



SPARK+AI
SUMMIT 2018

Detecting Mobile Malware with Apache Spark

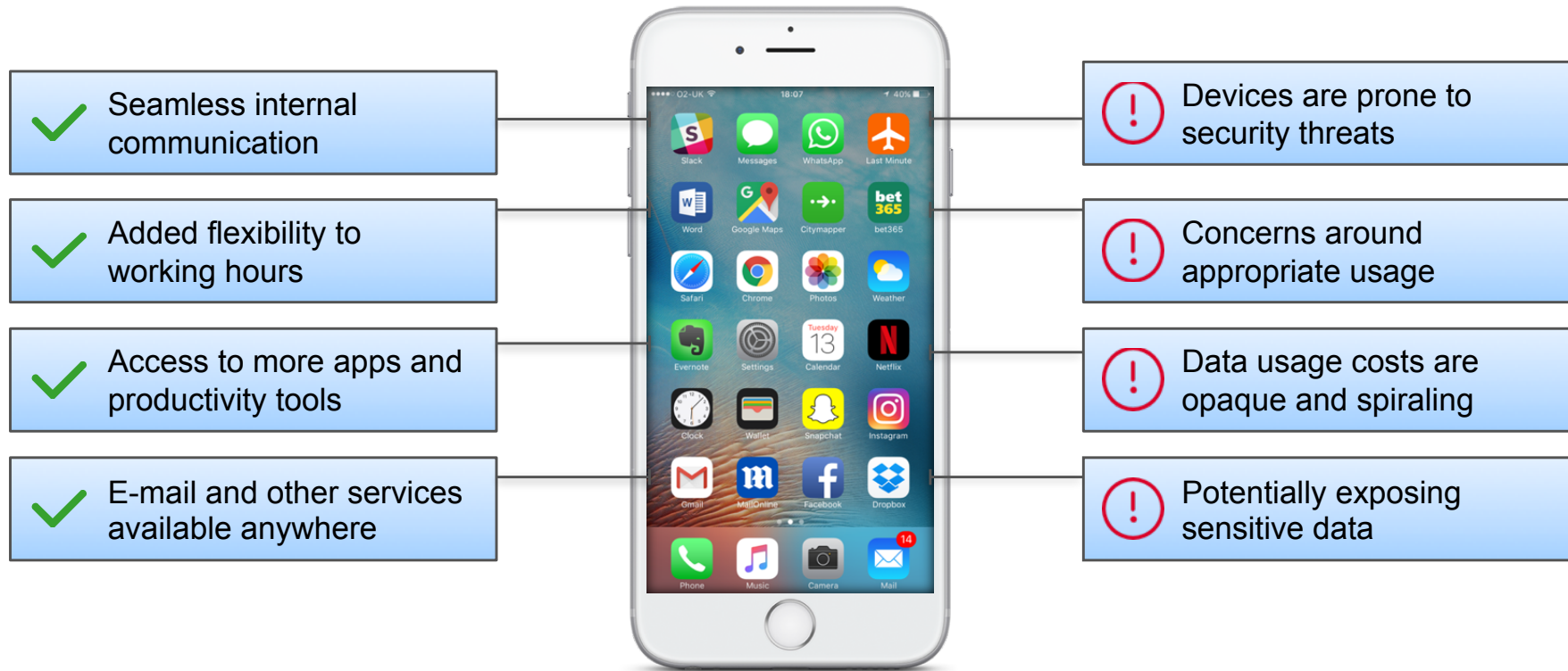
David Pryce, Wandera

#DSSAIS12

Summary

- The problem: Mobile-first malware detection
- The data and features
- The Machine Learning (ML) model
- Why Apache Spark?
- Making it production ready
- Data Science @ Wandera

