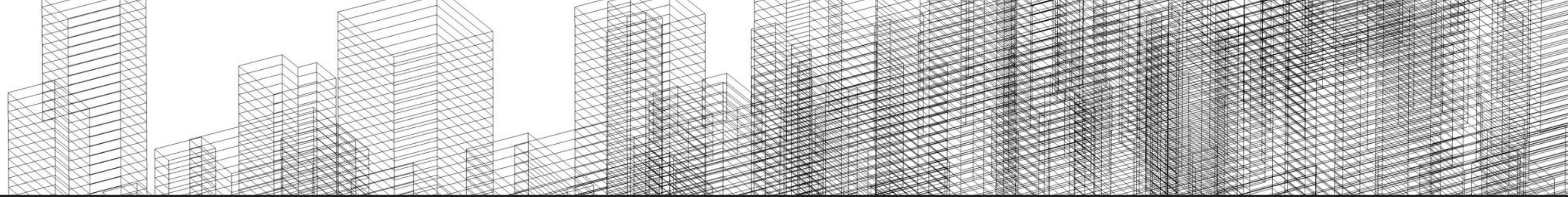
Batch and Online Anomaly Detection for Scientific Applications in a Kubernetes Environment

Matias Carrasco Kind
Senior Research Scientist, NCSA

Sahand Hariri
PhD Student, MechSE UIUC

ScienceCloud 18': 9th Workshop on Scientific Cloud Computing
June 11th, Tempe, AZ



Overview

Goal: Build a resilient scalable anomaly detection service.

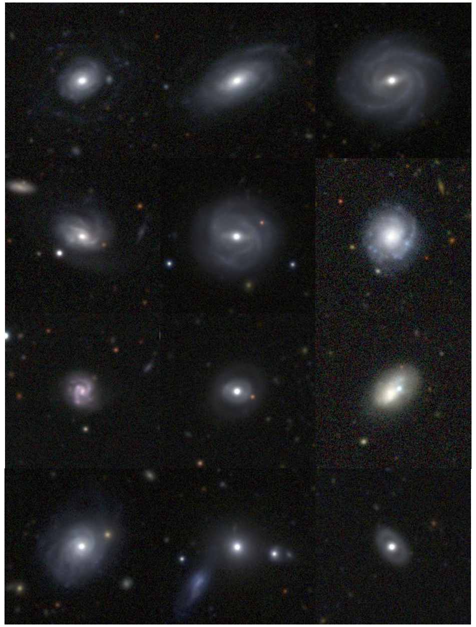
Motivation: Astronomical data (both literal and figurative)

Algorithm: Extended Isolation Forest

Infrastructure: Kubernetes cluster

Mapreduce package: Spark

Part of the Motivation



Astronomy is just one example where data exploration needs to be automated.

Large catalogs, Large number of images, many unexpected objects/problems → <u>Anomaly detection</u>

Large Synoptic Survey Telescope

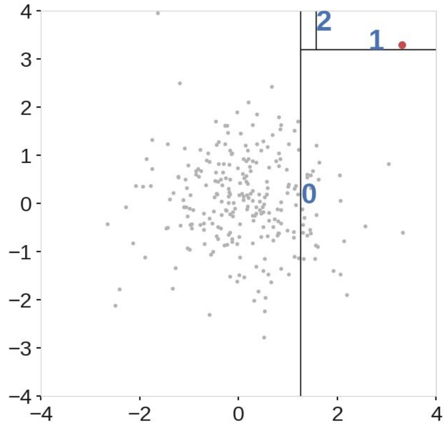
- In operations 2020
- Every night for 10 years
- 18 billions objects (first year), ~40 billions by the end of survey
- ~1500 images per night
- Stream and static data
- Target to capture new physics (moving and variable objects)

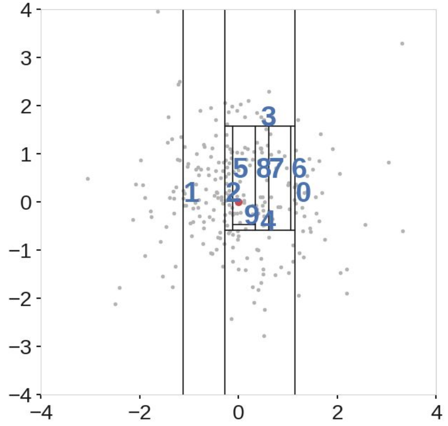
DARK ENERGY SURVEY

- More than 500 nights of observation over 5 years
- 500 millions cataloged galaxies and 100 millions stars
- Many open problems: Systematics, new objects, new physics, etc.
- Almost completed

Anomaly Detection with Isolation Forest

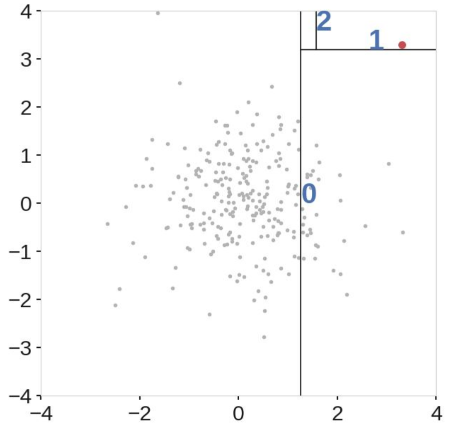
- Few and different to be isolated quicker

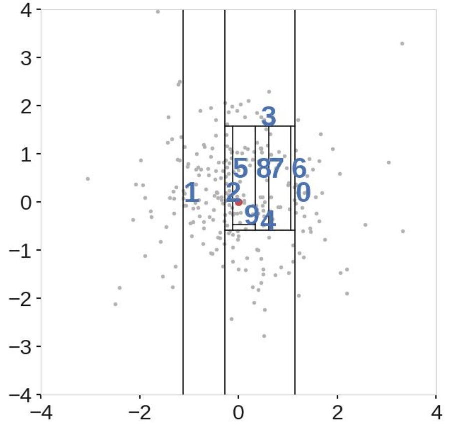




Anomaly Detection with Isolation Forest

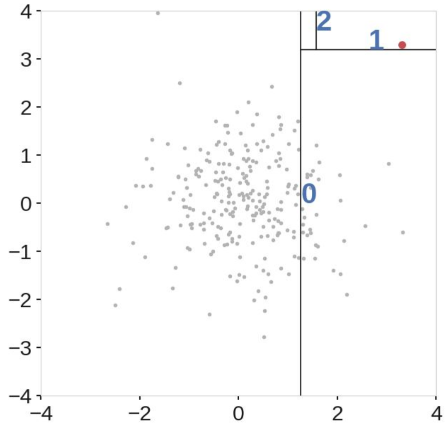
- Few and different to be isolated quicker
- For each tree:
    - Get a sample of the data

