"},
   …
```

Nested Data

```
pd.read_json( 'nested_data.json' )
```

Nested Data

```
pd.read_json( 'nested_data.json' )
```

	DNS	Ethernet	IP	TCP	UDP	time	timestamp
0	NaN	{'type': 2048, 'dst': 'fe:ff:20:00:01:00', 'sr...	{'dst': '65.208.228.223', 'len': 48, 'version'...	{'window': 8760, 'chksum': 49932, 'sport': 337...	NaN	1.084443e+09	2004-05-13 10:17:07.311224
1	NaN	{'type': 2048, 'dst': '00:00:01:00:00:00', 'sr...	{'dst': '145.254.160.237', 'len': 48, 'version...	{'window': 5840, 'chksum': 23516, 'sport': 80,...	NaN	1.084443e+09	2004-05-13 10:17:08.222534
2	NaN	{'type': 2048, 'dst': 'fe:ff:20:00:01:00', 'sr...	{'dst': '65.208.228.223', 'len': 40, 'version'...	{'window': 9660, 'chksum': 31076, 'sport': 337...	NaN	1.084443e+09	2004-05-13 10:17:08.222534
3	NaN	{'type': 2048, 'dst': 'fe:ff:20:00:01:00', 'sr...	{'dst': '65.208.228.223', 'len': 519, 'version...	{'window': 9660, 'chksum': 43352, 'sport': 337...	NaN	1.084443e+09	2004-05-13 10:17:08.222534
4	NaN	{'type': 2048, 'dst': '00:00:01:00:00:00', 'sr...	{'dst': '145.254.160.237', 'len': 40, 'version...	{'window': 6432, 'chksum': 33825, 'sport': 80,...	NaN	1.084443e+09	2004-05-13 10:17:08.783340

Nested Data

```
from pandas.io.json import json_normalize
import json
import pandas as pd

with open('nested.json') as data_file:
    pcap_data = json.load(data_file)

df = pd.DataFrame( json_normalize(pcap_data) )
```

Nested Data

```
df = pd.DataFrame( json_normalize(pcap_data) )
```

...	TCP.seq	TCP.sport	TCP.urgptr	TCP.window	L
...	951057939.0	3372.0	0.0	8760.0	
...	290218379.0	80.0	0.0	5840.0	
...	951057940.0	3372.0	0.0	9660.0	
...	951057940.0	3372.0	0.0	9660.0	
...	290218380.0	80.0	0.0	6432.0	

ElasticSearch & Splunk

elasticsearch

splunk>

ElasticSearch & Splunk

```
pip install huntlib
```

```
e = ElasticDF(
  url="https://localhost:9200",
  ssl=True,
  username="myuser",
  password="mypass"
)
df = e.search_df(
  lucene="proto:tcp AND port:80",
  index="myindex-*",
  days=1,
  normalize=False
)
```

```
s = SplunkDF(
  host="<splunk_ip>",
  username="myuser",
  password="mypass"
)
df = s.search_df(
  spl='search index=win_events src="10.9.*.*"',
  start_time=datetime.now()-timedelta(days=2),
  end_time=datetime.now()
)
```

Manipulating a DataFrame

```
series = df['column1']
```

Returns a **series**

```
df['ip'].value_counts().head()
```

Two ways of accessing columns

```
series = df['column1']
```

```
series = df.column1
```

Don't use the dots!

```
df = df[['column1','column2','column3']]
```

Returns a DataFrame

Filtering a DataFrame

```
df[<boolean condition>]
```

```
df[['col1','col2']][df['col3'] > 5]
```

Columns

Filter

```
data.loc[<index>]
data.loc[<list of indexes>]
data.sample(<n>)
```

data.apply(<function>)

IF
Call Apply On...

THEN
Function
Receives

data.apply(<function>)

IF
Call Apply On...

THEN
Function
Receives

IF
Function
Returns

THEN
Apply
Returns...

Questions?

1	2	3
4	5	6
7	8	9

`data.T`

1	4	7
2	5	8
3	6	9

```
#Gets you the sum of columns
data.sum( axis=0 )

#Gets you the sum of the rows
data.sum( axis=1 )
```

```
DataFrame.drop(labels,
axis=0,
level=None,
inplace=False,
errors='raise')¶
```

Merging Data Sets

Union

Union

Series 1

Index	Value
0	6
1	4
2	2
3	3

Union

Series 1

Index	Value
0	6
1	4
2	2
3	3

Series 2

Index	Value
0	7
1	5
2	3
3	4

Union

combinedSeries

Index	Value
0	6
1	4
2	2
3	3
4	7
5	5
6	3
7	4

```
combinedSeries = pd.concat(
                    [series1, series2],
                    ignore_index=True )
```

Joins

Inner Join (Intersection)

Left Join

Left Join

Right Join

```
pd.merge( leftData, rightData )
```

```
pd.merge( leftData,
          rightData,
          how="<join type>" )
```

```
pd.merge( leftData, rightData,
how="<join type>" )
```

Option	SQL Equivalent	Description
inner	INNER JOIN	Intersection
left	LEFT OUTER JOIN	Returns items in Set A, but not in Set B
right	RIGHT OUTER JOIN	Returns items in Set B, but not in Set A
outer	FULL OUTER JOIN	Returns the union of both sets

```
pd.merge( leftData, rightData,
    how="<join type>",
    on=<field list> )
```

Grouping and Aggregating Data

Grouping and Aggregating Data

date	src_ip	dst_ip	port
2018-06-21	192.168.20.2	10.10.4.1	80
2018-06-21	192.168.20.1	10.10.4.2	443
2018-06-21	192.168.20.2	10.10.5.1	80
2018-06-22	192.168.20.2	10.10.4.1	80

```
df.groupby(field or list of fields)
```

Grouping and Aggregating Data

date	src_ip	dst_ip	port
2018-06-21	192.168.20.2	10.10.4.1	80
2018-06-21	192.168.20.1	10.10.4.2	443
2018-06-21	192.168.20.2	10.10.5.1	80
2018-06-22	192.168.20.2	10.10.4.1	80

```
df.groupby('src_ip')['port'].count()
```

src_ip	count
192.168.20.1	1
192.168.20.2	3

Grouping and Aggregating Data

date	src_ip	dst_ip	port
2018-06-21	192.168.20.2	10.10.4.1	80
2018-06-21	192.168.20.1	10.10.4.2	443
2018-06-21	192.168.20.2	10.10.5.1	80
2018-06-22	192.168.20.2	10.10.4.1	80

```
df.groupby(['date','src_ip'])['port'].count()
```

Multi-Index

```
date          src_ip
2018-06-21    192.168.20.1    1
              192.168.20.2    2
2018-06-22    192.168.20.2    1
```

Grouping and Aggregating Data

date	src_ip	dst_ip	port
2018-06-21	192.168.20.2	10.10.4.1	80
2018-06-21	192.168.20.1	10.10.4.2	443
2018-06-21	192.168.20.2	10.10.5.1	80
2018-06-22	192.168.20.2	10.10.4.1	80

```
df.groupby(['date','src_ip'], as_index=False)
['port'].count()
```

	date	src_ip	port
0	2018-06-21	192.168.20.1	1
1	2018-06-21	192.168.20.2	2
2	2018-06-22	192.168.20.2	1

Grouping Functions

Pivot

df

	foo	bar	baz	zoo
0	one	A	1	x
1	one	B	2	y
2	one	C	3	z
3	two	A	4	q
4	two	B	5	w
5	two	C	6	t

```
df.pivot(index='foo',
         columns='bar',
         values='baz')
```

bar	A	B	C
foo			
one	1	2	3
two	4	5	6

http://jalammar.github.io/visualizing-pandas-pivoting-and-reshaping/