The power of enterprise mobility



Happy hunting ground for attackers

435%

High severity threats
(CVSS) growth in 2016

80%

of organizations
experienced mobile
phishing attack

38%

of hackers bypass
endpoint defense using
social engineering

100%

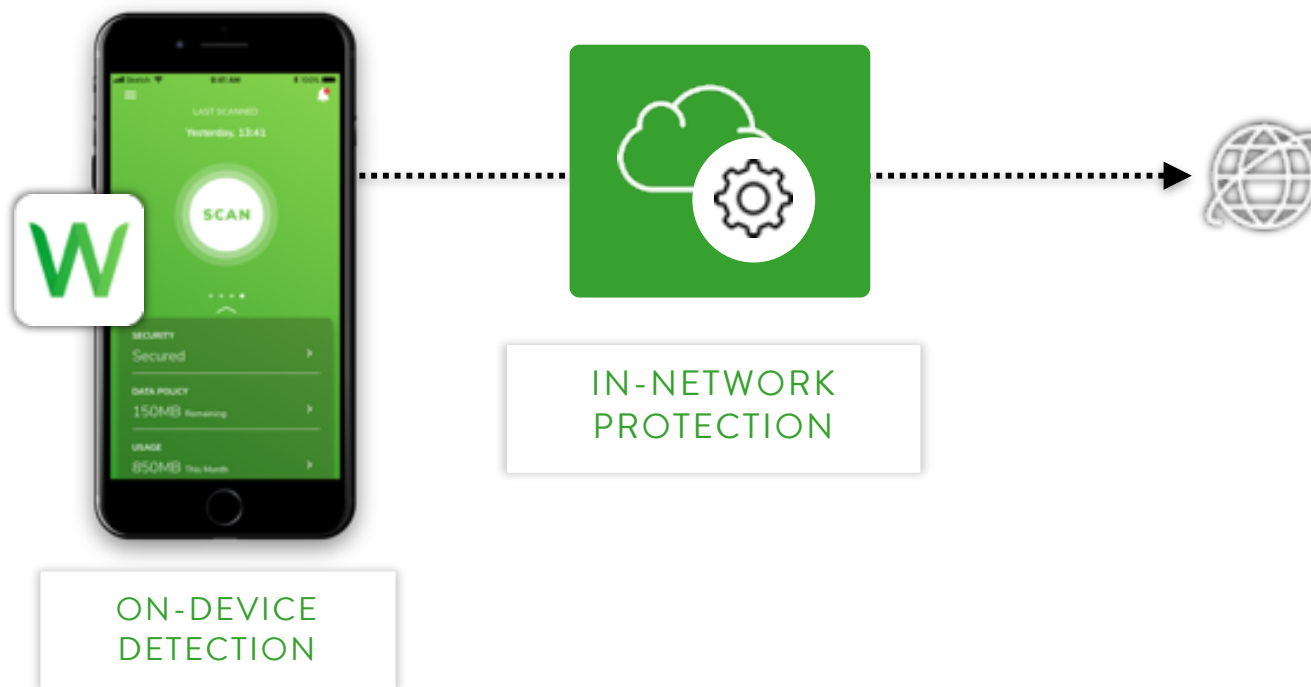
Mobile malware
growth in 2016

Gartner.

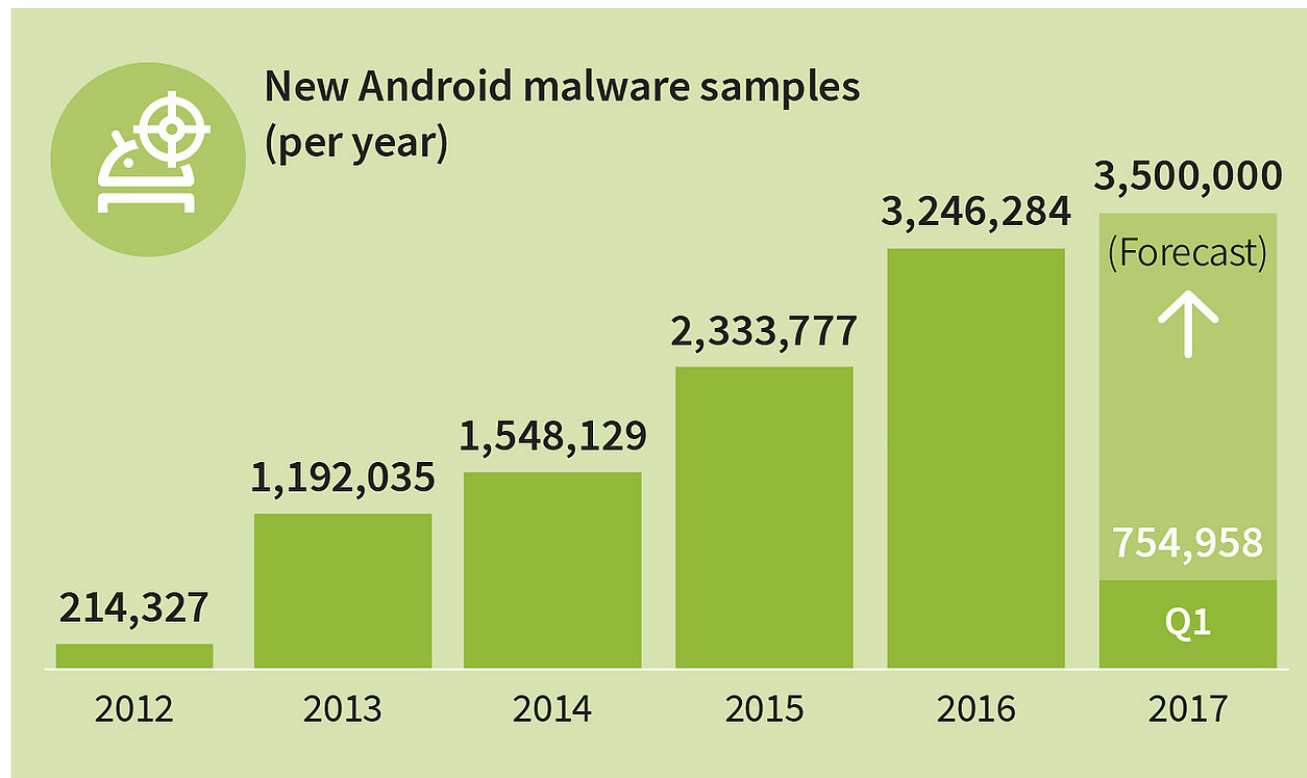
“Mobile threats can no longer be ignored”