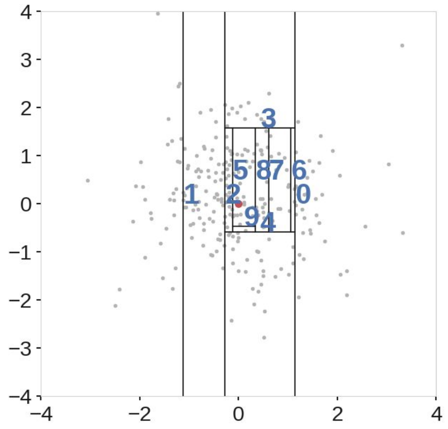




Anomaly Detection with Isolation Forest

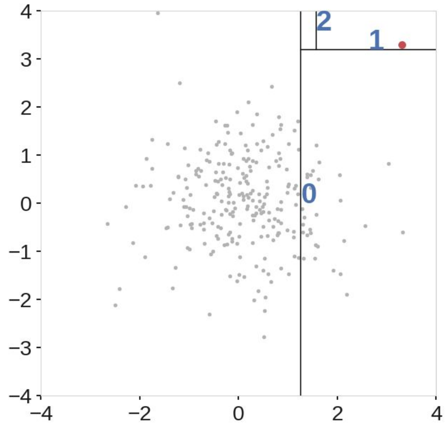
- Few and different to be isolated quicker
- For each tree:
  - Get a sample of the data
  - Randomly select a dimension
  - Randomly pick a value in that dimension

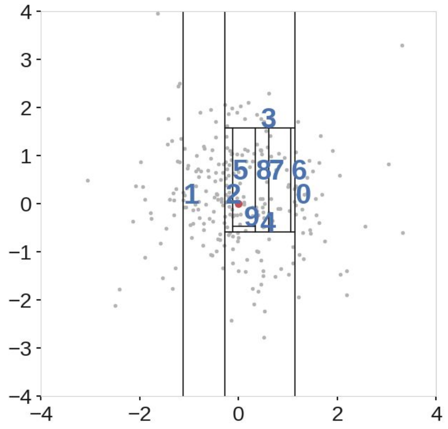




Anomaly Detection with Isolation Forest

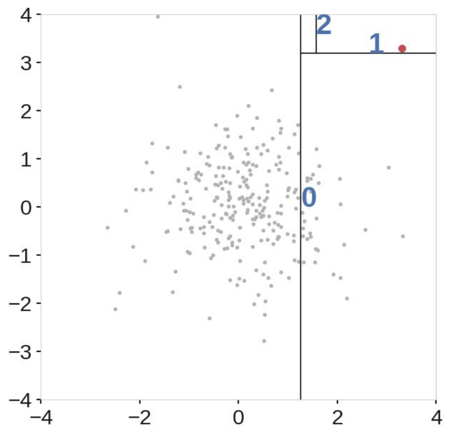
- Few and different to be isolated quicker
- For each tree:
  - Get a sample of the data
  - Randomly select a dimension
  - Randomly pick a value in that dimension
  - Draw a straight line through the data at that value and split data

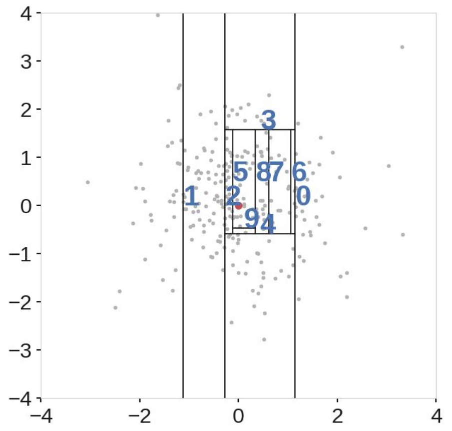




Anomaly Detection with Isolation Forest

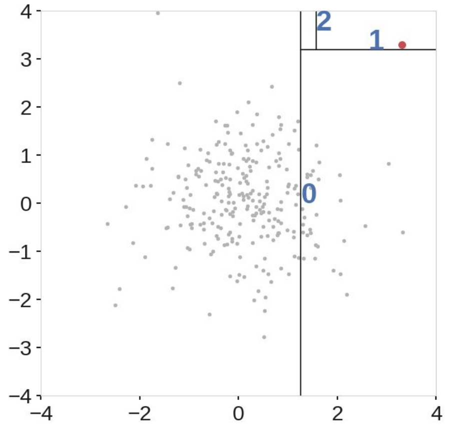
- Few and different to be isolated quicker
- For each tree:
  - Get a sample of the data
  - Randomly select a dimension
  - Randomly pick a value in that dimension
  - Draw a straight line through the data at that value and split data
  - Repeat until tree is complete

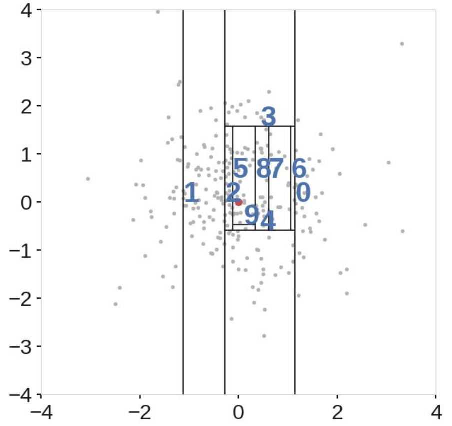




Anomaly Detection with Isolation Forest

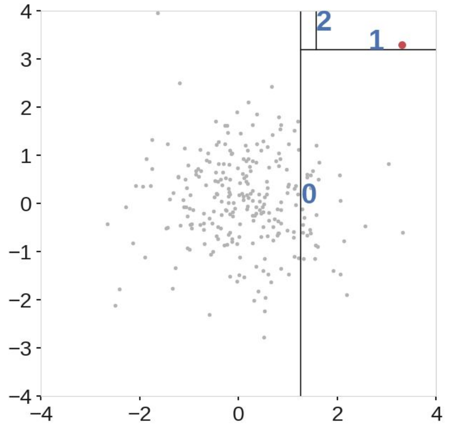
- Few and different to be isolated quicker
- For each tree:
  - Get a sample of the data
  - Randomly select a dimension
  - Randomly pick a value in that dimension
  - Draw a straight line through the data at that value and split data
  - Repeat until tree is complete
- Generate multiple trees → forest

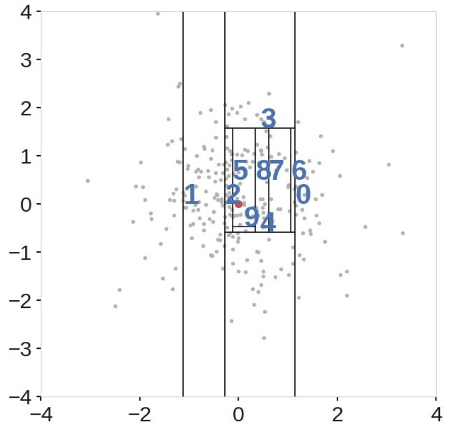




Anomaly Detection with Isolation Forest

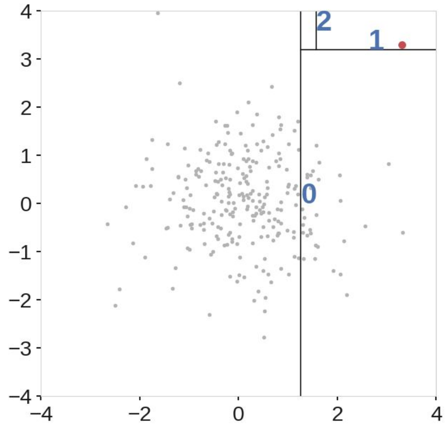
- Few and different to be isolated quicker
- For each tree:
  - Get a sample of the data
  - Randomly select a dimension
  - Randomly pick a value in that dimension
  - Draw a straight line through the data at that value and split data
  - Repeat until tree is complete
- Generate multiple trees → forest
- Anomalies will be isolated in only a few steps

