ICDCN 2018

7TH INTERNATIONAL WORKSHOP ON COMPUTING AND NETWORKING FOR IOT AND BEYOND

Distributed Synthetic Minority Oversampling Technique

Avnish Kumar Rastogi, Nitin Narang, Mohammad Ajmal

Agenda

1. Paper Overview, What and Why of the Problem — 5 Minutes
2. Algorithm Overview and Implementation Approach — 10 Minutes
3. Algorithm Evaluation and Results — 5 Minutes

Image Detection

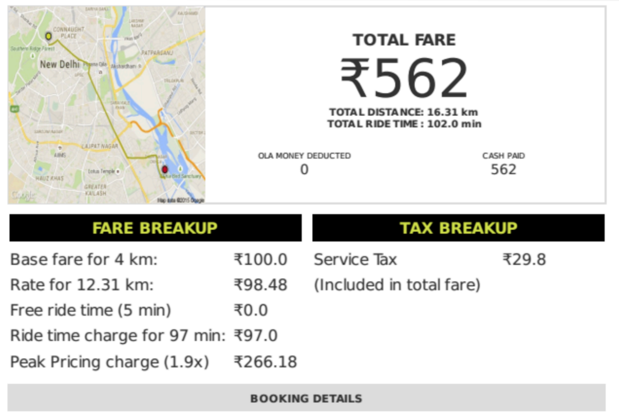
Fraud Detection

Voice and Speech Processing

Context Based Intelligence

E-Commerce

Product Launches


TOTAL FARE
₹562
TOTAL DISTANCE: 16.31 km
TOTAL RIDE TIME : 102.0 min

OLA MONEY DEDUCTED	CASH PAID
0	562

FARE BREAKUP		TAX BREAKUP	
Base fare for 4 km:	₹100.0	Service Tax	₹29.8
Rate for 12.31 km:	₹98.48	(Included in total fare)	
Free ride time (5 min)	₹0.0		
Ride time charge for 97 min:	₹97.0		
Peak Pricing charge (1.9x)	₹266.18		

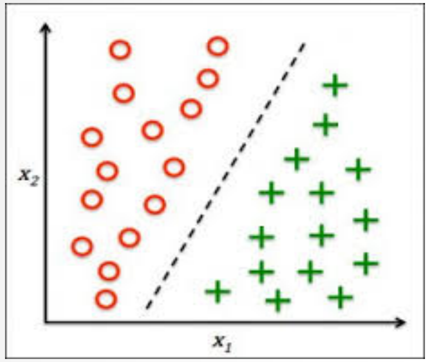
BOOKING DETAILS

Unbalanced
Datasets

Problem - Predictive Analytics – Highly Unbalanced data

Supervised Learning from Imbalanced Data Sets

- *18 real-valued features in a dataset of over 3.4 billion records with majority vs. minority distribution of 98:2*



ML Algorithms using standard classifiers are overwhelmed by the majority class and ignore the minority. But we are interested in Minority identification ☹

- *Very High Accuracy by predicting all as majority class*
- *Poor identification of minority class…. Outliers, ….*