```
Series.dropna()
Series.fillna(value="<something>")
```

In-Class Exercise

Please complete Worksheet 1: Exploring Your Data

Module 3: Machine Learning Part 1
Feature Engineering

Agenda

- Feature Selection & Engineering
- Math free overview of classification models
- Evaluating Model Performance
- Improving model performance

Machine Learning Process

Machine Learning Terms

- **Features:** The mathematical representation of the original data. The features are the columns in your data set. Since the features will be a matrix, the are often written as X.

- **Observations:** The rows of your feature set.

- **Target:** The variable that you are trying to predict. Often represented as y.

Do you have labeled data?
Yes
No
Categorical or Quantity?
Quantity
Category
Regression
Classification
Yes
Can you label it?
No
Clustering/ Patterns
Prediction, Inference
Prediction
Nothing proven
Recommenders, Similar, anomalous

Features

http://www.google.com

Features

http://www.google.com

domain_length	vowel_count	digit_count
6	3	0

Representation of URL Knowledge

- Come up with a **representation/set of knowledge** that has **enough complexity** to accurately describe the problem for the computer
- Knowledge here does not mean hard-coded knowledge or formal set of rules
- **The computer rather uses the knowledge we provide to extract patterns and acquire own knowledge**
- We should provide knowledge about reality that has **high variance about the problem** it describes (e.g. a feature that is high when it rains and low when it's sunshine)

URL Definition

https://www.google.com/search?q=URL&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjcl6ut-IDUAhVEPCYKHdJGDsYQ_AUIDCgD&biw=1215&bih=652

https://	protocol
www	subdomain
google.com	zone apex
google	domain
.com	top-level-domain (tld)
/search?q=URL…	path

DNS 101

- **Domain Name Service** (DNS) resolves domain names to IP addresses (like a phone book)
- **Domain Registrars**: authority that signs unique domain names (GoDaddy, BlueGtaor)
- **State of Authority** (SOA): Contains for example name of server for zone, administrator of zone, default time-to-live (ttl = time a DNS record is cached), seconds of secondary name server should wait before checking for updates
- **Root Zone** controlled by Internet Assigned Numbers Authority (IANA)
- **Name Servers** (NS Records): used by tld servers to direct traffic to DNS server (which contains authoritative DNS records)
- **A records** (part of DNS record): “A” stands for IP Address
- **CNAME** (part of DNS record): resolves one domain name to another
- **Autonomous System** (AS) and Border gateway Protocol (BGP) info

Python libraries: `python-whois, dnspython, tldextract, ipaddress`

DNS Flow

Root NS
2
3
1
8
4
5
DNS
End User OS
DNS Resolver
TLD NS
9
6
7
Google Web server
Google NS

What makes them different?

URL BlackList

amazon-sicherheit.kunden-ueberpruefung.xyz

eclipsehotels.com/language/en-GB/eng.exe

bohicacapital.com/page

summerweb.net

ad.getfond.info

vdula.czystykod.pl/rxdjna2.html

svision-online.de/mgfi/administrator/components/com_babackup/classes/fx29id1.txt

URL WhiteList

gurufocus.com/stock/PNC

dvdtalk.ru/review

333cn.com/zx/zhxw.html

made-in-china.com/special/led-lighting

google.com/u/0/112261544981697332354/posts

youtube.com/watch?v=Qp8MQ4shN6U

unesco.org/themes/education-sustainable-developm

thisisthefirst.com/page/5

Malicious URL Detection Features (Literature)

1. **BlackList Features**: BlackLists suffer from a high false negative rate, but can still be useful as machine learning feature.
2. **Lexical Features**: Capture the property that malicious URLs tend to "look different" from benign URLs. **Contextual information** such as the length of the URL, number of digits, lengths of different parts, entropy of domain name.
3. **Host-based Features**: Properties of web site host. **"Where"** the site is hosted, **"who" owns it** and **"how" it is managed**. API queries are needed (WHOIS, DNS records). Examples: Date of registration, the geolocations, autonomous system (AS) number, connection speed or time-to-live (TTL).
4. **Content-based Features**: Less commonly used feature as it requires **execution of web-page**. Can be not only be not safe, but also increases the computational cost. Examples: HTML or JavaScript based.

Sahoo et al. 2017: [Malicious URL Detection using Machine Learning: A Survey]

Preparation In Class Exercise
ML Feature Engineering

Lexical Features

1. Length of URL
2. Length of domain
3. Count of digits
4. Entropy of domain
5. Position (or index) of the first digit
6. **Bag-of-words** for tld, domain and path parts of the URL

Host-based Features

1. Time delta between today's date and creation date
2. Check if it is an IP address

Data Set (Features and Target)

	url	isMalicious	domain	created
56675	jeita.biz/w/google/drive/document.html?ssl=yes	1	jeita.biz	2012-04-11 17:08:19
73229	sosnovskoe.info/layouts/plugins/mailbox	1	sosnovskoe.info	2011-09-19 09:53:07
60112	teothemes.com/html/mp3pl/blue-preview.jpg	1	teothemes.com	2011-09-08 21:43:00
66946	kfj.cc:162/17852q	1	kfj.cc	2013-08-18 05:52:47
81906	verapdpf.info/db/6d1b281b5c4bbcfe3b99228680c232fa	1	verapdpf.info	2016-08-18 07:09:03

Features or X

Target or y

	isMalicious	isIP	Length	LengthDomain	DigitsCount	EntropyDomain	FirstDigitIndex	com	org	net	...	w	waset
73320	1	0	27	21	0	3.558519	0	0	1	0	...	0	0
30785	0	0	77	11	14	3.095795	22	1	0	0	...	0	0
60789	1	0	141	11	5	3.459432	103	0	1	0	...	0	0
19495	0	0	59	13	20	3.546594	31	1	0	0	...	0	0
45022	1	0	23	11	7	3.277613	13	0	0	0	...	0	0

Preprocessing - an Art Work!

- Imputing missing values
- Scaling/Normalization
- One-Hot Encoding (Encoding categorical features)
- Embedding (e.g. word2vec)
- Binarizing (e.g. needed for Deep Learning multi-class target vector encoding)
- Encoding strings as int
- Dimensionality Reduction (e.g. PCA)
- Augmentation (e.g. tild/zoom images)
- Feature selection based on classifier
- Variance threshold

Data Types

Primary Python libraries: `pandas, sklearn, scipy`

http://scikit-learn.org/stable/modules/preprocessing.html

Imputing Missing Values

```
# using the most_frequent value
df['src_bytes'] = df['src_bytes'].fillna
(df['src_bytes'].value_counts().index[0])

# using the mean value
df['dst_bytes'] = df['dst_bytes'].fillna(df['dst_bytes'].mean())
```

One Hot Encoding

Color
Red
Red
Blue
Green
Yellow
Red

One Hot Encoding

4 Categories → 4 Columns with 1 when Category is True and delete original column!

Color
Red
Red
Blue
Green
Yellow
Red

Color_Red	Color_Blue	Color_Yellow	Color_Green
1	0	0	0
1	0	0	0
0	1	0	0
0	0	0	1
0	0	1	0
1	0	0	0

One Hot Encoding

```
colors = ['Red', 'Red', 'Blue', 'Green', 'Yellow', ‘Red']
series_data = pd.Series( colors )
pd.get_dummies( series_data )

# df scenario
df=pd.get_dummies(df, prefix=None, prefix_sep='_',
dummy_na=False, columns=['protocol_type','flag'],
sparse=False)
```

Encoding Strings as Integers

```
PROBE = ['portsweep.', 'satan.', 'nmap.', 'ipsweep.']
df = df.replace(to_replace = PROBE, value=1)
```

Feature Scaling

When you're creating a scaling object, you should first "**fit**" it to the **training data**, then **transform** both the **training and testing data** using the "fit" scaler.

If you try to fit the training and testing data separately, **you will get inaccurate results**.

Standard Scaling

$$zscore(x_i) = \frac{x_i - \mu}{\sigma}$$

```
scaled_feature = (feature - column_mean) / standard_deviation
```

Standard Scaling

$$zscore(x_i) = \frac{x_i - \mu}{\sigma}$$

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(features)
features_scaled = scaler.transform(features)

or
features_scaled = scaler.fit_transform(features)
```

Standard Scaling

```
original values:
 [[  0.9    0.1   40. ]
 [  0.3    0.2   50. ]
 [  0.6    0.8   60. ]]

Mean of each column:
 [  0.6   0.3667   50.]

SD of each column:
 [ 0.2449   0.3091   8.165 ]
```

```
scaled values:
 [[ 1.2247 -0.8627 -1.2247]
 [-1.2247 -0.5392  0.     ]
 [ 0.      1.4018  1.2247]]

Means of scaled data, per column:
 [ 0. -0.  0.]

SD's of scaled data, per column:
 [ 1.  1.  1.]
```

Notice that the mean of standard scaled data is zero and the StdDev is 1.