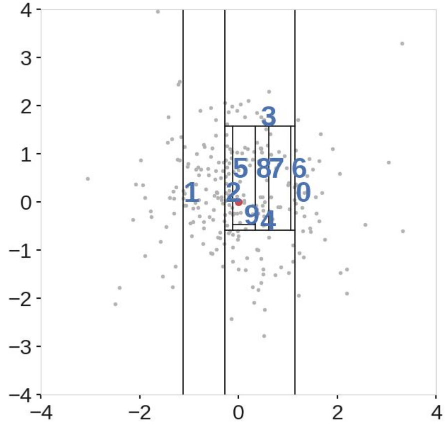




Anomaly Detection with Isolation Forest

- Few and different to be isolated quicker
- For each tree:
  - Get a sample of the data
  - Randomly select a dimension
  - Randomly pick a value in that dimension
  - Draw a straight line through the data at that value and split data
  - Repeat until tree is complete
- Generate multiple trees → forest
- Anomalies will be isolated in only a few steps
- Nominal points in more

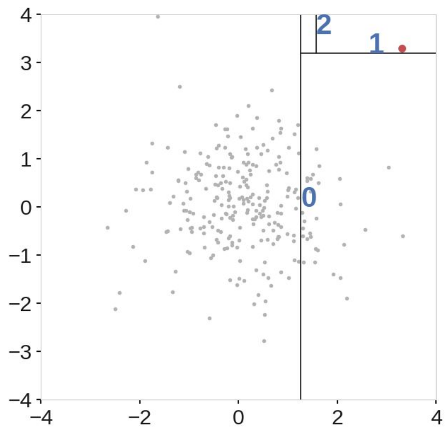


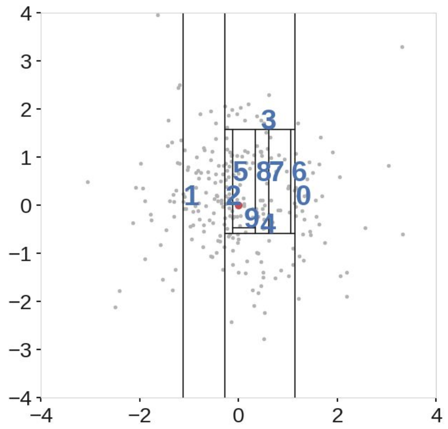


Anomaly Detection with Isolation Forest

- Few and different to be isolated quicker
- For each tree:
  - Get a sample of the data
  - Randomly select a dimension
  - Randomly pick a value in that dimension
  - Draw a straight line through the data at that value and split data
  - Repeat until tree is complete
- Generate multiple trees $\longrightarrow$ forest
- Anomalies will be isolated in only a few steps
- Nominal points in more

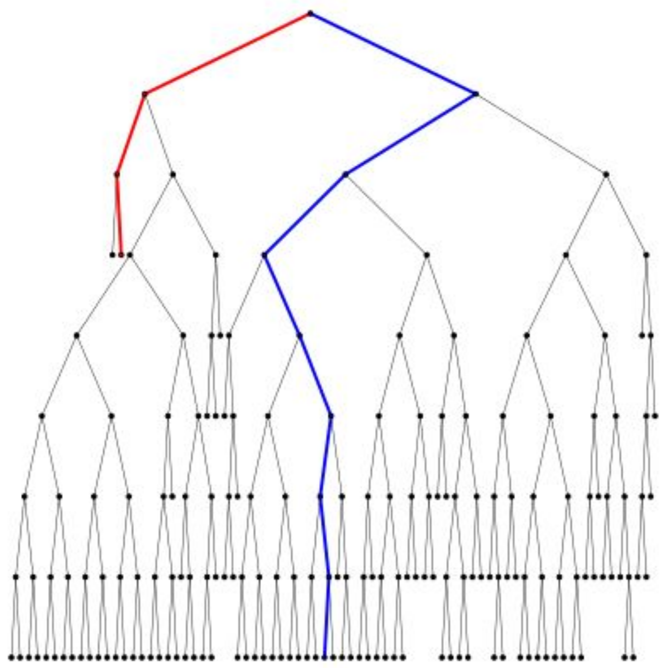
- To score points:
  - Run point down tree, record path
  - Repeat for each tree, aggregate scores $s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$
  - Score distribution



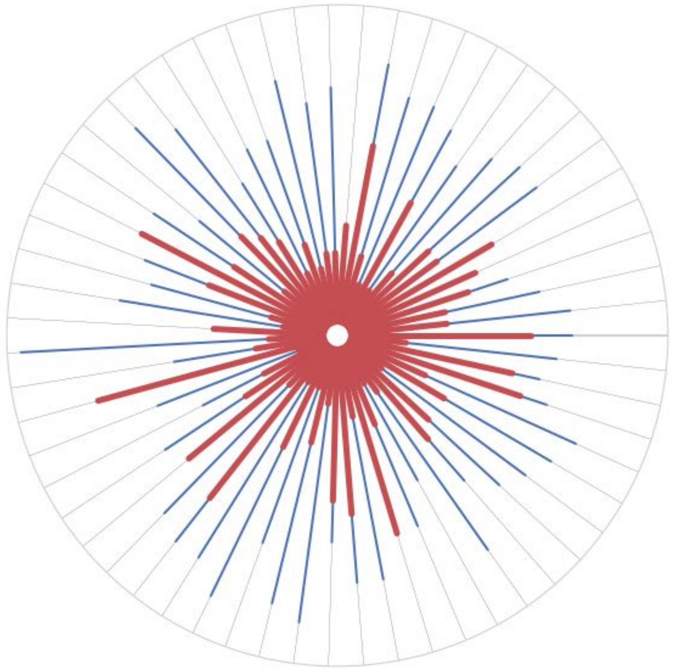


Anomaly Detection with Isolation Forest

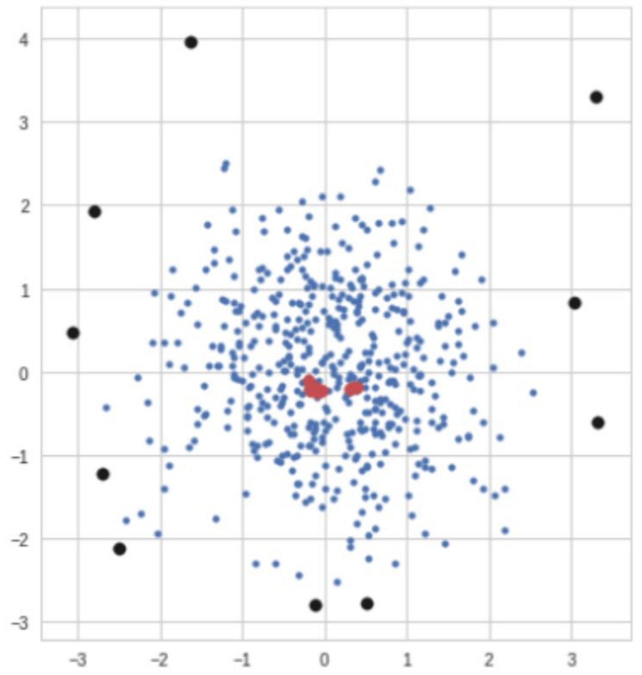
Single Tree scores for
anomaly and nominal points