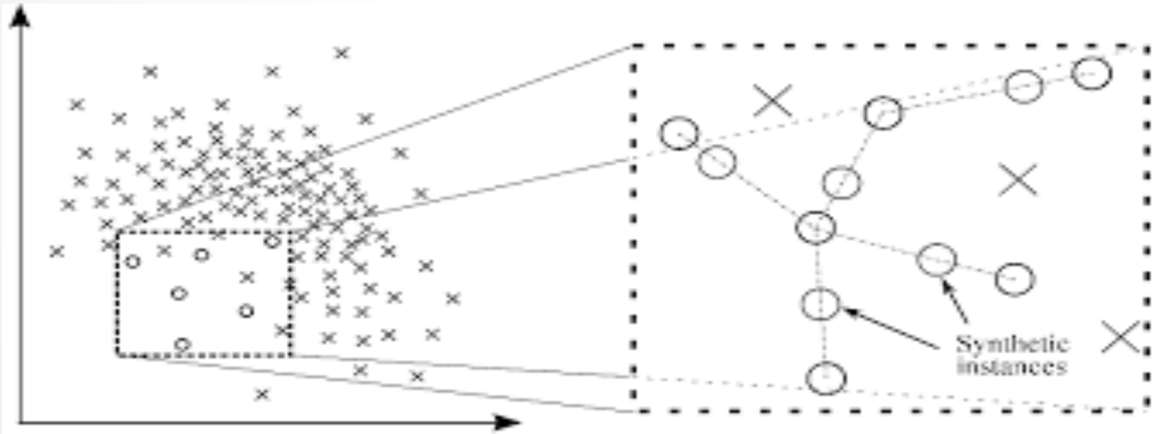
Solution to Class Imbalance

- ***Under-Sampling*** *- extract a smaller set of majority instances while preserving all the minority instances*
- *Stratified sampling*
- ***Over sampling*** *- increases the number of minority instances by over-sampling*
  - ❖ *Over sampling by duplication*
  - ❖ *SMOTE – Synthetic Minority Over Sampling of Minority (*Normal, Borderline, Borderline-2 and SVM)

Over-Sampling but How?

- *Decision Tree Classifiers, random under-sampling and over-sampling with SMOTE significantly improve accuracy.*
- *Neural Network classifier with over-sampling with SMOTE gives the best accuracy among all re-sampling techniques.*

➢Identify neighbors of a minority sample
➢For each of the neighbor, generate random point near the sample



Chawla, Nitesh An insight into imbalanced Big Data classification: outcomes and challenges, March 2017 Complex Intelligent Systems pp 105-120

Small Data Set

Key Challenge

- *Implementation in python – single machine*
- *On-Going Research for distributed implementation – Map-Reduce.*