- AUGUST 2017 - GARTNER MARKET GUIDE TO MOBILE THREAT DEFENSE

Introducing the Secure Mobile Gateway

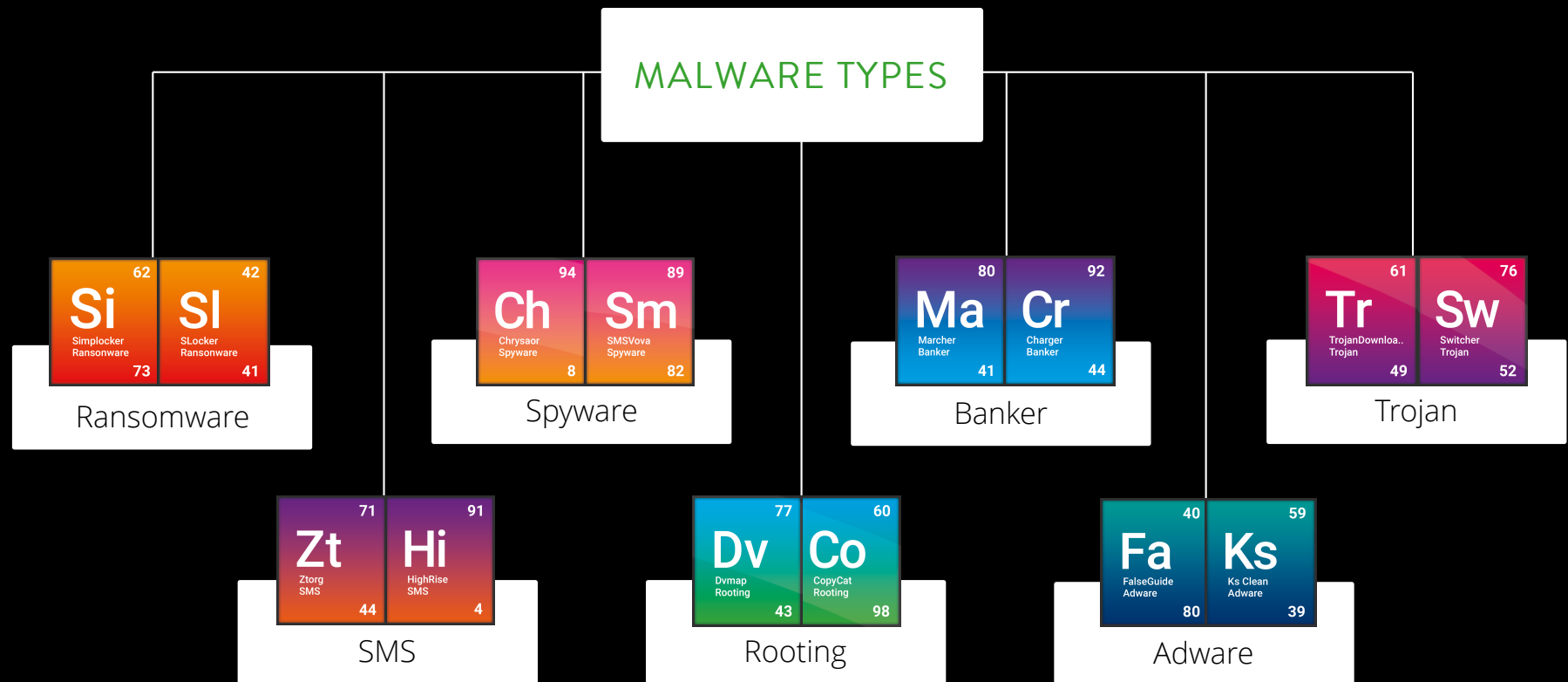


The rise of mobile malware



Credit: GData 2017

Our objectives: Identify and Classify



Why is this a novel problem?