Min/Max Scaling

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

```
normed_feature = (feature - col_min) / (col_max - col_min)
```

Min/Max Scaling

```
from sklearn.preprocessing import MinMaxScaler
minmax = MinMaxScaler()

minmax.fit(features)
features_scaled_minmax = minmax.transform(features)

or

features_scaled_minmax = minmax.fit_transform(features)
```

Min/Max Scaling

```
original values:
 [[  0.9    0.1   40. ]
 [  0.3    0.2   50. ]
 [  0.6    0.8   60. ]]

Mean of each column:
 [  0.6   0.3667   50.]

SD of each column:
 [ 0.2449   0.3091   8.165 ]
```

```
scaled values:
 [[ 1.       0.       0.     ]
 [ 0.       0.1429   0.5    ]
 [ 0.5      1.       1.     ]]

Means of scaled data, per column:
[ 0.5    0.381   0.5   ]

SD's of scaled data, per column:
[ 0.4082   0.4416   0.4082]
```

Dealing with Imbalanced Classes

Dealing with Imbalanced Classes

- A lot of real-world security data will have very imbalanced targets.
- Fortunately, there is a library called imbalanced-learn which can assist.
- Imbalanced-learn provides a series of options to resample the data so that you have more balanced classes
- Docs available here: http://imbalanced-learn.org/

Dealing with Imbalanced Classes

```
from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler(random_state=0)
X_resampled, y_resampled = ros.fit_resample(X, y)
```

Selecting Features

Should we use all of them?

How do we know which features to use and which to discard?

Selects *k* features according to the highest score

```
best_features = SelectKBest(score_func=chi2,k=3).fit_transform(features,target)
```

Selects all features above a given threshold in the scoring function

```
best_features =
SelectPercentile(score_func=chi2,percentile=3).fit_transform(features,target)
```

Available Scoring Functions:

- **For regression:** f_regression, mutual_info_regression
- **For classification:** chi2, f_classif, mutual_info_classif

References:
http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html
http://scikit-learn.org/stable/modules/feature_selection.html#univariate-feature-selection

How do we know which features to use and which to discard?

Visualize them!!

Introducing Yellowbrick and scikit-plot

http://www.scikit-yb.org/en/latest/api/features/rankd.html

```
import pandas as pd
import seaborn as sns
from yellowbrick.features.rankd import Rank2D

# Extract the numpy arrays from the data frame
features = df[features]
target = df['diagnosis']

# Instantiate the visualizer with the Covariance ranking algorithm
visualizer = Rank2D(features=features, algorithm='covariance')

visualizer.fit(features, target)
visualizer.transform(features)
visualizer.poof()
```


http://www.scikit-yb.org/en/latest/api/features/rankd.html

```
import seaborn as sns
sns.pairplot(<features>, hue='<target>' )
```

```
from yellowbrick.features.radviz import RadViz
…
visualizer = RadViz(classes=<target classes>, features = <features>)
visualizer.fit(features, target)
visualizer.transform(features)
visualizer.poof()
```

WHAT IF I TOLD YOU...
FEATURE ENGINEERING
CAN BE AUTOMATED
imgflip.com

Feature Tools

- Open Source
- Automated Feature Engineering

https://www.featuretools.com/

Feature Tools

1. Features are derived from relationships between the data points in a dataset.

To calculate a customer's most expensive purchase, we apply the **Max** primitive to the purchase amount field in all related purchases. When we perform the same steps to a dataset of airplane flights, we calculate "the longest flight delay".

https://www.featuretools.com/

Feature Tools

2. Across datasets, many features are derived by using similar mathematical operations.

To calculate a customer's most expensive purchase, we apply the **Max** primitive to the purchase amount field in all related purchases. When we perform the same steps to a dataset of airplane flights, we calculate "the longest flight delay".

https://www.featuretools.com/

Feature Tools

3. New features are often composed from utilizing previously derived features.

https://www.featuretools.com/

In-Class Exercise

Please complete Worksheet 2: Feature Engineering

Module 4: Machine Learning

Part 2: Modeling

Features and Labels

FUN Features and Labels

beer and cyber !!!

Our CTF!!! www.beerandcyber.com

Making a new decision

Making a new decision

unclear?

Decision surface concept

Support Vector Machines (SVM)

How can we **linearly** separate beer and cyber?

How can we **linearly** separate beer and cyber?

Answer: straight line

Same question, choose the best line!

Same question, choose the best line!

Answer: 2

How about now?

How about now?

Answer: 2

Open BlackBox…

Another test, how do we best separate beer and cyber linearly?

Choose the best line!

Choose the best line!

Answer: 2

Wait and now?

Just ignore?

Open BlackBox…

Simple Decision Tree (DT)

Should I buy a new tech gadget?

This is Charles' example!

How can we separate beer and cyber?

gtkcyber.com

How can we separate beer and cyber?

Pen and paper worksheet: solve!

Please take 10 minutes and complete **printed Worksheet - create a Decision Tree by hand!**

Pen and paper worksheet: solve!

Answer:

1. secrecy threshold 5.27
2. alcohol threshold 4.47
3. secrecy threshold 2.70

Ultimate last challenge! Who can solve it?

Such boundaries can be a sign of over-fitting!

Lastly: How does a Decision Tree exactly decide to split?

Criterion for best splits: **Impurity**

A

B

Quiz: By intuition which example appears to be more pure, that is, separates the two classes better?

Lastly: How does a Decision Tree exactly decide to split?

Criterion for best splits: **Information Gain**

A

B

Answer: A

Which feature below gives the best split?

Information Gain = entropy(parent) - [weighted average]entropy(children)

entropy(parent)=1.0
most impure binary system

choosing next best feature to split next

?

Which feature below gives the best split?

Information Gain = entropy(parent) - [weighted average]entropy(children)

entropy(parent)=1.0
most impure binary system

choosing next best feature to split next

	feature 1	feature 2	feature 3
Information Gain	**0.3112**	**0**	**1**

Advantages of Decision Trees

- Can be used for regression or classification
- Can be displayed graphically
- Highly interpretable
- Can be specified as a series of rules, and more closely approximate human decision-making than other models
- Prediction is fast
- Features don't need scaling
- Automatically learns feature interactions (they are non-linear models)
- Tend to ignore irrelevant features (especially when there are lots of features)
- Because decision trees are non-linear models they will outperform linear models if the relationship between features and response is highly non-linear

Disadvantages of Decision Trees

- Performance is (generally) not competitive with the best supervised learning methods
- Can easily overfit the training data (tuning is required)
- Small variations in the data can result in a completely different tree (they are high variance models)
- Recursive binary splitting makes "locally optimal" decisions that may not result in a globally optimal tree
- Don't tend to work well if the classes are highly unbalanced
- Don't tend to work well with very small datasets

K-Nearest Neighbors Classifier

K-nearest neighbors

Define K, and for every unknown observation:

1. Find k-nearest neighbors (by *distance*)
2. Assign class based on dominate class
3. Optional weight by distance

K-nearest neighbors

K-nearest neighbors

k = 1
1 Normal
0 Infected
Assign “Normal”

K-nearest neighbors

k = 5
4 Normal
1 Infected
Assign “Normal”

K-nearest neighbors

Further reading: https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

K-NN Classifiers

Advantages of K-NN Classifiers

- Robust to noisy, non-linear training data
- Flexible to feature / distance choices
- Works with large datasets
- Naturally handles multi-class cases

Disadvantages of K-NN Classifiers:

- Need to determine the value of K
- Training is fast but predictions are slow. Indexing can help
- Must have meaningful distance function

Ensembles

Ensembles

Ensemble learning is the process of combining several predictive models in order to produce a combined model that is more accurate than any individual model.

- **Regression:** take the average of the predictions
- **Classification:** take a vote and use the most common prediction, or take the average of the predicted probabilities

Ensembles

For ensembling to work well, the models must have the following characteristics:

- **Accurate:** they outperform random guessing
- **Independent:** their predictions are generated using different processes

The big idea: If you have a collection of individually imperfect (and independent) models, the "one-off" mistakes made by each model are probably not going to be made by the rest of the models, and thus the mistakes will be discarded when averaging the models.

Note: As you add more models to the voting process, the probability of error decreases, which is known as Condorcet's Jury Theorem.

Bagging

1. Grow *n* trees using *n* bootstrap samples from the training data.
2. Train each tree on its bootstrap sample and make predictions.
3. Combine the predictions:
   - Average the predictions for **regression trees**
   - Take a majority vote for **classification trees**

Random Forest

Random Forests are a **slight variation of bagged trees** that have even better performance:

- Just like bagging, we create an ensemble of decision trees using bootstrapped samples of the training set.