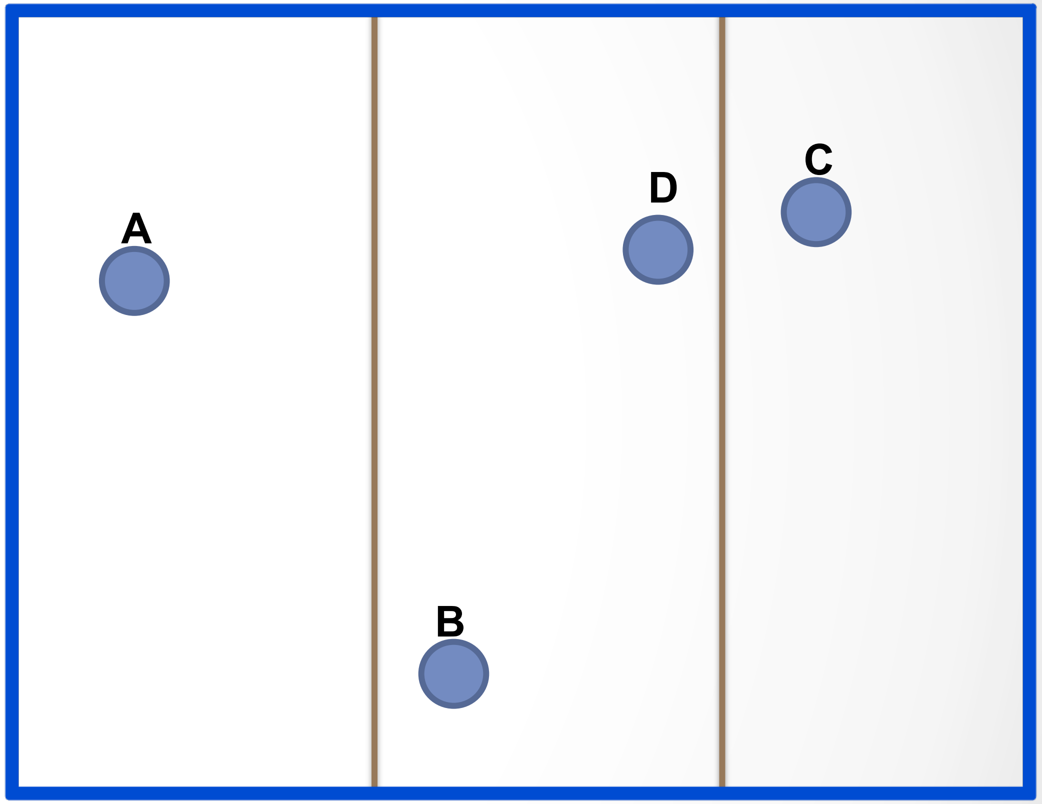
SMOTE – Basic Algorithm

Proposed by Nitesh Chawla in 2001

```
while N ≠ 0
        Choose a random number between 1 and k, call it nn. This step chooses one of
        the k nearest neighbors of i.
        for attr ← 1 to numattrs
                Compute: dif = Sample[nnarray[nn]][attr] − Sample[i][attr]
                Compute: gap = random number between 0 and 1
                Synthetic[newindex][attr] = Sample[i][attr] + gap ∗ dif
        endfor
        newindex++
        N = N − 1
endwhile
return (∗ End of Populate. ∗)
End of Pseudo-Code.
```

Difficulties with Big Data

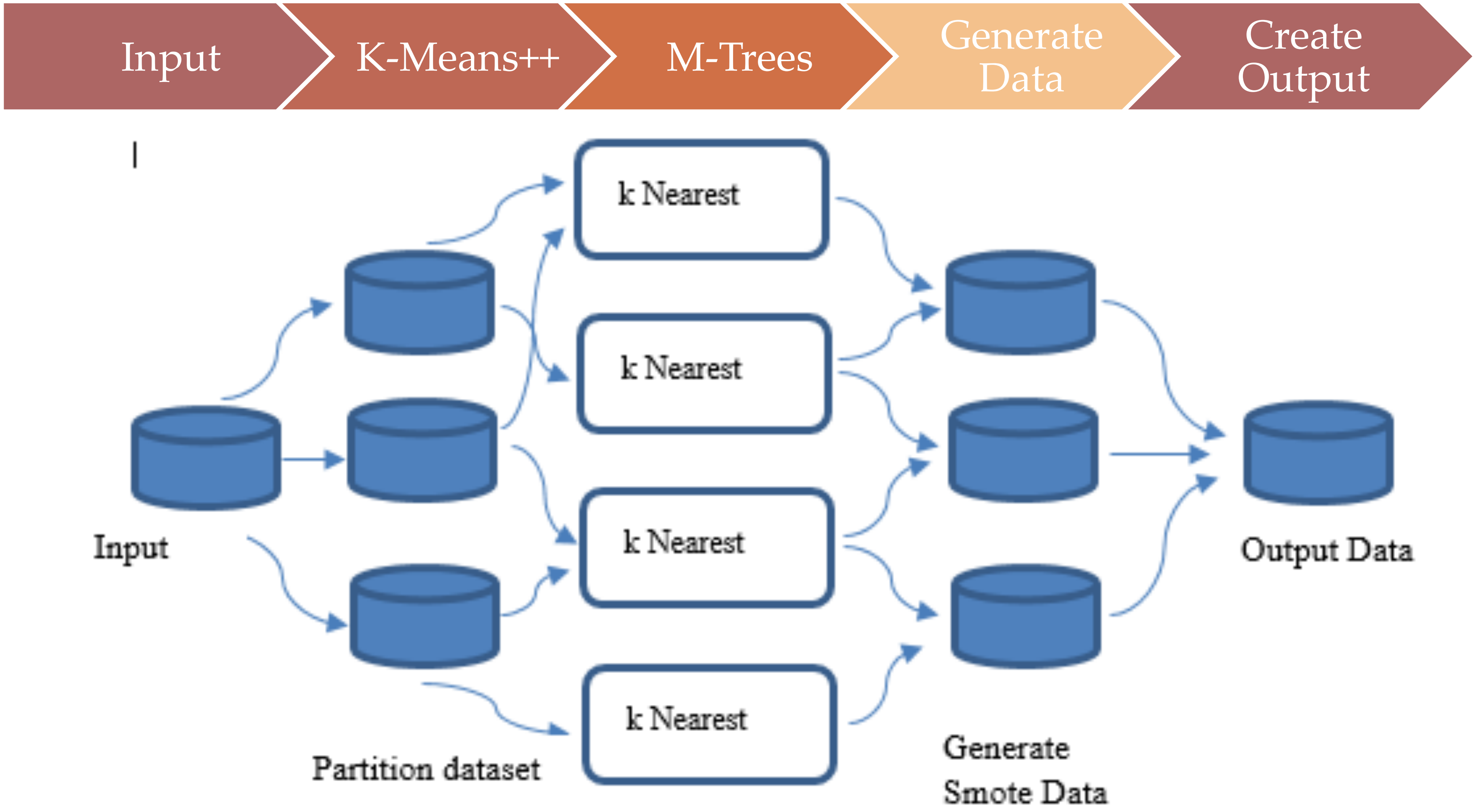
- Data is huge
- Does not fit single machine
- Divide it between different nodes
- Destroys distribution of data and small drifts