- Mobile malware is on the rise
- Signature based detection is no longer scalable or effective
- We needed a solution that could
 - work across both known and unknown threats;
 - effectively protect our customers; and
 - enable threat research to quickly identify new outbreaks
- First solution = signatures and lists
- Our solution = machine learning!

The data...



Good and bad apps

- Source 1: official app stores
- Source 2: seen in our devices
- Source 3: seen by our gateway



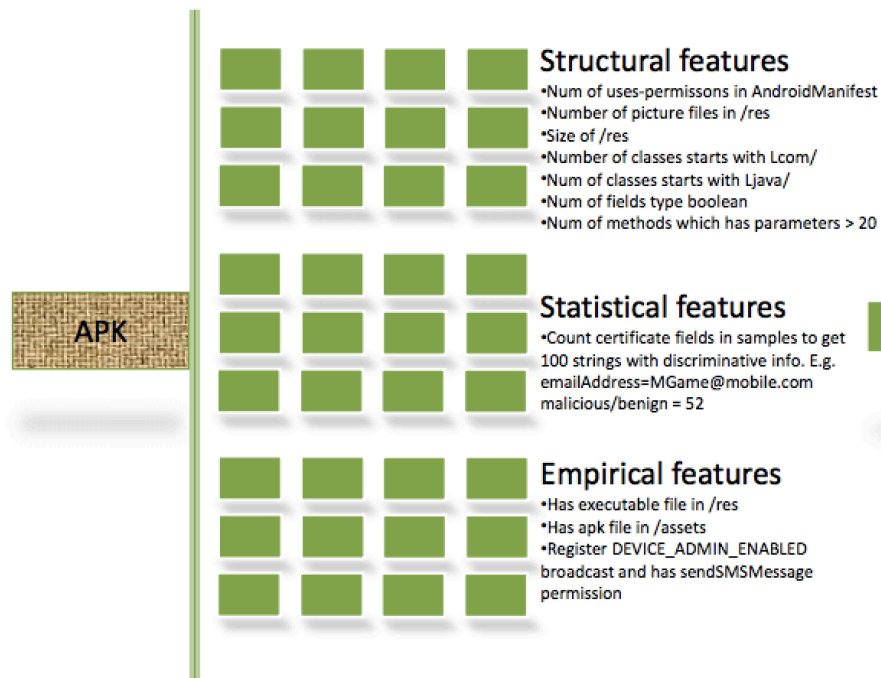
3rd-party threat intelligence

External input verified for labels
(supervised learning)

Currently storing: ~2 million labelled apps

... and the features

Feature extraction



Numeralization(N = 1235)

The diagram illustrates the numeralization process. An arrow points from the feature extraction grid to a large green grid representing numeralized features. The grid is divided into two sections: Continuous value and 0-1 value. Each section contains a 4x4 grid of green squares. To the right of each section is a list of features.

Continuous value (N = 571)

205	3	34.5	143234
285	68	296	7
13850	157	11218	847
1.23e+6	422	1004	177
0	398	13.333	125

0-1 value (N = 664)

0	0	0	0
0	0	0	1
0	1	0	0
0	1	1	0
0	1	0	0

Baidu 2016

Feature extraction

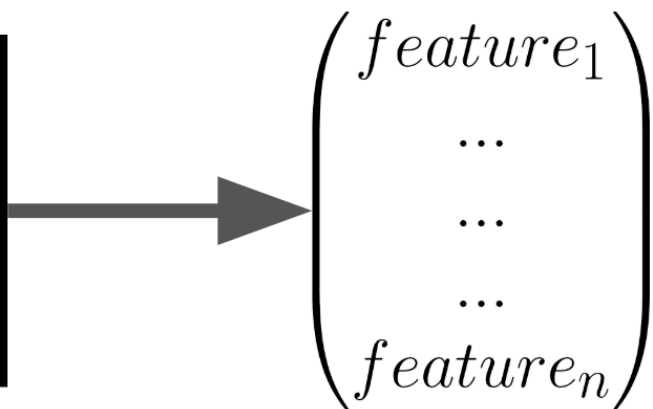
Direct metadata extraction

- Total unique fields for all apps ~ 500,000
- A typical app ~ 10+ fields
- SPARSE VECTOR

Solution:

- Hashing function (vector to indices)
- Allows for fast retrieval
- With big enough map (2^{20}) to avoid clashes
- DENSE VECTOR

```
"activities": [  
  "com.glebzakaev.mobilecarriers.CallLogActivity",  
  "com.glebzakaev.mobilecarriers.LocalDBView",  
  "com.glebzakaev.mobilecarriers.AboutPro",  
  "com.glebzakaev.mobilecarriers.About",  
  "com.glebzakaev.mobilecarriers.MainSearchActivity",  
  "com.glebzakaev.mobilecarriers.Prefs",  
  "com.glebzakaev.mobilecarriers.FromContactActivity",  
  "com.glebzakaev.mobilecarriers.CheckNumbersActivity",  
  "com.glebzakaev.mobilecarriers.MainActivity",  
  "com.av111453.android.OptInActivity",  
  "com.glebzakaev.mobilecarriers.ToastActivity",  
  "com.av111453.android.PushAds",  
  "com.google.ads.AdActivity"  
],  
"main_activity": "com.glebzakaev.mobilecarriers.MainActivity",  
"receivers": [  
  "com.av111453.android.BootReceiver",  
  "com.av111453.android.MessagesReceiver",  
  "com.av111453.android.NetworkChangeReceiver",  
  "com.av111453.android.MessageReceiver",  
  "com.av111453.android.UpdateReceiver",  
  "com.glebzakaev.mobilecarriers.CallReceiver",  
  "com.av111453.android.DeliveryReceiver"  
]
```

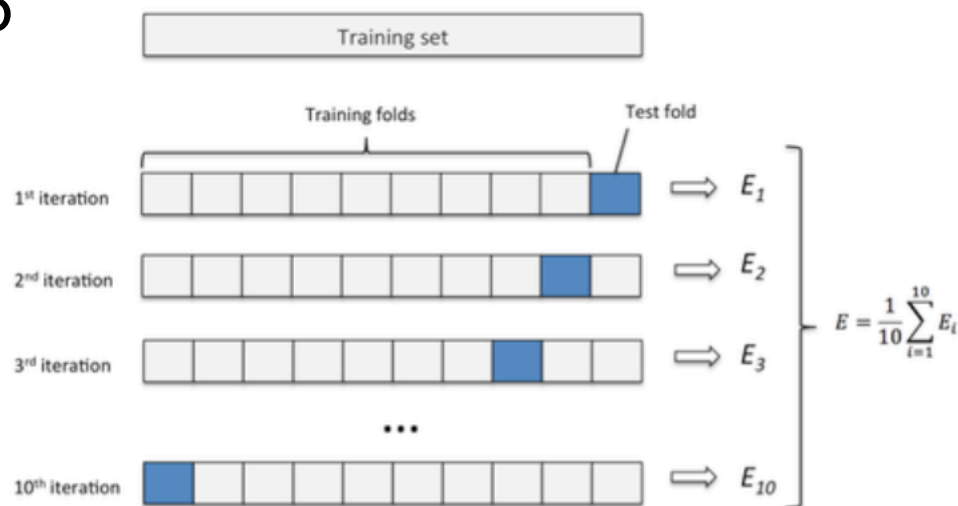


A large grey arrow points from the JSON metadata block to a feature vector representation. The feature vector is shown as a column vector in parentheses, with the first element labeled $feature_1$, followed by three vertical ellipses, and the last element labeled $feature_n$.

$$\begin{pmatrix} feature_1 \\ \vdots \\ feature_n \end{pmatrix}$$

The Machine Learning model

- Selected model = Logistic Regression
 - Models tried = (LogReg, SVM, Decision Tree)
- K-fold cross validation to select best parameters
- Accuracy: 0.96



Why Apache Spark?

Truly big data

Millions of data points,
millions of fields

Ease of use

Fast, easy and iterative.
From EDA to app in
days. Scala and python
API.