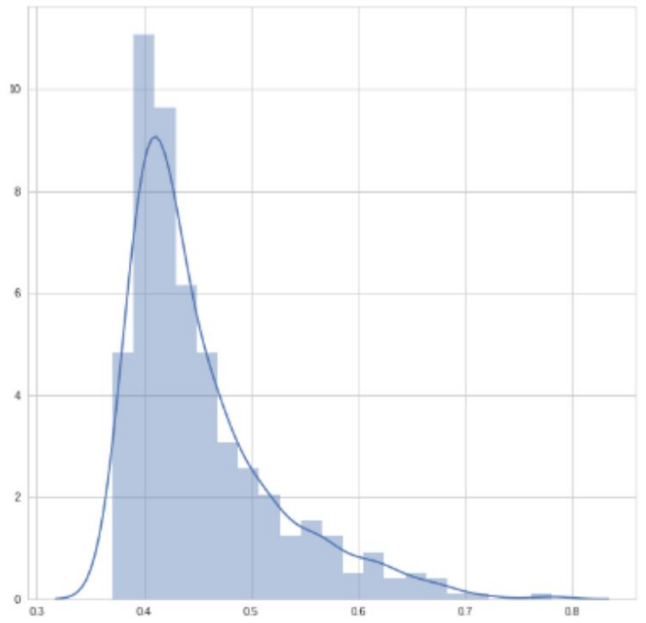


Forest plotted radially.
Scores for anomaly and
nominal shown as lines



Anomaly Detection with Isolation Forest

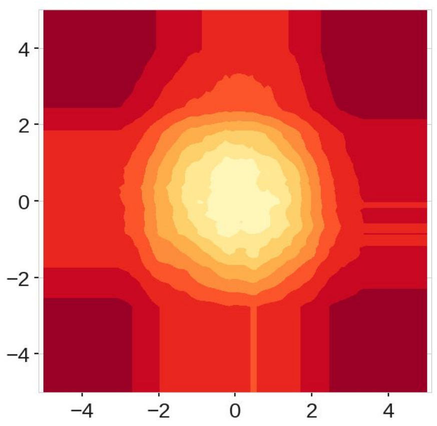


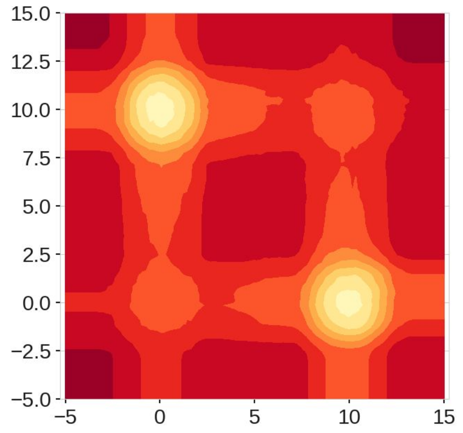


Anomaly Detection with Extended Isolation Forest

Isolation Forest:

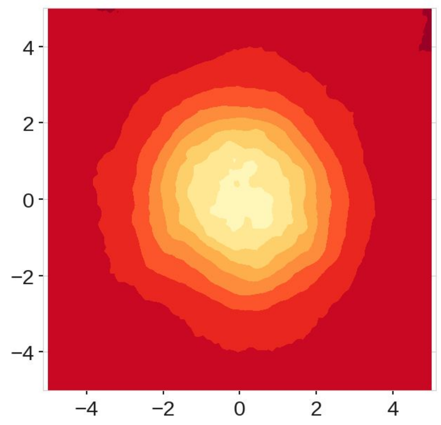
- ✓ Model free
- ✓ Computationally efficient
- ✓ Readily applicable to parallelization
- ✓ Readily application to high dimensional data
- ✕ Inconsistent scoring seen in score maps

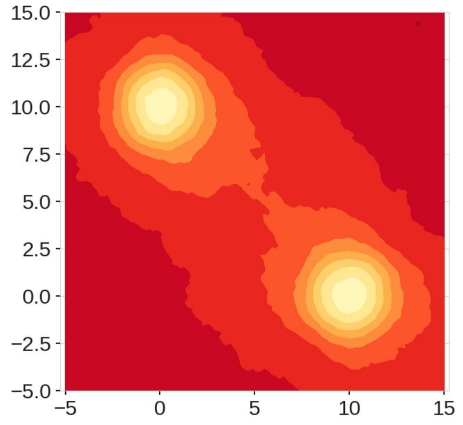




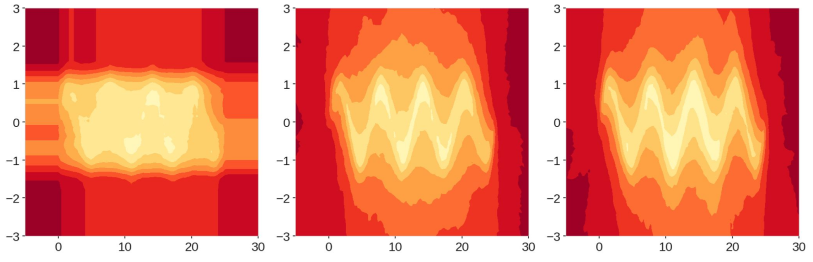
Extended Isolation Forest:

- ✓ Model free
- ✓ Computationally efficient
- ✓ Readily applicable to parallelization
- ✓ Readily application to high dimensional data
- ✓ Consistent scoring



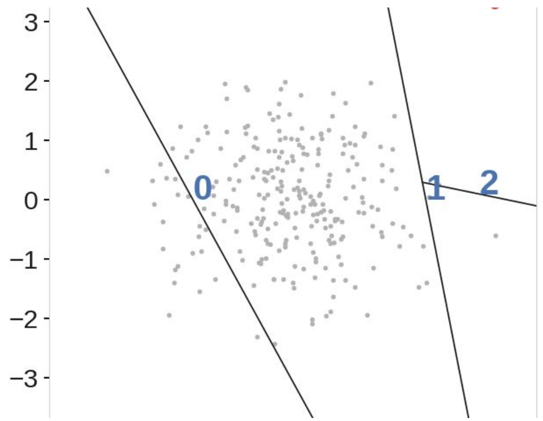


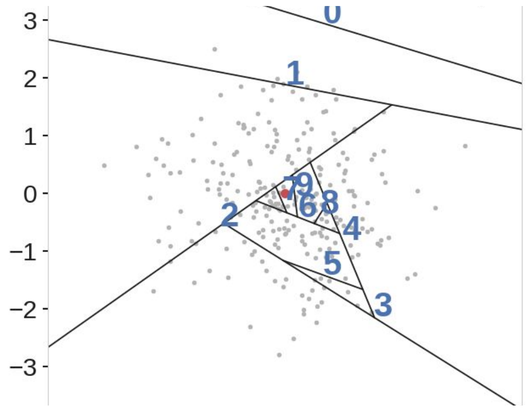
Anomaly Detection with Extended Isolation Forest