- However, when building each tree, each time a split is considered, a **random sample of *m* features** is chosen as split candidates from the **full set of *p* features**. The split is only allowed to use **one of those *m* features**.

  - A new random sample of features is chosen for **every single tree at every single split**.

  - For **classification**, m is typically chosen to be the square root of p (the total number of features).

  - For **regression**, m is typically chosen to be somewhere between $p/3$ and p.

Tuning a Random Forest

2 important parameters that should be tuned when creating a random forest model are:

- The number of trees to grow (called **n_estimators** in scikit-learn)
- The number of features that should be considered at each split (called **max_features** in scikit-learn)

Tuning a Random Forest

Tuning a Random Forest

Tuning a Random Forest

Class A: p% Class B: 1-p%

Majority Win

Random Forests

Advantages of Random Forests:

- Performance is competitive with the best supervised learning methods
- Provides a more reliable estimate of feature importance
- Allows you to estimate out-of-sample error without using train/test split or cross-validation

Disadvantages of Random Forests:

- Less interpretable
- Slower to train
- Slower to predict

Supervised Machine Learning

Implementation

GTK CYBER
GET ON THE FAST TRACK

Machine Learning Process

Scikit-Learn Process

The basic process for any supervised learning is the same for any supervised learning in Python.

1. Create the Classifier or Regressor object.

```
clf = RandomForestClassifier()
```

2. Call the **`.fit()`** method using the **training feature matrix (X)** and the **training target vector (y)**.

```
clf.fit( training_features, training_target )
```

3. Call the **`.predict()`** method using a feature matrix

#Returns vector of predictions

```
clf.predict( testing_features )
```

Train-Predict in sklearn

```
## Support Vector Machine Classification

clf = svm.SVC()
clf = clf.fit(X, target)
target_pred = clf.predict(X)
```

clf means classifier

Train-Predict in sklearn

```
## Decision Tree Classification

clf = tree.DecisionTreeClassifier()
clf = clf.fit(X, target)
target_pred = clf.predict(X)
```

clf means classifier

Classification Algorithms in Scikit-Learn

- Generalized Linear Models
- Kernel Ridge
- Support Vector Machines
- Nearest Neighbors
- Gaussian Processes
- Naive Bayes
- Trees
- Neural Networks
- AdaBoost
- Gradient Tree Boosting
- Ensemble methods

Classification Algorithms in Scikit-Learn

http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

Python Sklearn Overview

http://scikit-learn.org/stable/

scikit learn

Home | Installation | Documentation | Examples

powered by Google

Google Custom Search | Search

scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.
Algorithms: SVM, nearest neighbors, random forest, ... — Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, ... — Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, ... — Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization. — Examples

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: grid search, cross validation, metrics. — Examples

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: preprocessing, feature extraction. — Examples

Sklearn Cheat-Sheet

gtkcyber.com

Sklearn Classifiers Navigation

sklearn.tree.DecisionTreeClassifier

class sklearn.tree. **DecisionTreeClassifier** (*criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_split=1e-07, class_weight=None, presort=False*) [source]

A decision tree classifier.

Read more in the User Guide.

Parameters: **criterion** : string, optional (default="gini")

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

splitter : string, optional (default="best")

The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

max_features : int, float, string or None, optional (default=None)

The number of features to consider when looking for the best split:

Check parameters for each classifier!

http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

Sklearn Classifiers Navigation

fit (*X, y, sample_weight=None, check_input=True, X_idx_sorted=None*) [source]

Build a decision tree classifier from the training set (X, y).

Parameters: **X** : array-like or sparse matrix, shape = [n_samples, n_features]

The training input samples. Internally, it will be converted to `dtype=np.float32` and if a sparse matrix is provided to a sparse `csc_matrix`.

y : array-like, shape = [n_samples] or [n_samples, n_outputs]

The target values (class labels) as integers or strings.

sample_weight : array-like, shape = [n_samples] or None

Sample weights. If None, then samples are equally weighted. Splits that would create child nodes with net zero or negative weight are ignored while searching for a split in each node. Splits are also ignored if they would result in any single class carrying a negative weight in either child node.

predict (*X, check_input=True*) [source]

Predict class or regression value for X.

For a classification model, the predicted class for each sample in X is returned. For a regression model, the predicted value based on X is returned.

Parameters: **X** : array-like or sparse matrix of shape = [n_samples, n_features]

The input samples. Internally, it will be converted to `dtype=np.float32` and if a sparse matrix is provided to a sparse `csr_matrix`.

check_input : boolean, (default=True)

Allow to bypass several input checking. Don't use this parameter unless you know what you do.

Returns: **y** : array of shape = [n_samples] or [n_samples, n_outputs]

The predicted classes, or the predict values.

Check input and output dimensions of fit and predict methods for feature matrix X and target vector y!

http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

Sklearn Classifiers Navigation

Attributes:

classes_ : array of shape = [n_classes] or a list of such arrays

The classes labels (single output problem), or a list of arrays of class labels (multi-output problem).

feature_importances_ : array of shape = [n_features]

The feature importances. The higher, the more important the feature. The importance of a feature is computed as the (normalized) total reduction of the criterion brought by that feature. It is also known as the Gini importance [R245].

max_features_ : int,

The inferred value of max_features.

n_classes_ : int or list

The number of classes (for single output problems), or a list containing the number of classes for each output (for multi-output problems).

n_features_ : int

The number of features when `fit` is performed.

After calling fit method you can retrieve certain attributes of your clf object!

http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

Sklearn Classifiers Navigation

fit (*X, y=None*) [source]

Compute k-means clustering.

Parameters:	**X** : array-like or sparse matrix, shape=(n_samples, n_features) Training instances to cluster.

fit_predict (*X, y=None*) [source]

Compute cluster centers and predict cluster index for each sample.

Convenience method; equivalent to calling fit(X) followed by predict(X).

fit_transform (*X, y=None*) [source]

Compute clustering and transform X to cluster-distance space.

Equivalent to fit(X).transform(X), but more efficiently implemented.

transform (*X, y=None*) [source]

Transform X to a cluster-distance space.

In the new space, each dimension is the distance to the cluster centers. Note that even if X is sparse, the array returned by transform will typically be dense.

Parameters:	**X** : {array-like, sparse matrix}, shape = [n_samples, n_features] New data to transform.
Returns:	**X_new** : array, shape [n_samples, k] X transformed in the new space.

Clustering and dimensionality reduction methods like PCA have more fit and transform rather than predict and often combine fit_transform in one method!

http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

"Why should I trust you?"
Explaining the predictions of any classifier

```
import lime
import lime.lime_tabular
explainer = lime.lime_tabular.LimeTabularExplainer(<training_data>,
        feature_names=<feature names>,
        class_names=<class names>,
        discretize_continuous=False)

exp = explainer.explain_instance( <test_row>,
   <model>.predict_proba,
   num_features=5,
   top_labels=1)

exp.show_in_notebook(show_table=True, show_all=False)
```

Feature	Value
length	6.00
digits	0.00
entropy	2.58
vowel-cons	0.50
ngrams	1606.84

https://github.com/marcotcr/lime

gtkcyber.co

Supervised Machine Learning

Metrics and cross-validation

GTK CYBER
GET ON THE FAST TRACK

Performance Metrics for Classifiers

Type I Error
(False Positive)

Type II Error
(False Negative)

Confusion Matrix

```
target =      [1,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0]
target_pred = [0,1,1,0,0,1,1,1,1,0,0,0,0,0,0,0]
print(confusion_matrix(target, target_pred))
```

True target

0 [[7 2]
1 [3 4]]

0 1

Predicted target

True positive	**False Negative** (Type II error)
False Positive (Type I error)	**True negative**

Accuracy

```
accuracy = accuracy_score(target, target_pred)
```

True positive	**False Negative** (Type II error)
False Positive (Type I error)	**True negative**

True target

0 [[7 2]

1 [3 4]]

0 1

Predicted target

Accuracy = (TP + TN)/ N

Quiz: Calculate by hand!

Accuracy

```
accuracy = accuracy_score(target, target_pred)
```

True positive	False Negative (Type II error)
False Positive (Type I error)	True negative

True target

0 [[7 2]

1 [3 4]]

0 1

Predicted target

Accuracy = (TP + TN)/ N

= (7+4)/16

= 0.6875

Precision

Column-wise!