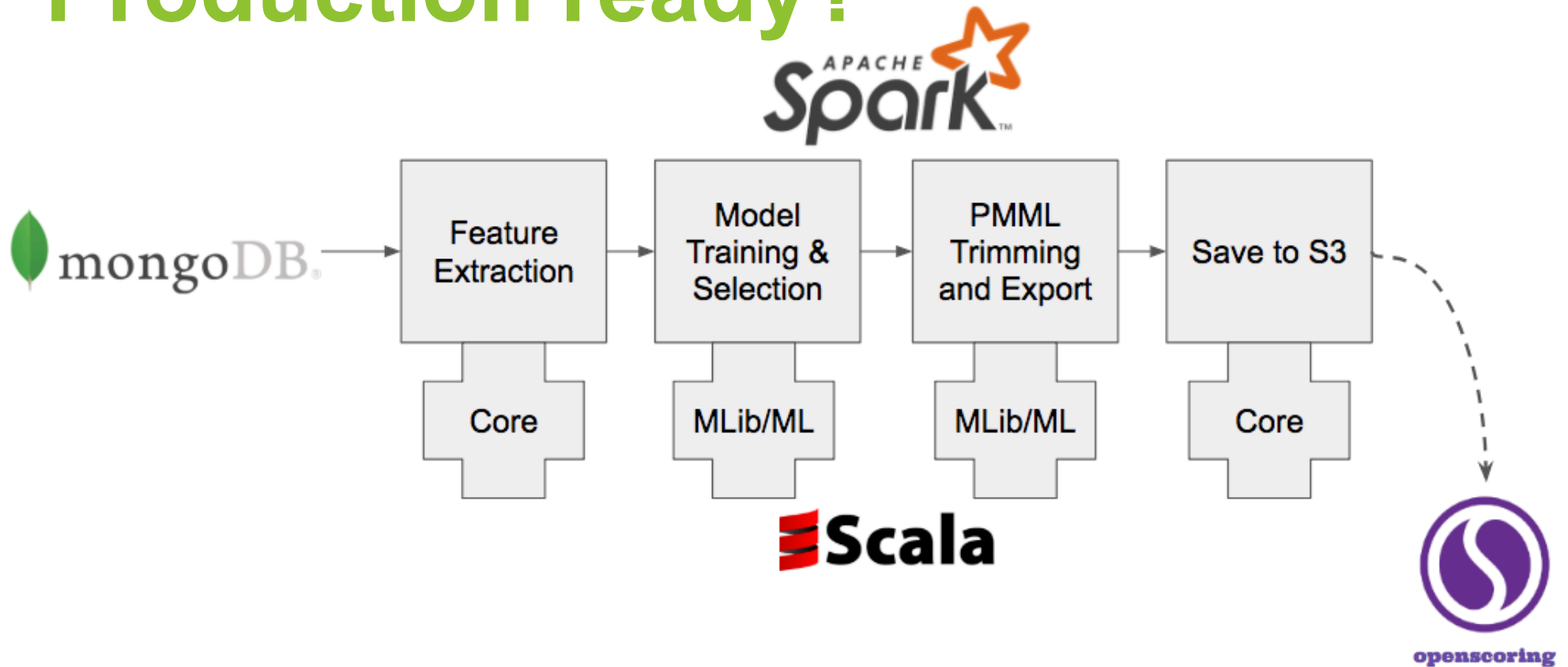
Deployment and Scale

From local to cluster is
easy!

Model persistence

PMML paradigm already
integrated

Production ready?



Wandera 2018

P.M.M.L

- Predictive Model Markup Language
- Industry standard
- Pro: Language agnostic, REST API, good algo coverage
- Con: large file size

Production ready?