Technical Problem

- Cluster the points
- Find k Nearest Neighbors for each sample
- Up sample by generating points randomly between - sample & neighbor

Distributed SMOTE High Level Design



Clustering

- We used parallel K-means++ algorithm for clustering the points in "N" buckets

- Algorithm proposed by Bahman Bahmani and others in "**Scalable KMeans++**"

M-Trees

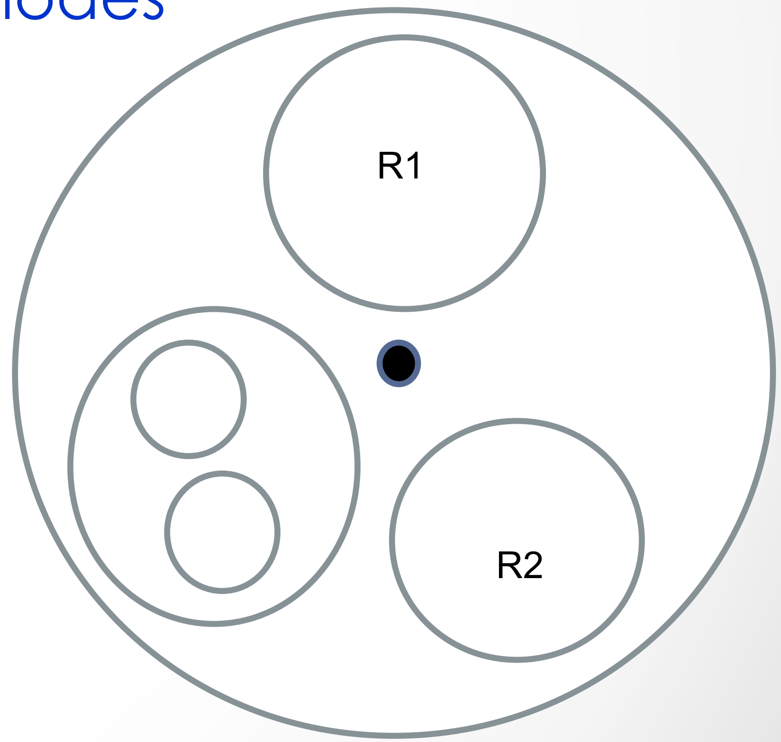
Proposed by P.Ciaccia, M.Patella, F.Rabitti, P.Zezula in their research published as **Indexing Metric Spaces with M-tree**

M Tree indexes a metric space where, the distance function "d" satisfies:

- $d(O_x, O_y) = d(O_y, O_x)$
- $d(O_x, O_y) > 0$ if $O_x \neq O_y$ and $d(O_x, O_x) = 0$
- $d(O_x, O_y) \leq d(O_x, O_z) + d(O_z, O_y)$ – Triangle Inequality