Precision

```
precision_class = precision_score(target, target_pred, average = None)
precision_avg = precision_score(target, target_pred, average = 'binary')
```

True positive	**False Negative** (Type II error)
False Positive (Type I error)	**True negative**

True target

0 [[7 2]

1 [3 4]]

0 1

Predicted target

Precision = TP/ (TP + FP)

Quiz: Calculate precision by hand for both classes!

Precision

```
precision_class = precision_score(target, target_pred, average = None)
precision_avg = precision_score(target, target_pred, average = 'binary')
```

True positive	False Negative (Type II error)
False Positive (Type I error)	True negative

True target

0 [[7 2]

1 [3 4]]

0 1

Predicted target

Precision = TP/ (TP + FP)

Prec_Class0 = 7/(7+3) = 0.7

Prec_Class1 = 4/(4+2) = 0.666

Precision

Accurate
Precise
Not Accurate
Precise
Accurate
Not Precise
Not Accurate
Not Precise

Recall

Row-wise!

Recall

```
recall_class = recall_score(target, target_pred, average = None)
recall_avg = recall_score(target, target_pred, average = 'binary')
```

True positive	**False Negative** (Type II error)
False Positive (Type I error)	**True negative**

True target

0 [[7 2]

1 [3 4]]

0 1

Predicted target

Recall = TP/ (TP + FN)

Quiz: Calculate recall by hand for both classes!

Recall

```
recall_class = recall_score(target, target_pred, average = None)
recall_avg = recall_score(target, target_pred, average = 'binary')
```

True positive	**False Negative** (Type II error)
False Positive (Type I error)	**True negative**

True target

0 [[7 2]

1 [3 4]]

0 1

Predicted target

Recall = TP/ (TP + FN)

Rec_Class0 = 7/(7+2) = 0.777

Rec_Class1 = 4/(4+3) = 0.571

Receiver Operating Characteristic (ROC)

- Compares the True Positive Rate (TPR) on Y-axis, to the False Positive Rate (FPR) on the X- axis

```
import matplotlib.pyplot as plt
import scikitplot as skplt

skplt.metrics.plot_roc(target_test,
predicted_probs)

plt.show()
```


https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

Area Under Curve (AUC)

- AUC is an aggregate measure of performance across all possible classification thresholds.

- AUC ranges from 0 to 1 and a model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0.

```
from sklearn.metrics import
roc_auc_score

roc_auc_score(target_test,
target_preds)
```

Quiz: compute metrics for each class of 3-class confusion matrix

```
target =      [1,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,2,2,2,2,1,1,2,0,0,0,0,0]
target_pred = [0,1,1,0,0,1,1,1,1,0,0,0,0,0,0,0,2,2,2,2,2,2,1,2,2,2,2,2]
print(confusion_matrix(target, target_pred))
```

True positive	**False Negative** (Type II error)
False Positive (Type I error)	**True negative**

True target

```
[[7 2 5]
 [3 4 2]
 [0 1 4]]
```

Predicted target

Accuracy = (Sum Diagonal)/ N

Precision = TP/ (TP + FP)

Recall = TP/ (TP + FN)

Quiz: compute metrics for each class of 3-class confusion matrix

```
target =      [1,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,2,2,2,2,1,1,2,0,0,0,0,0]
target_pred = [0,1,1,0,0,1,1,1,1,0,0,0,0,0,0,0,2,2,2,2,2,2,1,2,2,2,2,2]
print(confusion_matrix(target, target_pred))
```

True positive	**False Negative** (Type II error)
False Positive (Type I error)	**True negative**

True target

```
[[7 2 5]
 [3 4 2]
 [0 1 4]]
```

Predicted target

Accuracy = (7+4+4)/28 = 0.5357

Prec_Class2 = 4/(4 + 5 + 2) = 0.3636

Rec_Class2 = 4/(4 + 1 + 0) = 0.8

…

```
Precision:  [ 0.7  0.57142857  0.36363636]
Recall:  [ 0.5  0.44444444  0.8]
```

Where is the biggest crime scene?

Confusion matrices all with equal accuracy 0.6875!!!
How about precision and recall?

```
[[10  0]
 [ 5  1]]
```

A

```
[[7  2]
 [3  4]]
```

B

```
[[0  4]
 [1  11]]
```

C

Where is the biggest crime scene?

Confusion matrices all with equal accuracy 0.6875!!!
How about precision and recall?

```
Precision:  [ 0.7  1]
Recall:  [ 1  0.2]
```

```
[[10  0]
 [5  1]]
```

A

```
Precision:  [ 0.7  0.7]
Recall:  [ 0.8  0.6]
```

```
[[7  2]
 [3  4]]
```

B

```
Precision:  [ 0  0.7]
Recall:  [ 0  0.9]
```

```
[[0  4]
 [1  11]]
```

Visualizing the Confusion Matrix

```
from yellowbrick.classifier import ConfusionMatrix

#Create the models for both the Random Forest
random_forest_model = RandomForestClassifier()

random_forest_conf_matrix = ConfusionMatrix(
                                random_forest_model )
random_forest_conf_matrix.fit( X_train, y_train )
random_forest_conf_matrix.score( X_test, y_test )
random_forest_conf_matrix.poof()
```

Visualizing the Classification Report

Visualizing the Classification Report

Visualizing the Classification Report

```
from yellowbrick.classifier import ClassificationReport

random_forest_class_report = ClassificationReport( random_forest_model,
                                                   classes=['Benign', 'Malignant'])
random_forest_class_report.fit(X_train, y_train)
random_forest_class_report.score(X_test, y_test)
random_forest_class_report.poof()
```

Cross-Validation

Split data for train and test

features(X)

70%

target (y)

Simple train test split!

```
# Simple Cross-Validation: Split the data set into training and test data

X_train, X_test, target_train, target_test =
model_selection.train_test_split(X, target, test_size=0.25)
```

Simple cross-validation!

```
## Train the classifier
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_train, target_train)

## Making predictions using test data
target_pred = clf.predict(X_test)

## Report metrics
accuracy = metrics.accuracy_score(target_test, target_pred)
```

Cross Validation

Better approach: Create a bunch of train/test splits, calculate the testing accuracy for each, and average the results together.

Cross Validation

1. Split the dataset into K **equal** partitions (or "folds").

2. Use fold 1 as the **testing set** and the union of the other folds as the **training set**.

3. Calculate testing accuracy.

4. Repeat steps 2 and 3 *K* times, using a **different fold** as the testing set each time.

5. Use the **average testing accuracy** as the estimate of out-of-sample accuracy.

K-Fold Cross Validation

K-fold Cross-Validation

```
scores = cross_val_score(clf, features, target, cv=<n>)
scores.mean()
```

Visualizing Goodness of Fit

https://jmetzen.github.io/2015-01-29/ml_advice.html

Visualizing Goodness of Fit

```
from yellowbrick.classifier.learning_curve import LearningCurveVisualizer


viz = LearningCurveVisualizer(GaussianNB())
viz.fit(X,y)
viz.poof()
```

https://jmetzen.github.io/2015-01-29/ml_advice.html
https://github.com/DistrictDataLabs/yellowbrick/blob/master/examples/jkeung/learning_curve.ipynb

Visualizing Goodness of Fit

```
from yellowbrick.classifier.learning_curve import LearningCurveVisualizer
viz = LearningCurveVisualizer(GaussianNB())
viz.fit(X,y)
viz.poof()
```

https://jmetzen.github.io/2015-01-29/ml_advice.html

In-Class Exercise

Please complete Worksheet 3: Implementing a Model

Hyper-Parameter Tuning!

General Optimization: e.g. Gradient Descent!