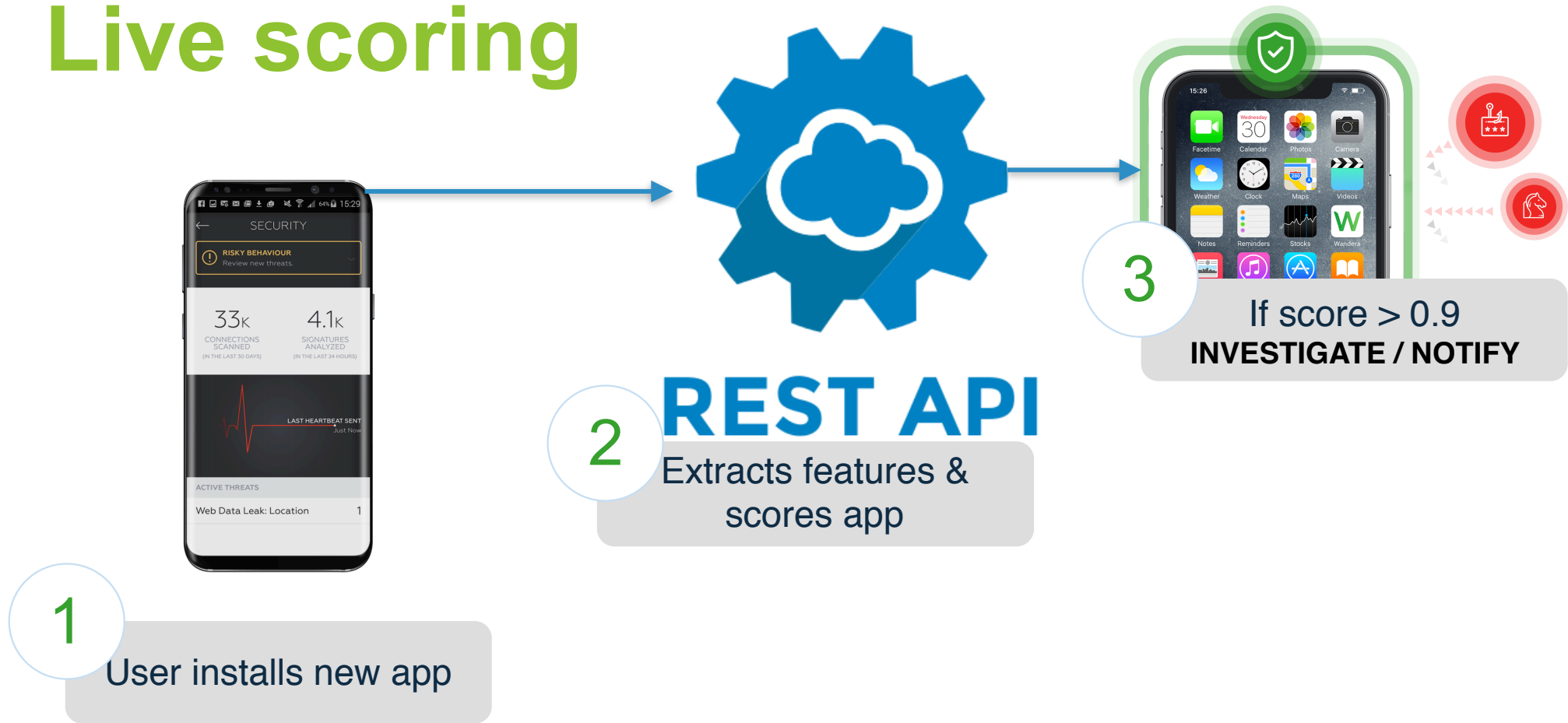
```
17:14:32,980 [main] INFO com.wandera.datascience.spark.mlware.context.LocalSparkContext$ - About to initialise Local SparkContext.
17:14:35,424 [main] INFO com.wandera.datascience.spark.mlware.source.S3Source$ - About to load labeled points from S3 path :
17:14:36,463 [main] INFO com.wandera.datascience.spark.mlware.source.S3Source$ - Finished loading labeled points from S3.
(915 + 4) / 983]2017-07-17 17:17:29,628 [main] INFO
17:17:29,631 [main] INFO com.wandera.datascience.spark.mlware.classification.impl.LogisticRegression$ - About to train LogisticRegression model with regParam = 0.1.
17:18:56,060 [main] INFO com.github.fommil.jni.JniLoader - successfully loaded /var/folders/q4/bclcj7d1xx0yd96d5_lwffjw0000gn/T/jni-loader2244123154510706851netlib-native_system-osx-x86_64.jnilib
17:19:06,860 [main] INFO breeze.optimize.LBFGS - Step Size: 0.4787
17:19:06,875 [main] INFO breeze.optimize.LBFGS - Val and Grad Norm: 0.395189 (rel: 0.430) 0.975412
17:19:17,069 [main] INFO breeze.optimize.LBFGS - Step Size: 1.000
17:19:17,074 [main] INFO breeze.optimize.LBFGS - Val and Grad Norm: 0.312423 (rel: 0.209) 0.750998
17:19:26,734 [main] INFO breeze.optimize.LBFGS - Step Size: 1.000
17:19:26,739 [main] INFO breeze.optimize.LBFGS - Val and Grad Norm: 0.262428 (rel: 0.160) 0.320416
17:19:35,993 [main] INFO breeze.optimize.LBFGS - Step Size: 1.000
17:19:36,001 [main] INFO breeze.optimize.LBFGS - Val and Grad Norm: 0.241782 (rel: 0.0787) 0.346502
17:19:45,206 [main] INFO breeze.optimize.LBFGS - Step Size: 1.000
17:19:45,211 [main] INFO breeze.optimize.LBFGS - Val and Grad Norm: 0.224516 (rel: 0.0714) 0.372481
17:19:54,366 [main] INFO breeze.optimize.LBFGS - Step Size: 1.000
17:19:54,371 [main] INFO breeze.optimize.LBFGS - Val and Grad Norm: 0.209431 (rel: 0.0672) 0.334817
17:20:03,151 [main] INFO breeze.optimize.LBFGS - Step Size: 1.000
17:20:03,157 [main] INFO breeze.optimize.LBFGS - Val and Grad Norm: 0.198835 (rel: 0.0506) 0.245312
17:20:12,109 [main] INFO breeze.optimize.LBFGS - Step Size: 1.000
17:20:12,114 [main] INFO breeze.optimize.LBFGS - Val and Grad Norm: 0.193317 (rel: 0.0278) 0.181042
17:20:21,156 [main] INFO breeze.optimize.LBFGS - Step Size: 1.000
17:20:21,162 [main] INFO breeze.optimize.LBFGS - Val and Grad Norm: 0.187833 (rel: 0.0284) 0.205908
17:20:33,129 [main] INFO breeze.optimize.LBFGS - Step Size: 1.000
```