(a) Standard IF (b) Rotated IF (c) Extended IF

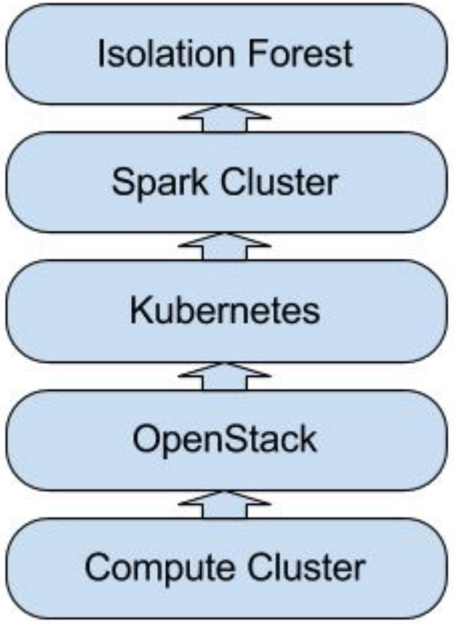
```
Algorithm 2 iTree(X, e, l)
Require: X - input data, e - current tree height, l - height limit
Ensure: an iTree
1:  if $e \geq l$ or $|X| \leq 1$ then
2:      return $exNode\{Size \leftarrow |X|\}$
3:  else
4:      randomly select a normal vector $n \in \mathbb{R}^{|X|}$ by drawing each coordinate of $\vec{n}$ from a uniform
    distribution.
5:      randomly select an intercept point $p \in \mathbb{R}^{|X|}$ in the range of $X$
6:      set coordinates of $n$ to zero according to extension level
7:      $X_l \leftarrow filter(X, (X - p) \cdot n \leq 0)$
8:      $X_r \leftarrow filter(X, (X - p) \cdot n > 0)$
9:      return inNode{ $Left \leftarrow iTree(X_l, e + 1, l)$,
                      $Right \leftarrow iTree(X_r, e + 1, l)$,
                      $Normal \leftarrow n$,
                      $Intercept \leftarrow p$}
10: end if
```





Technology Stack For Anomaly Service

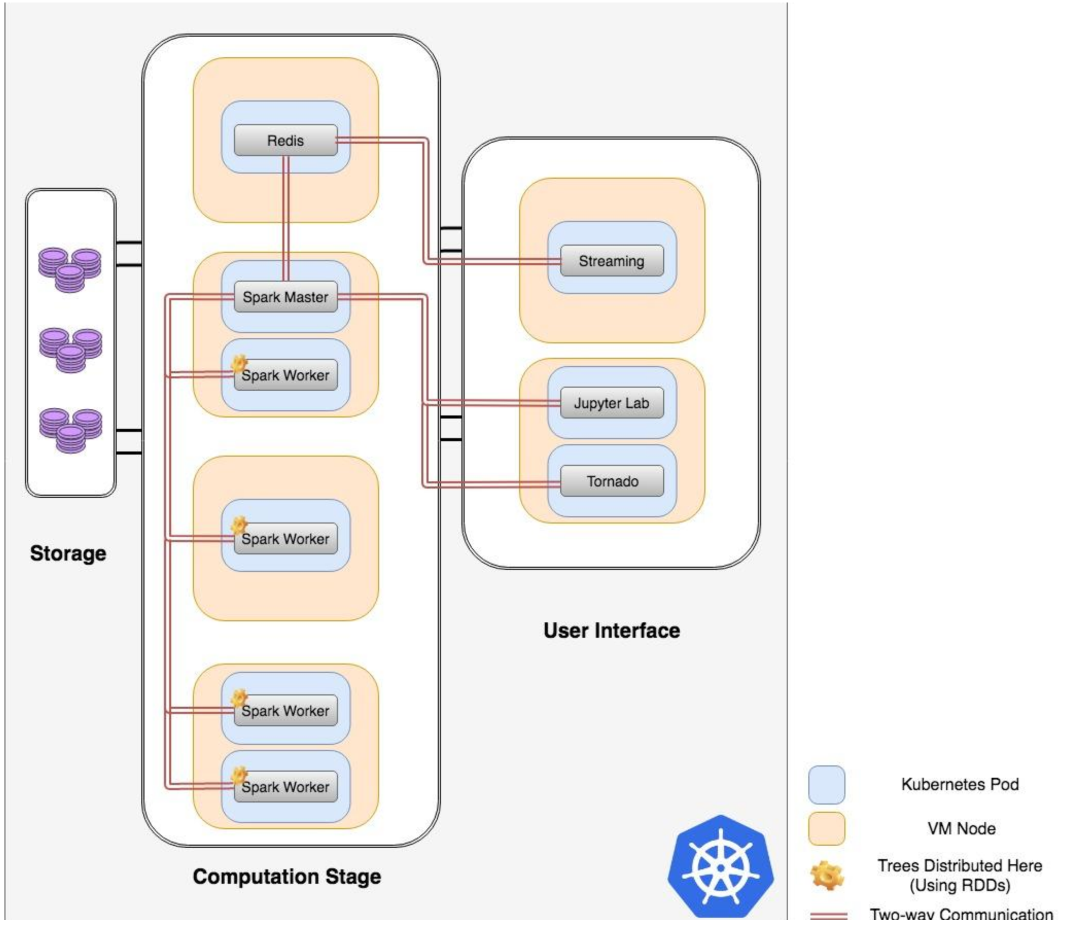
- Use Extended Isolation Forest as core algorithm
- Use Spark to parallelize trees and scoring
- Use Redis as a broker communicator
- To easily deploy in any environment, use Docker
- For orchestration of Docker containers, use Kubernetes
- Kubernetes cluster built on top of OpenStack, but it can be deployed also in AWS, GKE, etc.



Framework Architecture

There are three main components:

1. Storage
2. Computation Stage
3. User Interface / Streaming



Framework Architecture

Storage:

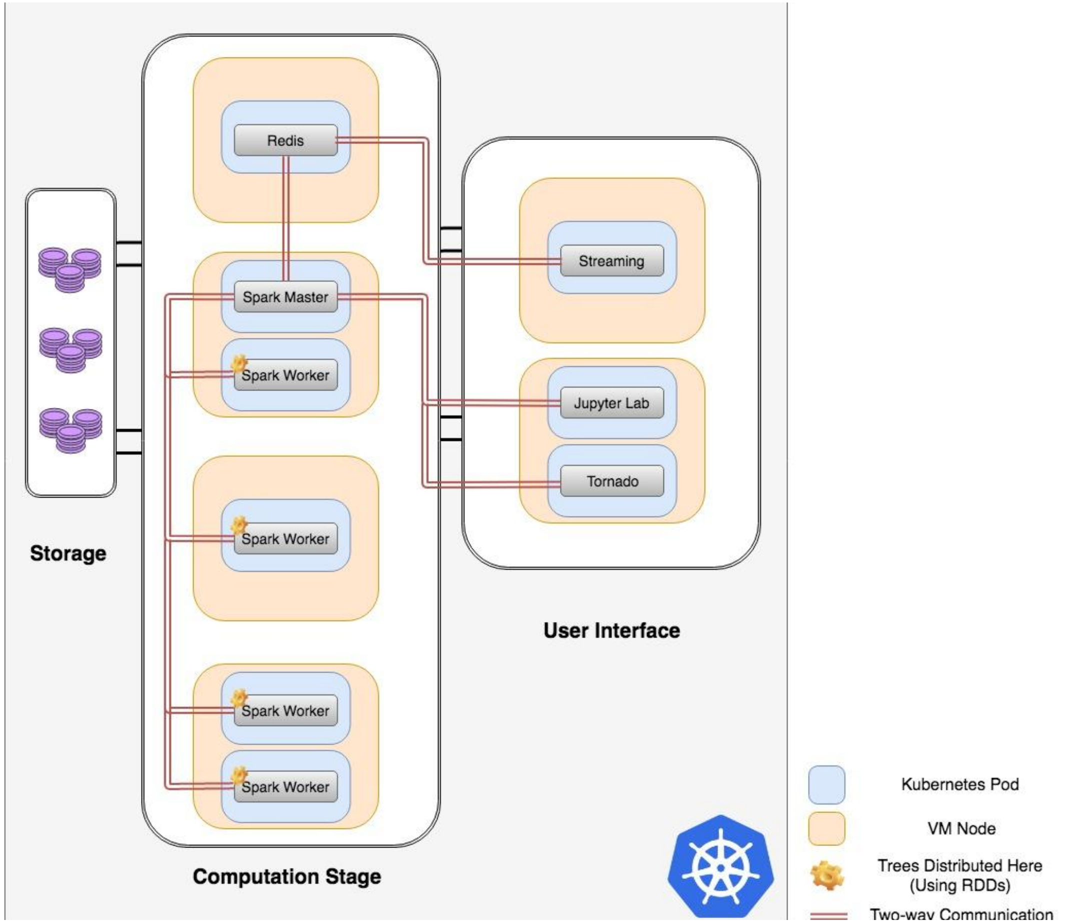
- NFS (Kubernetes PV/PVC)
- Redis
- RDD for Trees and Spark