Grid Search

```
RandomForestClassifier(bootstrap=True,
class_weight=None,
criterion='gini',
max_depth=None,
max_features='auto',
max_leaf_nodes=None,
min_impurity_decrease=0.0,
min_impurity_split=None,
min_samples_leaf=1,
min_samples_split=2,
min_weight_fraction_leaf=0.0,
n_estimators=10,
n_jobs=1,
oob_score=False
)
```

Tuning these parameters

- GridSearchCV: You provide a list of possible parameters
- RandomizedSearchCV: Random combinations are searched

Grid Search

```
param_grid = [
  {'C': [1, 10, 100, 1000], 'kernel': ['linear']},
  {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001],
'kernel': ['rbf']},
 ]
```

Grid Search

```
# use a full grid over all parameters
param_grid = {"max_depth": [3, None],
              "max_features": [1, 3, 10],
              "min_samples_split": [2, 3, 10],
              "min_samples_leaf": [1, 3, 10],
              "bootstrap": [True, False],
              "criterion": ["gini", "entropy"]}

# run grid search
grid_search = GridSearchCV(clf, param_grid=param_grid)
start = time()
grid_search.fit(X, y)

print("GridSearchCV took %.2f seconds for %d candidate parameter settings."
      % (time() - start, len(grid_search.cv_results_['params'])))
report(grid_search.cv_results_)
```

http://scikit-learn.org/stable/auto_examples/model_selection/plot_randomized_search.html

Random Search

```
# specify parameters and distributions to sample from
param_dist = {"max_depth": [3, None],
              "max_features": sp_randint(1, 11),
              "min_samples_split": sp_randint(2, 11),
              "min_samples_leaf": sp_randint(1, 11),
              "bootstrap": [True, False],
              "criterion": ["gini", "entropy"]}

# run randomized search
n_iter_search = 20
random_search = RandomizedSearchCV(clf, param_distributions=param_dist,
                                   n_iter=n_iter_search)

start = time()
random_search.fit(X, y)
print("RandomizedSearchCV took %.2f seconds for %d candidates"
      " parameter settings." % ((time() - start), n_iter_search))
report(random_search.cv_results_)
```

http://scikit-learn.org/stable/auto_examples/model_selection/plot_randomized_search.html

Bayesian Search

```
pip install scikit-optimize
```

```
# run bayesian-optimized hyperparameter search
n_iter_search = 20
bayes_search = BayesSearchCV(clf, param_distributions=param_dist,
                                  n_iter=n_iter_search)

start = time()
bayes_search.fit(X, y)
print("BayesSearchCV took %.2f seconds for %d candidates"
      " parameter settings." % ((time() - start), n_iter_search))
report(bayes_search.cv_results_)
```

https://scikit-optimize.github.io/notebooks/sklearn-gridsearchcv-replacement.html

Auto ML

What else can you automate?

Auto ML

What else can you automate? It turns out, **almost everything** with TPOT.

TPOT will automate the most tedious part of machine learning by intelligently exploring thousands of possible pipelines to find the best one for your data.

Auto ML

https://github.com/EpistasisLab/tpot

Auto ML

https://github.com/EpistasisLab/tpot

Auto ML

```
from tpot import TPOTClassifier

tpot = TPOTClassifier(generations=5, population_size=20, verbosity=2)
tpot.fit(features_train, target_train)
print(tpot.score(features_test, target_test))
tpot.export('tpot_pipeline.py')
```


https://github.com/EpistasisLab/tpot

Auto ML

Docs available here: http://epistasislab.github.io/tpot/

Questions?

Module 5: Hunting with Data Science

Increasing the Signal-to-Noise Ratio

GTK CYBER
GET ON THE FAST TRACK

What is it?

Semantics

Threat Hunting

Cyber threat hunting is "the process of ***proactively and iteratively*** searching through networks to detect and isolate **advanced threats** that ***evade*** existing security solutions."

Data Science

Data science is an interdisciplinary field about scientific methods, processes, and systems to ***extract knowledge or insights from data*** in various forms, either structured or unstructured, similar to data mining.

Threat Hunting

+

Data Science

Hunting with Data Science

Data Science Hunting Funnel

@HuntOperator

gtkcyber.com

Cyber Kill Chain

Cyber Kill Chain

gtkcyber.com

Beaconing

Beaconing

- Post-Infection
- Early network-related indication of infection
- Used by malware to “phone home” to command and control server

Detection Challenges

- **Hardset Intervals**
  - Varying window sizes
- **Legit Services**
  - Windows update
  - Virus definition updates

Beaconing Detection

Beaconing: Detection

- Free Open Source Software
- Designed for data scientists, security researchers
- Written in Python
- Used for rapid prototyping and development of behavioral analytics
- Intended to make identifying malicious behavior in networks as simple as possible.

https://github.com/austin-taylor/flare

Beaconing: Detection

```
[beacon]
es_host=localhost            # IP address of ES Host, which we forwarded to localhost
es_index=logstash-flow-*     # ES index
es_port=9200                 # Logstash port (we forwarded earlier)
es_timeout=480               # Timeout limit for elasticsearch retrieval
min_occur=50                 # Minimum of 50 network occurrences to appear in traffic
min_interval=30              # Minimum interval of 30 seconds per beacon
min_percent=30               # Beacons must represent 30% of network traffic per dyad
window=3                     # Accounts for jitter... For example, if 60 second beacons
                             # occurred at 58 seconds or 62 seconds, a window of 3 would
                             # factor in that traffic.
threads=8                    # Use 8 threads to process (Should be configured)
period=24                    # Retrieve all flows for the last 24 hours.
kibana_version=5             # Your Kibana version. Currently works with 4 and 5
verbose=True                 # Display output while running script
```

https://github.com/austin-taylor/flare

Beaconing: Data Science

Time	EveBox	src_ip	src_port	proto	dest_ip	dest_port	http.http_method	http.hostname
June 13th 2017, 10:49:01.935	Correlate Flow	192.168.0.53	49,182	TCP	160.153.76.129	80	GET	www.huntoperator.com

Beaconing: Data Science

Simple: src_ip, dest_ip, dest_port -> hash

More Complex: Discrete Fourier Transform (DFT)/Fast Fourier transform (FFT)

Scenario 1

A piece of malware has infected a computer (192.168.0.53) on your network and is trying to reach back to its Command and Control (C2) server (160.153.76.129) in periodic intervals

GTK CYBER

GET ON THE FAST TRACK

HUNT!

gtkcyber.com

Beaconing: Hunt

flare_beacon -c configs/selks4.ini -csv beacons.csv

```
Austins-MacBook-Pro:flare huntoperator$ flare_beacon -c configs/selks4.ini -csv beacons.csv
[INFO] Attempting to connect to elasticsearch...
[SUCCESS] Connected to elasticsearch on localhost:9200
[INFO] Gathering flow data... this may take a while...
[INFO] Calculating destination degree.
[SUCCESS] Writing csv to beacons.csv
```

108

events to process

Beaconing: Hunt

flare_beacon -c configs/selks4.ini -csv —group —whois —focus_outbound beacons_filtered.csv

```
Austins-MacBook-Pro:flare huntoperator$ flare_beacon -c configs/selks4.ini --whois --focus_outbound -csv beacons_filtered.csv
[INFO] Attempting to connect to elasticsearch...
[SUCCESS] Connected to elasticsearch on localhost:9200
[INFO] Gathering flow data... this may take a while...
[INFO] Calculating destination degree.
[INFO] Enriching IP addresses with whois information
[INFO] Applying outbound focus - filtering multicast, reserved, and private IP space
[SUCCESS] Writing csv to beacons_filtered.csv
```

31

events to process

Beaconing: Hunt

flare_beacon -c configs/selks4.ini -csv —group —whois —focus_outbound beacons_filtered.csv

```
Austins-MacBook-Pro:flare huntoperator$ flare_beacon -c configs/selks4.ini --whois --focus_outbound -csv beacons_filtered.csv
[INFO] Attempting to connect to elasticsearch...
[SUCCESS] Connected to elasticsearch on localhost:9200
[INFO] Gathering flow data... this may take a while...
[INFO] Calculating destination degree.
[INFO] Enriching IP addresses with whois information
[INFO] Applying outbound focus - filtering multicast, reserved, and private IP space
[SUCCESS] Writing csv to beacons_filtered.csv
```

What was applied?

–group: This will group the results making it visually easier to identify anomalies.

–whois: Enriches IP addresses with WHOIS information through ASN Lookups.