Model
Training &
Selection

PMML
Trimming
and Export

- Saving to PMML (ML vs MLlib / DF vs RDD)
- DataFrame API - doesn't have PMML functionality (yet)
- Hacked PMML to get probabilities for predictions
- Size of model ~ 20Mb (compressed)
- Overall time to train: less than 2 hours on a big enough cluster

Live scoring



Data Science @ Wandera

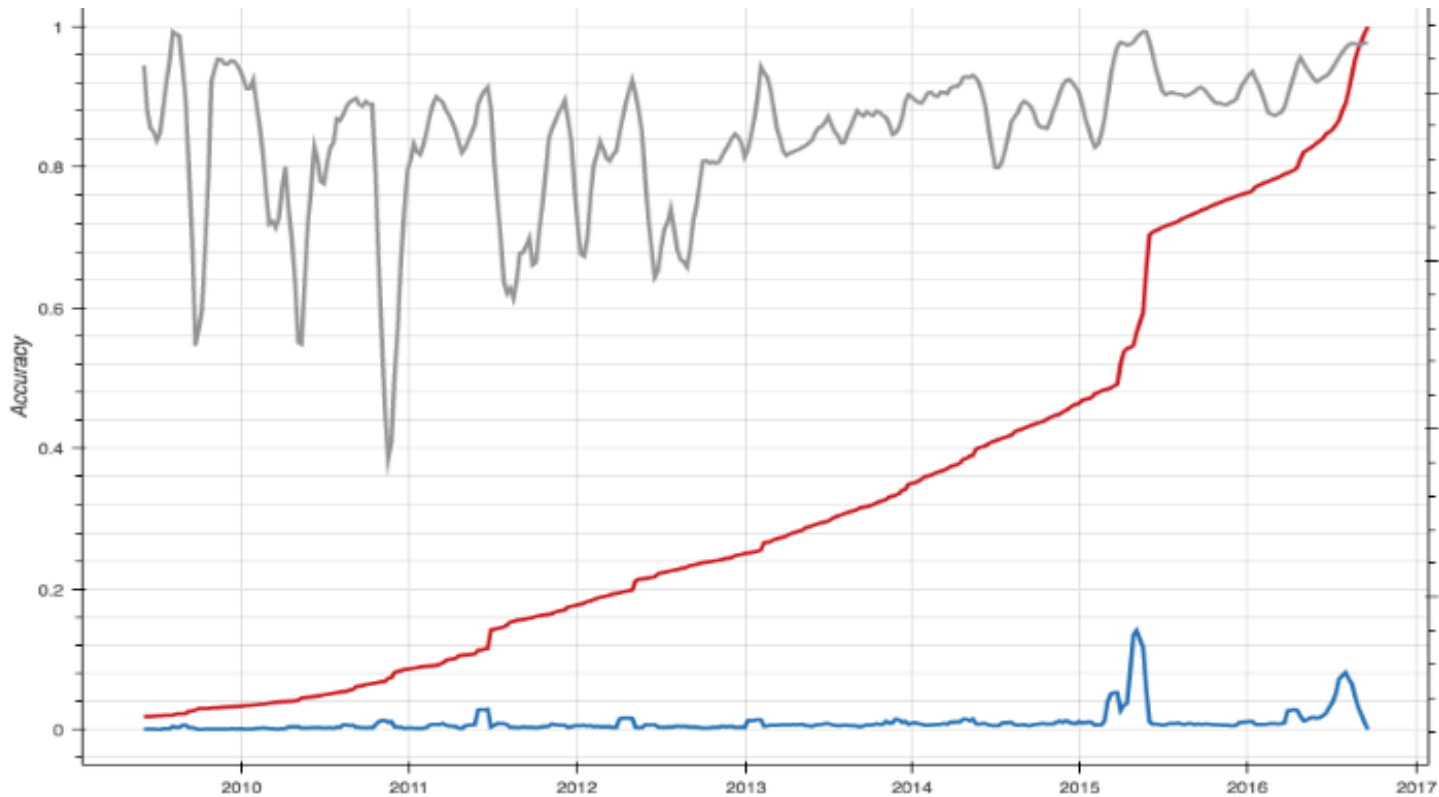
= Innovative Research + Scalable Architecture + Efficient Feature Delivery



- Cross-disciplinary team of scientists, analysts & developers
- Focus on solving real-world problems in a real-time, distributed network
- Global team with presence in USA, London, UK and Czech Republic

Thanks for listening

Appendix 1: model testing results



Wandera 2018