User Interface:

- Jupyter notebooks
- Interactive web app for submitting jobs
- Streaming service

Computation Stage:

- Spark Master and Workers
- Communicator with Spark Master
- Suscripcion

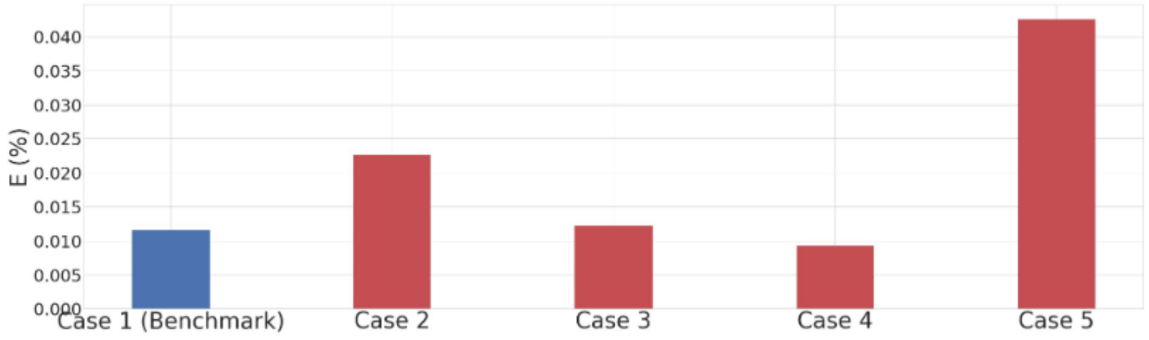


Deployment

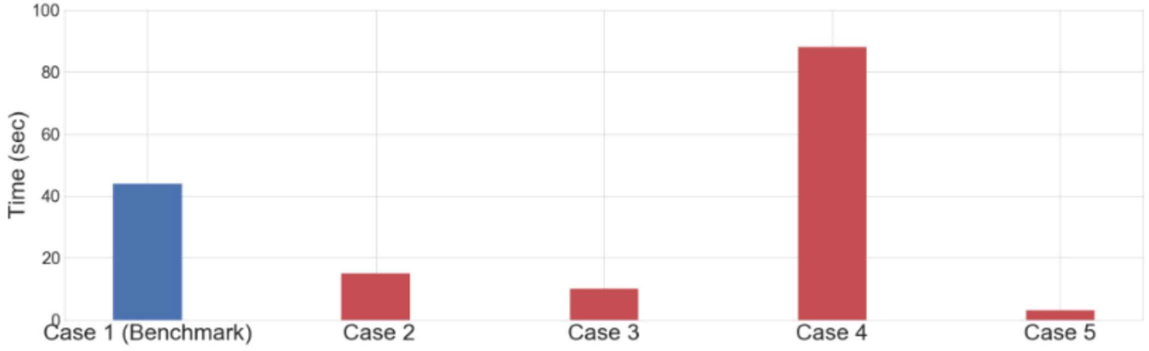
- Kubernetes allows very easy deployment, orchestration, scalability, resilience, replication, workloads and more
- Federation of services and Jobs
- From 0 to anomaly service → in minutes and config files
- Scale up/down (spark cluster and front-end) → Auto-scaling as an option
- Prototype support multiple users/projects, batch and streaming process
- Fault tolerant, disaster recovery

Spark Configuration Example

- Case 1: 800 trees, single core, serial mode
- Case 2: 100 trees on each core, aggregation and MapReduce. Each core access same data
- Case 3: Sample data on each core for 100 trees each, aggregation and MapReduce
- Case 4: Sample data on each core, 800 trees each
- Case 5: Sample data on each core, 100 trees on each, no aggregation