–**focus_outbound**: **Filters out multicast, private and broadcast addresses from destination IPs**

Beaconing: Hunt

src_ip	dest_whois	dest_ip	dest_port	bytes_toserver	dest_degree	occurances	percent	interval
192.168.0.100	GOOGLE - Google Inc., US	8.8.8.8	53	82	16	71	98	3601
	SOFTLAYER - SoftLayer Technologies Inc., US	198.23.79.227	123	90	1	71	98	3601
192.168.0.120	AMAZON-02 - Amazon.com, Inc., US	52.192.78.22	80	1432	3	34	70	1201
	GOOGLE - Google Inc., US	8.8.4.4	53	74	7	139	63	61
		8.8.8.8	53	74	16	140	62	61
192.168.0.53	AS-26496-GO-DADDY-COM-LLC - GoDaddy.com, LLC, US	160.153.76.129	80	620	3	383	97	61
	GOOGLE - Google Inc., US	8.8.4.4	53	233	7	386	95	61
		8.8.8.8	53	65	16	386	95	61
192.168.0.60	AKAMAI-ASN1 , US	104.95.176.44	80	8399	1	3	33	97
	AMAZON-AES - Amazon.com, Inc., US	174.129.1.163	80	1691	1	77	38	101
		204.236.238.3	80	885	1	84	39	111
		23.21.110.37	80	1691	1	53	35	200
		23.21.67.210	80	1691	1	101	32	200
		23.21.98.133	80	1691	1	65	40	101
		23.23.173.186	80	1691	1	43	32	200
		23.23.236.78	80	759	1	313	41	112
		50.16.215.193	80	1691	1	25	40	101
		50.17.254.18	80	1691	1	25	40	200
	DROPBOX - Dropbox, Inc., US	162.125.32.5	443	60	1	53	35	602
	EDGECAST - MCI Communications Services, Inc. d/b/a Verizon Business, US	152.195.54.20	80	556	2	69	49	601
	LLNW - Limelight Networks, Inc., US	69.164.0.0	80	6242	2	3	33	36
	MICROSOFT-CORP-MSN-AS-BLOCK - Microsoft Corporation, US	13.107.6.151	443	3985	1	151	41	300

- **bytes_toserver**: Total sum of bytes sent from IP address to Server
- **dest_degree**: Amount of source IP addresses that communicate to the same destination
- **occurrences**: Number of network occurrences between dyads identified as beaconing.
- **percent**: Percent of traffic between dyads considered beaconing.
- **interval**: Intervals between each beacon in seconds

Beaconing: Hunt

Validate Results

gtkcyber.com

Beaconing: Hunt

Drilling in

Beaconing: Hunt

gtkcyber.com

Beaconing: Hunt

SN HTTP-UserAgentOS

SN HTTP-UserAgentDevices

SN HTTP-UserAgentOSName

SN HTTP-EventsList

Time	EveBox	src_ip	src_port	proto	dest_ip	dest_port	http.http_method	http.hostname	http.status	http.protocol	http.server
June 13th 2017, 10:49:01.935	Correlate Flow	192.168.0.53	49,182	TCP	160.153.76.129	80	GET	www.huntoperator.com	503	HTTP/1.1	Apache/2.4.25
June 13th 2017, 10:48:01.680	Correlate Flow	192.168.0.53	49,180	TCP	160.153.76.129	80	GET	www.huntoperator.com	503	HTTP/1.1	Apache/2.4.25
June 13th 2017, 10:47:01.388	Correlate Flow	192.168.0.53	49,178	TCP	160.153.76.129	80	GET	www.huntoperator.com	503	HTTP/1.1	Apache/2.4.25
June 13th 2017, 10:46:01.102	Correlate Flow	[illegible]	49,176	TCP	160.153.76.129	80	GET	www.huntoperator.com	503	HTTP/1.1	Apache/2.4.25

Beaconing: Hunt

EveBox | Inbox | Escalated | Alerts | Events | Reports

721700456887339

Refresh | Event Type: All

	Timestamp	Type	Source/Dest	Description
>	2017-06-13 10:50:03 7 hours ago	FLOW	**S:** 192.168.0.53 **D:** 160.153.76.129	TCP 192.168.0.53:49182 -> 160.153.76.129:80; Age: 1; Bytes: 6223; Packets: 19
	2017-06-13 10:50:03 7 hours ago	FLOW	**S:** 192.168.0.53 **D:** 160.153.76.129	TCP 192.168.0.53:49182 -> 160.153.76.129:80; Age: 1; Bytes: 6223; Packets: 19
	2017-06-13 10:49:02 7 hours ago	FILEINFO	**S:** 160.153.76.129 **D:** 192.168.0.53	/ - Hostname: www.huntoperator.com; Content-Type: text/html
	2017-06-13 10:49:01 7 hours ago	HTTP	**S:** 192.168.0.53 **D:** 160.153.76.129	GET - www.huntoperator.com - /
	2017-06-13 10:49:01 7 hours ago	ALERT	**S:** 192.168.0.53 **D:** 160.153.76.129	ET POLICY curl User-Agent Outbound

Beaconing: Hunt

HTTP

Hostname: www.huntoperator.com
Http User Agent: curl/7.47.0
Status: 503
User Agent.Name: Other
Http Content Type: text/html
Length: 1093
Url: /
User Agent.Os: Other
Http Method: GET
Protocol: HTTP/1.1
User Agent.Device: Other
User Agent.Os Name: Other

GeoIP

City Name: Scottsdale
Coordinates.1: 33.6119
Country Name: United States
Latitude: 33.6119
Longitude: -111.8906
Region Name: Arizona
Continent Code: NA
Country Code2: US
Dma Code: 753
Location.0: -111.8906
Postal Code: 85260
Timezone: America/Phoenix
Coordinates.0: -111.8906
Country Code3: US
Ip: 160.153.76.129
Location.1: 33.6119
Region Code: AZ

Flow - TCP 192.168.0.53 -> 160.153.76.129 [Open]

Age: 1
Bytes Toserver: 752
Pkts Toserver: 10
State: closed
Alerted: true
End: 2017-06-13T10:49:02.275746-0400
Reason: timeout
Bytes Toclient: 5471
Pkts Toclient: 9
Start: 2017-06-13T10:49:01.717867-0400

Payload [PCAP]

```
GET / HTTP/1.1
Host: www.huntoperator.com
User-Agent: curl/7.47.0
Accept: */*
```

```
47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31 0d 0a
48 6f 73 74 3a 20 77 77 77 2e 68 75 6e 74 6f 70
65 72 61 74 6f 72 2e 63 6f 6d 0d 0a 55 73 65 72
2d 41 67 65 6e 74 3a 20 63 75 72 6c 2f 37 2e 34
37 2e 30 0d 0a 41 63 63 65 70 74 3a 20 2a 2f 2a
0d 0a 0d 0a
```

Beaconing: Hunt

Report for IP 160.153.76.129

Related Reports

Refresh | All Sensors | Filter... | Apply | Clear

Alerts Over Time

DNS Hostnames Returning 160.153.76.129

#	Hostname
1806	huntoperator.com

DNS: Top Requested Hostnames

No data.

Outgoing HTTP User Agents

No data.

HTTP: Incoming HTTP Request Hostnames

#	Hostnames
903	www.huntoperator.com

Alerts: Top Alerts

#	Signature
890	ET POLICY curl User-Agent Outbound

Flow

Flows As Client	0
Flows As Server	1794
Bytes To...	1.23 MB
Bytes From...	12.70 MB
Packets To...	16k (16908)
Packets From...	19k (19084)

Incoming TLS Server Names (SNI)

No data.

TLS Versions as Client

No data.

TLS Versions as Server

No data.

HTTP: Top Requested Hostnames

No data.

CASE SOLVED

Domain Generation Algorithms (DGA)

Domain Generation Algorithms (DGA)

`vtlfccmfxlkgifuf.com`

Why DGA?

- Deterministic value
- Generate large number of domain names
  - Easy to burn
  - Cheap to register
- Used as a rendezvous point by attacker

```
In [18]:
def generate_domain(year, month, day):
    """Generates a domain name for the given date."""
    domain = ""

    for i in range(16):
        year = ((year ^ 8 * year) >> 11) ^ ((year & 0xFFFFFFF0) << 17)
        month = ((month ^ 4 * month) >> 25) ^ 16 * (month & 0xFFFFFFF8)
        day = ((day ^ (day << 13)) >> 19) ^ ((day & 0xFFFFFFFE) << 12)
        domain += chr(((year ^ month ^ day) % 25) + 97)

    return domain + '.com'
```

```
In [19]:
generate_domain(2017, 6, 23)

Out[19]: 'vtlfccmfxlkgifuf.com'
```

Source: Aditya K. Sood, Sherali Zeadally, "A [illegible] of Domain-Generation Algorithms", IEEE Security & Privacy, vol. 14, no. , pp. 46-53, July-Aug. 2016, doi:10.1109/MSP.2016.76

We want to detect this

I'M GONNA HAVE TO DATA
SCIENCE THE SH*T OUT OF THIS
imgflip.com

Scenario 2

Scenario 2

A piece of malware has infected a computer on your network and is making request to domains using DGA in an attempt to communicate to a Command and Control Server

GTK CYBER

GET ON THE FAST TRACK

HUNT!