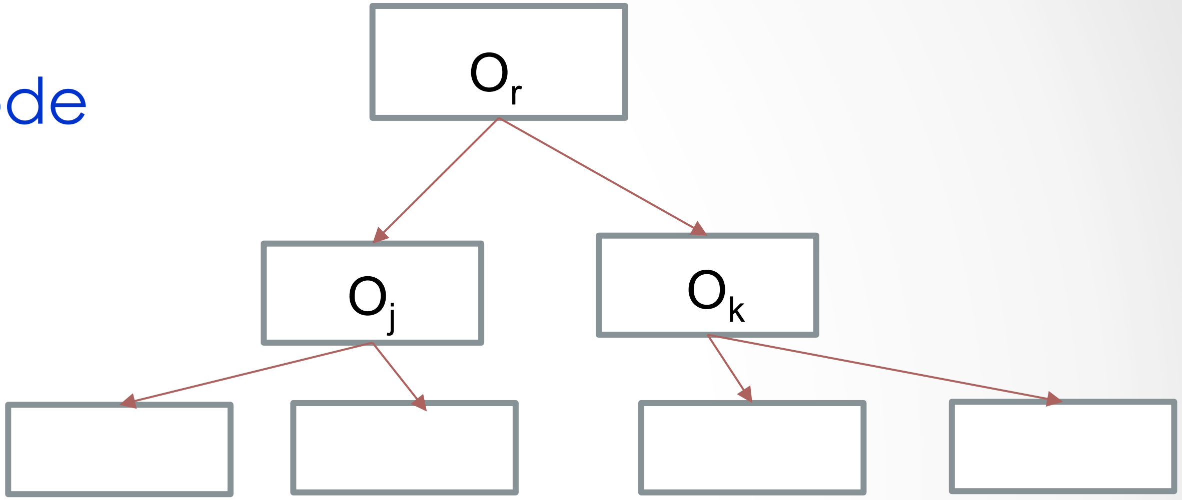
M-Tree – What is it?

- M-Tree partitions objects on the basis of their relative distance
- Fixed sized nodes, called the capacity of the nodes
- Leaf Nodes – Data nodes



M-Tree …

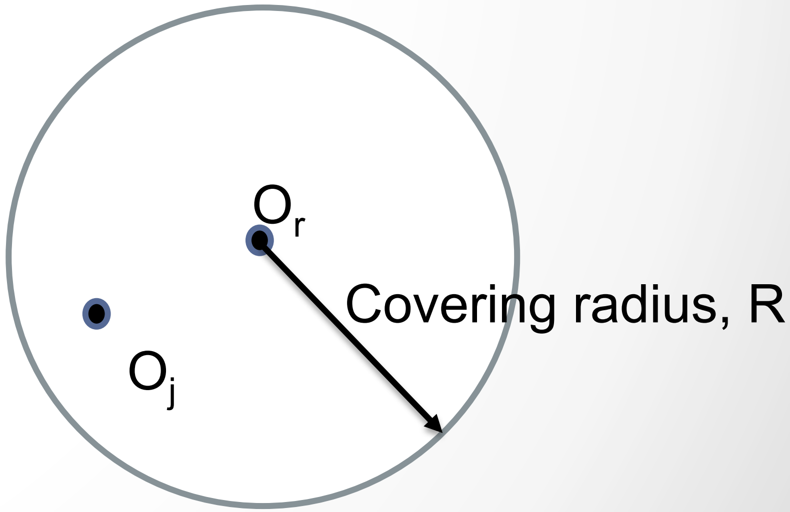
- Routing Node - internal (non-leaf) node
  - Pointer to the sub tree
  - Radius of the tree
  - Distance with the parent



- Covering Node – Nodes that contain data node

- Covering Radius -> d $(O_j, O_r) \leq r(O_r)$

  Maximum distance of all the points from the router stored/contained within the router



Build M-Tree

- Mark the first point as router
  - For all the other points, calculate the distance from the router
  - Add them as leaf Nodes to the router.
  - Update the radius of the router
  - If num of nodes >= capacity