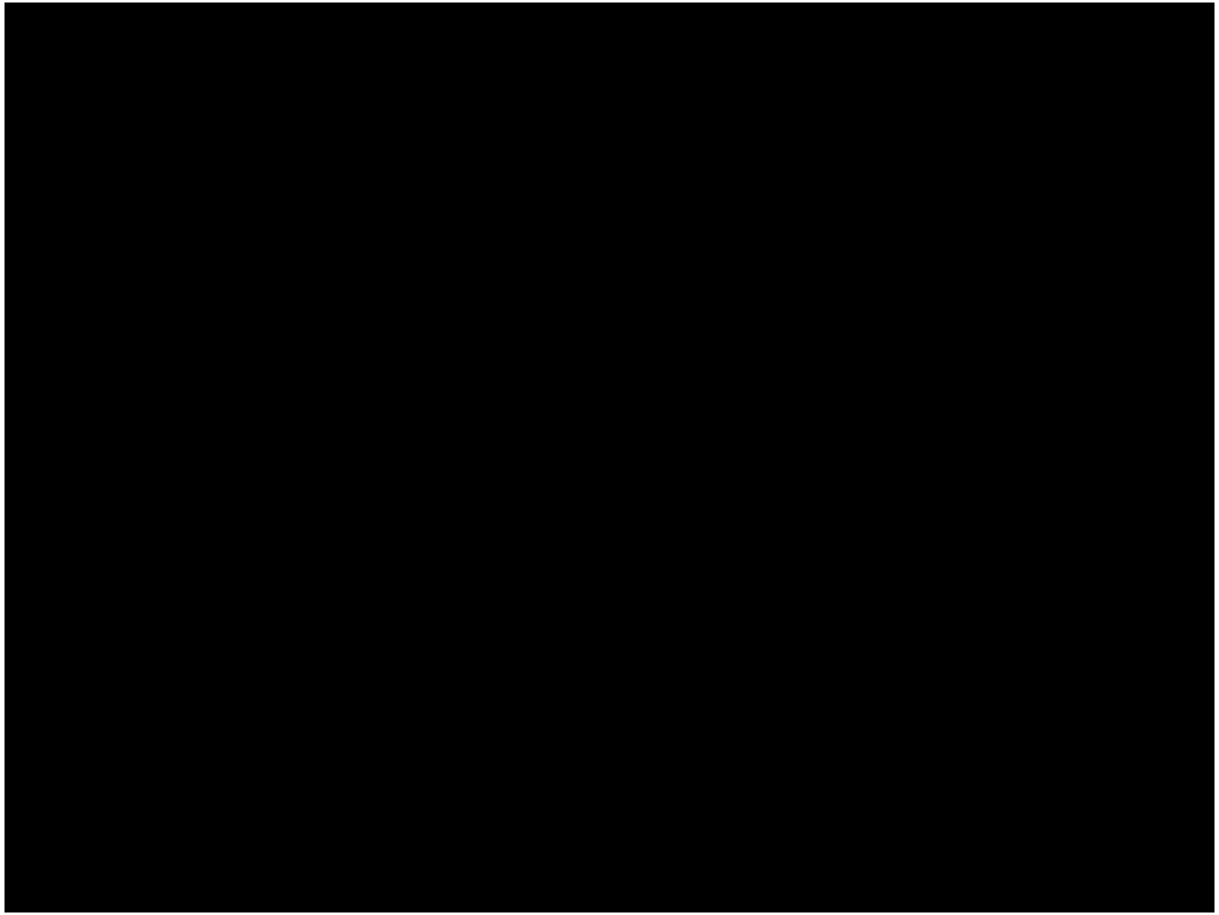


(a) Average difference computed as shown in equation 1

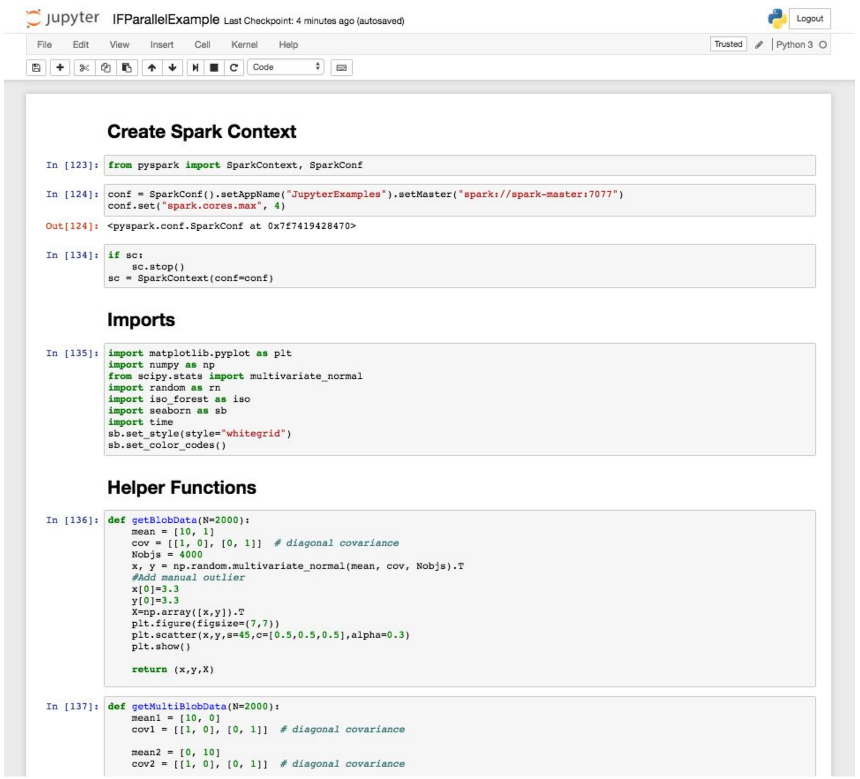


(b) Total average time taken to run each case.

Examples



Jupyter Notebooks



Create Spark Context

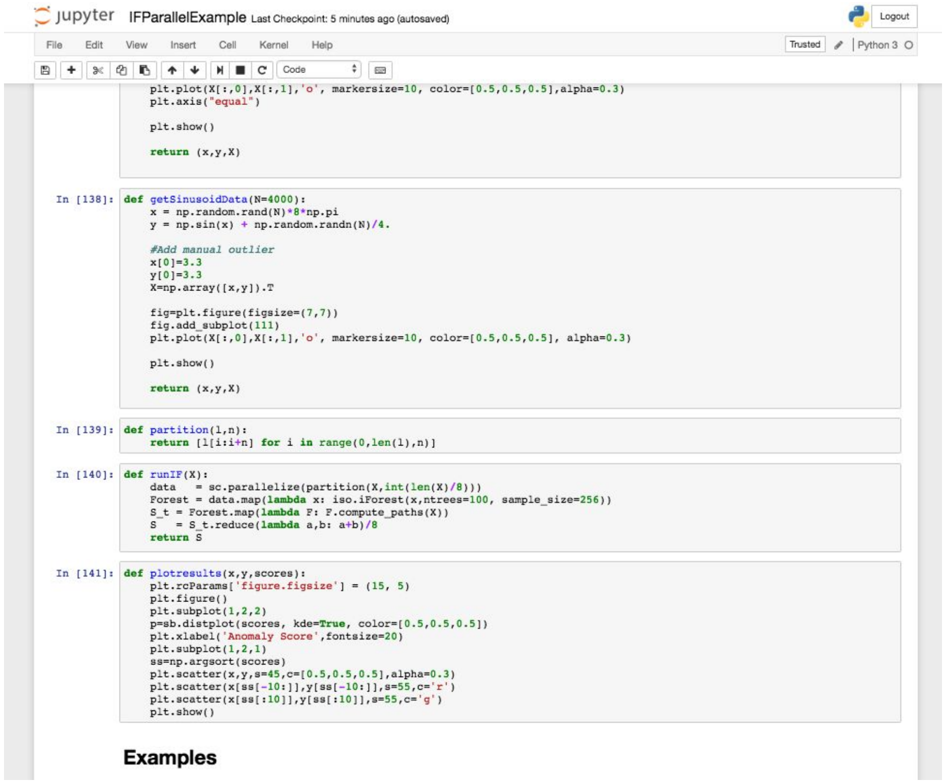
```
In [123]: from pyspark import SparkContext, SparkConf

In [124]: conf = SparkConf().setAppName("JupyterExamples").setMaster("spark://spark-master:7077")
          conf.set("spark.cores.max", 4)
Out[124]: <pyspark.conf.SparkConf at 0x7f7419428470>

In [134]: if sc:
              sc.stop()
          sc = SparkContext(conf=conf)
```

Imports

```
In [135]: import matplotlib.pyplot as plt
          import numpy as np
          from scipy.stats import multivariate_normal
          import random as rn
          import iso_forest as iso
          import seaborn as sb
          import time
          sb.set_style(style="whitegrid")
          sb.set_color_codes()
```