gtkcyber.com

DNS Records

Record Count: 15408

```
In [62]:  1 dns_records = pd.read_csv('/Users/huntoperator/Downloads/exported_dns_n
```

```
In [63]:  1 dns_records
```

Out[63]:

	dns_rrname	count
0	daisy.ubuntu.com	300,129
1	srv.myskybell.com	55,053
2	googleapis.l.google.com	37,005
3	ubuntu.com	30,549
4	www.google.com	28,780
5	www.example.org	26,367
6	www.example.com	26,354
7	www.example.net	26,298
8	myskybell.com	18,296

Import Flare Tools

```
In [64]:  from flare.data_science.features import dga_classifier
```

```
In [66]:  from flare.tools.alexa import Alexa
          from flare.data_science.features import domain_tld_extract
```

```
In [65]:  dga_c = dga_classifier()

[*] Initializing... training classifier - Please wait.
[+] Classifier Ready
```

- **DGA Classifier**
  - Random Forrest Classifier
  - N-Grams
  - Uses labelled data
- **Alexa - Top 1M most popular visited websites**
  - Must pay for service now.
  - Umbrella/Majestic are free alternatives
- **Domain TLD Extract - Extracts the Top Level Domain to be checked against Alexa**
  - Also calculate degree from here

Filter Results

```
In [73]:  dns_records['domain_tld'] = dns_records.dns_rrname.apply(lambda x: domain_tld_extract(str(x)))
```

```
In [78]:  dns_records['dga_predict'] = dns_records.domain_tld.apply(lambda x: dga_c.predict(x))
```

```
In [129]:  dns_records.head()
```

Out[129]:

	dns_rrname	count	domain_tld	dga_predict
0	daisy.ubuntu.com	300,129	ubuntu.com	legit
1	srv.myskybell.com	55,053	myskybell.com	legit
2	googleapis.l.google.com	37,005	google.com	legit
3	ubuntu.com	30,549	ubuntu.com	legit
4	www.google.com	28,780	google.com	legit

```
In [84]:  dns_filter = dns_records[dns_records.dga_predict=='dga']
```

```
In [131]:  dns_filter.shape
```

```
Out[131]: (240, 4)
```

still too many results…

Filter Results

```
beacon_df.groupby(self.beacon_dest_ip)[self.beacon_dest_ip].transform('count').fillna(0).astype(int)
```

```
dns_filter1['domain_degree'] = dns_filter1.groupby('domain_tld')['domain_tld'].transform('count').fillna(0).
```

and yet…

```
In [103]: dns_filter1 = dns_filter[dns_filter['count'] < 10]
```

```
In [105]: dns_filter1.shape
Out[105]: (162, 4)
```

Still too many results…

Filter Results

```
In [117]:  alexa = Alexa()

In [123]:  dns_filter2 = dns_filter1[dns_filter1['domain_degree'] < 2]

In [119]:  dns_filter2.shape

Out[119]: (78,

In [125]:  dns_filter2['in_alexa'] = dns_filter2.domain_tld.apply(lambda x: alexa.domain_in_alexa(x))
```

down to 78

And finally…

Filter Results

apply Alexa top 1 million check…

```
In [127]:  dns_filter2[dns_filter2['in_alexa']==False]
```

Out[127]:

	dns_rrname	count	domain_tld	dga_predict	domain_degree	in_alexa
5421	version.mos.svc.ovi.com.glb.as1248.net	9	as1248.net	dga	1	False
5765	ebdr3.com	8	ebdr3.com	dga	1	False
5847	i.s-jcrew.com	8	s-jcrew.com	dga	1	False
5885	in.ml314.com	8	ml314.com	dga	1	False
6382	vtlfccmfxlkgifuf.com	8	vtlfccmfxlkgifuf.com	dga	1	False
7510	tags.wdsvc.net	6	wdsvc.net	dga	1	False
7766	get35.com	5	get35.com	dga	1	False
7920	x64dbg.com	5	x64dbg.com	dga	1	False

and…

```
In [135]:  dns_filter2[dns_filter2['in_alexa']==False].shape

Out[135]:  (57,
```

57 Results!

Pass to Analyst

- **Identify Process Generating Traffic**
  - Isolate infected host
  - Begin endpoint investigation…

Natural Language Processing

Natural Language Processing
Use Case

Challenge: Distinguishing between base64 and URIs

String 1: **Q09NRSBUTyBNWSBQWVRIT04gQ0xBU1MgSU4gVkVHQVMhISEh**
String 2: **forums/diary/Detecting+Random+Finding+Algorithmically+chosen+DNS+names+DGA/19893**

Both of these are properly formatted BASE64 encoded strings,
but only one is intentional

https://github.com/sans-blue-team/blue-team-wiki/blob/gh-pages/Tools/freq.py.md

Natural Language Processing

Solution: Mark Baggett's freq.py

Produces a **probability score** based off frequency tables in:

- Dictionary
- Your environment

Higher likelihood scores indicate common

Lower likelihood scores indicate not common

https://github.com/MarkBaggett/freq

Freq.py

How does it work?

- Identifies likelihood of character occurrence based on frequency analysis

Example: In English text, “m” is usually followed by an “o,” so seeing a “m” followed by something else would be rather unlikely to occur

Similar to high order markov chain

https://github.com/sans-blue-team/blue-team-wiki/blob/gh-pages/Tools/freq.py.md

Using Freq.py

```
>>>  from freq import *
>>>  fc = FreqCounter()
>>>  fc.load("english_lowercase.freq")
```

Pre-built frequency table

https://isc.sans.edu/diary/freq.py+super+powers%3F/19903

Using Freq.py

How did we score?

```
>>> fc.probability('forums/diary/Detecting+Random+Finding+Algorithmically+chosen+DNS+names+DGA/19893')
9.490788394012279

>>> fc.probability('Q09NRSBUTyBNWSBQWVRIT04gQ0xBU1MgSU4gVkVHQVMhISEh')
2.578357325221811
```

https://isc.sans.edu/diary/freq.py+super+powers%3F/19903

Freq-ing awesome…

Build Separate frequency tables for:

- DNS Host names that are normally seen for your environment
- Names of files on your file server
- Host names inside of SSL certificates
- URLs for a specific Web Application
- * Insert any string you want to measure here *

Runs as a restful API and integrates with logstash

https://isc.sans.edu/forums/diary/Continuous+Monitoring+for+Random+Strings/20451/

Freq.py - Operational Examples

DNS - Parent Domain Frequency Analysis

Domain	Frequency Score
tahmqkmvzg.com	0.434
dsms0mj1bbhn4.cloudfront.net	1.085
oiugixyixy.com	1.683
yfksjfsunyiypu.com	2.053
fwhynaxzwkv.com	2.329
yahoodns.net	2.697
muihoc.com	2.914
qfrwozmoaqc.com	3.153
aemmiphbweeuef59.com	3.345
brovztkzbl.com	3.391

SSL - Certificate Common Name Frequency Analysis

Common Name	Frequency Score
d3fr3g4tgwf	2.781
www.spidh.org	3.425
imap.gmx.net	3.96
www.lilawelt.net	4.538
www.paypal.com	5.324
www.google.com	5.454
www.jeffbryner.com	6.144
*.s3.amazonaws.com	6.185
s3.amazonaws.com	6.185
plesk3.gigaspark.com	6.458

Logstash configs created by Justin Henderson (@SecurityMapper)

https://github.com/Security-Onion-Solutions/security-onion/wiki/FreqServer

Phishing

Certificate Transparency

Certificate Transparency

https://blog.cloudflare.com/introducing-certificate-transparency-and-nimbus/

Streaming Phish

https://github.com/wesleyraptor/streamingphish

Streaming Phish

```
[Not Phishing]  paypal.com  0.004
[Not Phishing]  apple.com 0.003
[Not Phishing]  patternex.com 0.000
[Phishing]  support-apple.xyz 0.965
[Phishing]  paypall.com 0.851
[Phishing]  pavpal-verify.com 1.000
```

- "High" threshold is 0.90 and above
- "Suspicious" threshold is between 0.90 and 0.75
- "Low" threshold is between 0.75 and 0.60

Any FQDN with a score of 0.60 or lower will not be logged.

https://github.com/wesleyraptor/streamingphish/blob/master/jupyter/notebooks/StreamingPhish.ipynb

In Class Exercise

Please take 30 minutes and complete
Worksheet 4: Hunting with Data Science

Thank You!