    select two routers from the group of nodes

    split the nodes into two groups

Splitting Policy

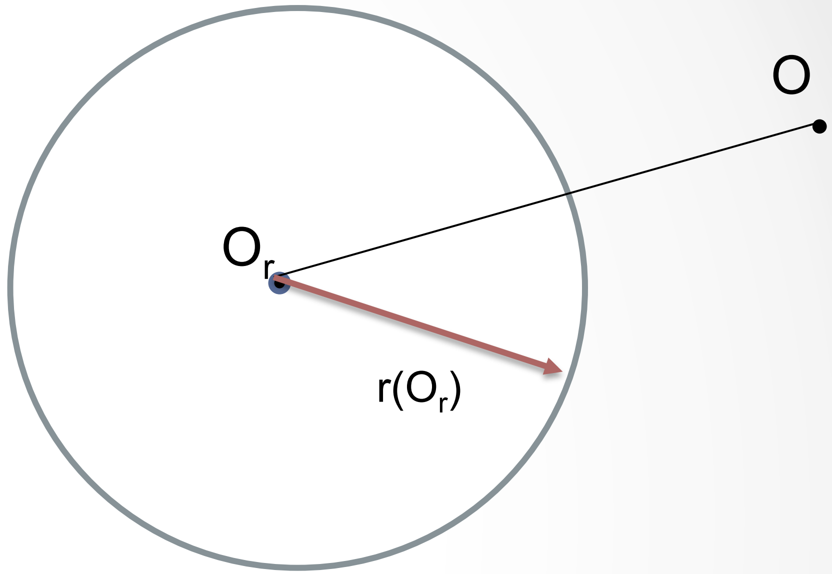
Max-Min Policy

- Choose the point with maximum distance from router as R_1
- Divide the group using "Generalized Hyperplane Approach"
  Assign object $O_j \in N$ to the nearest routing object
  if $d(O_j, Op_1) \leq d(O_j, Op_2)$ then assign O_j to N_1, else assign O_j to N_2.

Search K Neighbours M-Tree

Go to individual routers

$$dminT(Or) = \max \{d(Or - O) - r(Or), 0\}$$



For all these selected routing nodes, we select data nodes where

$$d(Op, O) \leq d(Op, Oj) + d\ (Oj, Q)$$

$$\Rightarrow d\ (Oj, O) \geq d(Op\text{-}O) - d\ (Op, Oj)$$

$\Rightarrow \ | \ d(Op, Q) - d(Oj, Op) \ | \leq dk$ where dK is the farthest distance of the nearest neighbour

Results

- Infrastructure and Datasets
- Validate Accuracy of generated data
- Ability to process large Datasets

Datasets

Datasets Used

Datasets	# Rows	# Attributes	Class (maj:min)	%Class
ECBDL 14	2.89 million	631	2849275: 48637	98.3% : 1.7%
KEEL abalone 19	4174	8	4142:32	99.23% : 0.77%
KEEL yeast4	1484	8	1433:51	96.56% : 3.44%
UCI SatImage	6435	36	5809:626	90.27% : 9.73%

Model Parameters

Model	Parameters
Distributed Random Forest (h2o)	Number of Tress : 50 Maximum Tree Depth 20 NBins: 20 Sample Rat : 0.63
Default Random Forest	Number of trees 10

Cluster Configuration

• Number of Machines/Nodes	4
• Machine	Centos 6.6 Linux
• Cores	8
• RAM	20 GB
• Spark/Hadoop	Distributed Framework

Results (Abalone Dataset/Yeast4)

Technique	AUC	Recall	GM	Confusion Matrix
Python SMOTE	0.78	0.60	0.76	3 2 26 804
Spark SMOTE	0.85	0.80	0.84	4 1 89 741

Technique	AUC	Recall	GM	Confusion Matrix
Python SMOTE	0.90	0.85	0.90	6 1 14 276
Spark SMOTE	0.92	0.85	0.91	5 2 11 279

Results (UCI/ECBDL)

Technique	AUC	Recall	GM	Confusion Matrix
Python SMOTE	0.78	0.75	0.76	160	51 113	1676
Spark SMOTE	0.78	0.83	0.87	175	36 169	1620

Technique	AUC	Recall	GM	Confusion Matrix
Python SMOTE	0.79	0.78	0.78	11442	3129 184402	670401
Spark SMOTE	0.89	0.81	0.81	11746	2823 166637	688166

Conclusions

- Python SMOTE being monolithic has challenges with scale on large datasets
- Our implementation of SMOTE gives comparable results (quality of synthetic minority data generation) to existing SMOTE implementation (in python/R)
- Further work needs to be done to extend this algorithm to Borderline, Borderline-2 and SVM versions of SMOTE

Questions ?

Thank You