Helper Functions

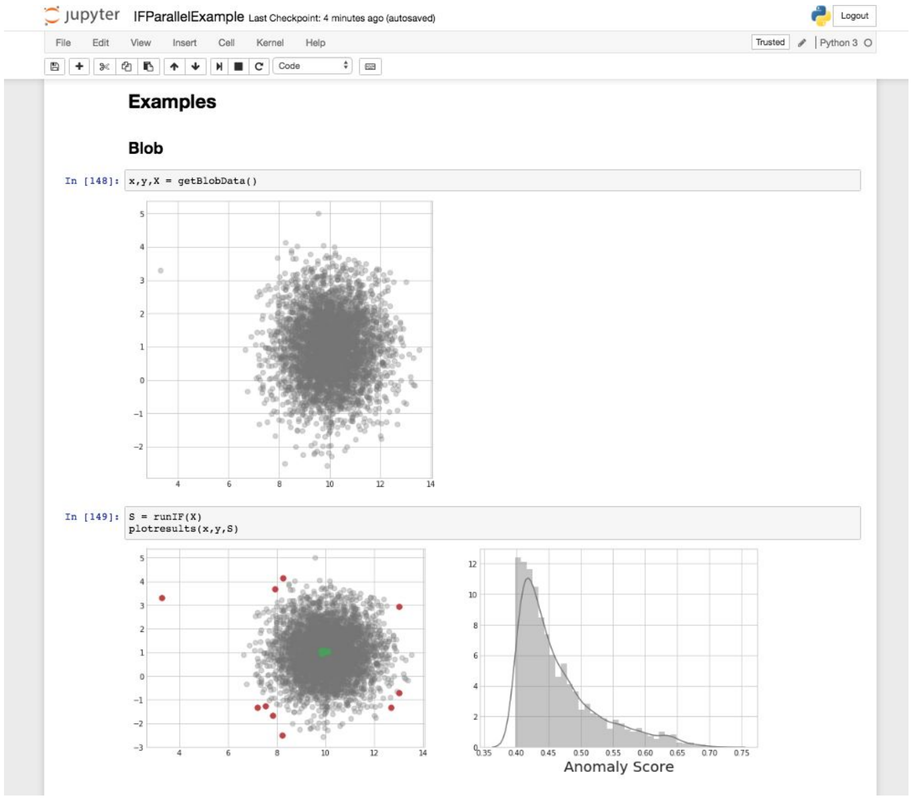
```
In [136]: def getBlobData(N=2000):
              mean = [10, 1]
              cov = [[1, 0], [0, 1]]  # diagonal covariance
              Nobjs = 4000
              x, y = np.random.multivariate_normal(mean, cov, Nobjs).T
              #Add manual outlier
              x[0]=3.3
              y[0]=3.3
              X=np.array([x,y]).T
              plt.figure(figsize=(7,7))
              plt.scatter(x,y,s=45,c=[0.5,0.5,0.5],alpha=0.3)
              plt.show()

              return (x,y,X)

In [137]: def getMultiBlobData(N=2000):
              mean1 = [10, 0]
              cov1 = [[1, 0], [0, 1]]  # diagonal covariance

              mean2 = [0, 10]
              cov2 = [[1, 0], [0, 1]]  # diagonal covariance
```



```
              plt.plot(X[:,0],X[:,1],'o', markersize=10, color=[0.5,0.5,0.5],alpha=0.3)
              plt.axis("equal")

              plt.show()

              return (x,y,X)

In [138]: def getSinusoidData(N=4000):
              x = np.random.rand(N)*8*np.pi
              y = np.sin(x) + np.random.randn(N)/4.

              #Add manual outlier
              x[0]=3.3
              y[0]=3.3
              X=np.array([x,y]).T

              fig=plt.figure(figsize=(7,7))
              fig.add_subplot(111)
              plt.plot(X[:,0],X[:,1],'o', markersize=10, color=[0.5,0.5,0.5], alpha=0.3)

              plt.show()

              return (x,y,X)

In [139]: def partition(l,n):
              return [l[i:i+n] for i in range(0,len(l),n)]

In [140]: def runIF(X):
              data   = sc.parallelize(partition(X,int(len(X)/8)))
              Forest = data.map(lambda x: iso.iForest(x,ntrees=100, sample_size=256))
              S_t = Forest.map(lambda F: F.compute_paths(X))
              S   = S_t.reduce(lambda a,b: a+b)/8
              return S

In [141]: def plotresults(x,y,scores):
              plt.rcParams['figure.figsize'] = (15, 5)
              plt.figure()
              plt.subplot(1,2,2)
              p=sb.distplot(scores, kde=True, color=[0.5,0.5,0.5])
              plt.xlabel('Anomaly Score',fontsize=20)
              plt.subplot(1,2,1)
              ss=np.argsort(scores)
              plt.scatter(x,y,s=45,c=[0.5,0.5,0.5],alpha=0.3)
              plt.scatter(x[ss[-10:]],y[ss[-10:]],s=55,c='r')
              plt.scatter(x[ss[:10]],y[ss[:10]],s=55,c='g')
              plt.show()
```

Examples

Jupyter Notebooks

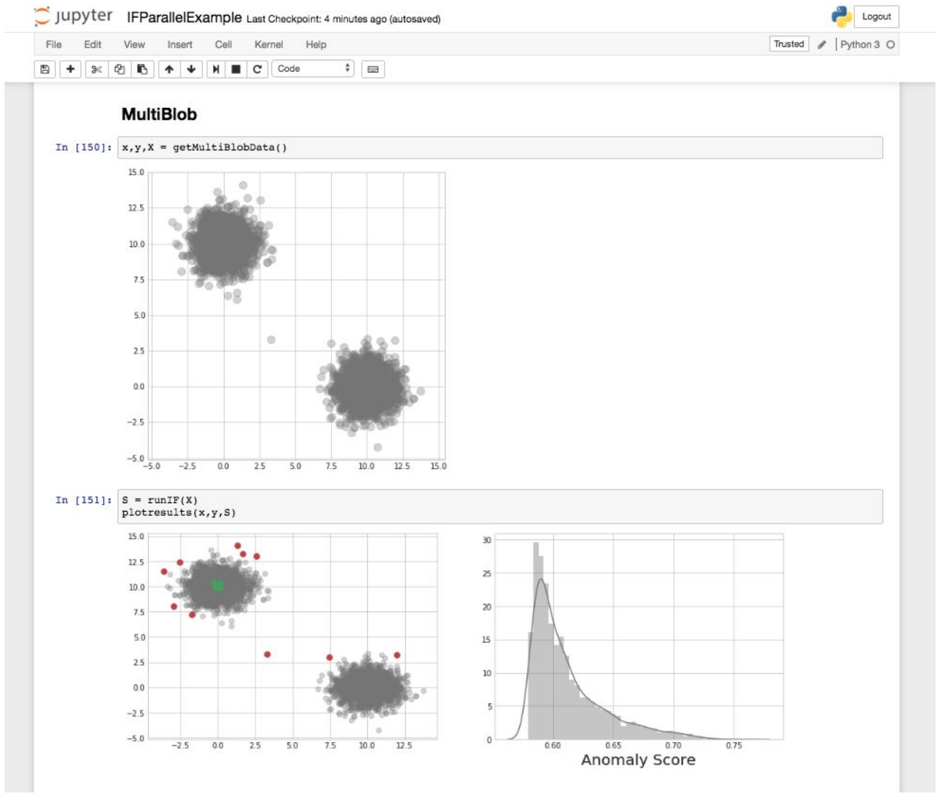


Examples

Blob

```
In [148]: x,y,X = getBlobData()
```

```
In [149]: S = runIF(X)
          plotresults(x,y,S)
```



MultiBlob

```
In [150]: x,y,X = getMultiBlobData()
```

```
In [151]: S = runIF(X)
          plotresults(x,y,S)
```

Conclusions

- Open source anomaly detection software package for scientific application using fast and efficient isolation forest
- Fault tolerant, robust, scalable deployment
- Train and scoring using Spark
- Ready-to-deploy infrastructure on Kubernetes
- Production services for large datasets

Thank you!

Questions?

Matias Carrasco Kind -- NCSA
mcarras2@illinois.edu
github.com/mgkind
matias-ck.com

Extra Slides

Streaming

- 2 cases: Time evolving data, Time accumulative data
- Streaming isolation forest exists, not extended
- We can adapt and retrain trees as new data is presented
- Replace trees one by one until whole forest is replaced
- Work with window size to retrain trees