Exact & Approximate SMOTE Algorithms for Handling Big Data in PySpark

Tom Cotter Hewen Pang Ting-Chun Chen

Brownbag Objectives

1

What is SMOTE?

How does it work?

2

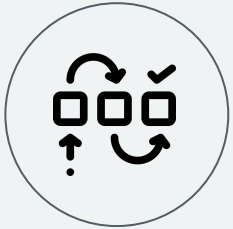
Why use SMOTE?

How does it help solve problems?

3

Our solution

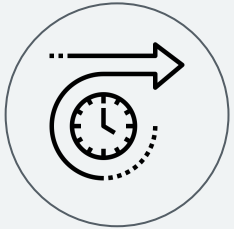
Fast, scalable big data SMOTE algorithms.



4

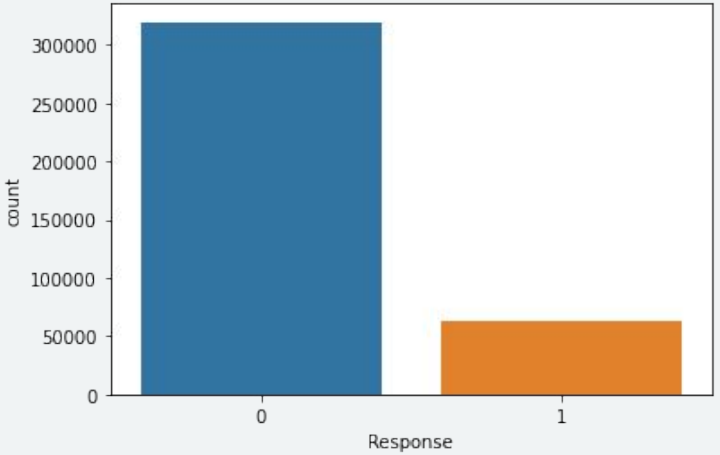
Future Scope

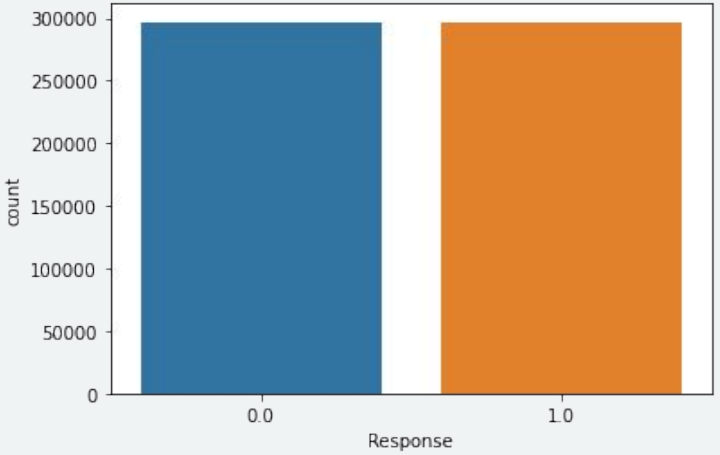
Potential improvements, ideas, ...



What is SMOTE?

- An Oversampling Algorithm
- Used in imbalanced datasets.
- Creates new synthetic data rather than using replacement

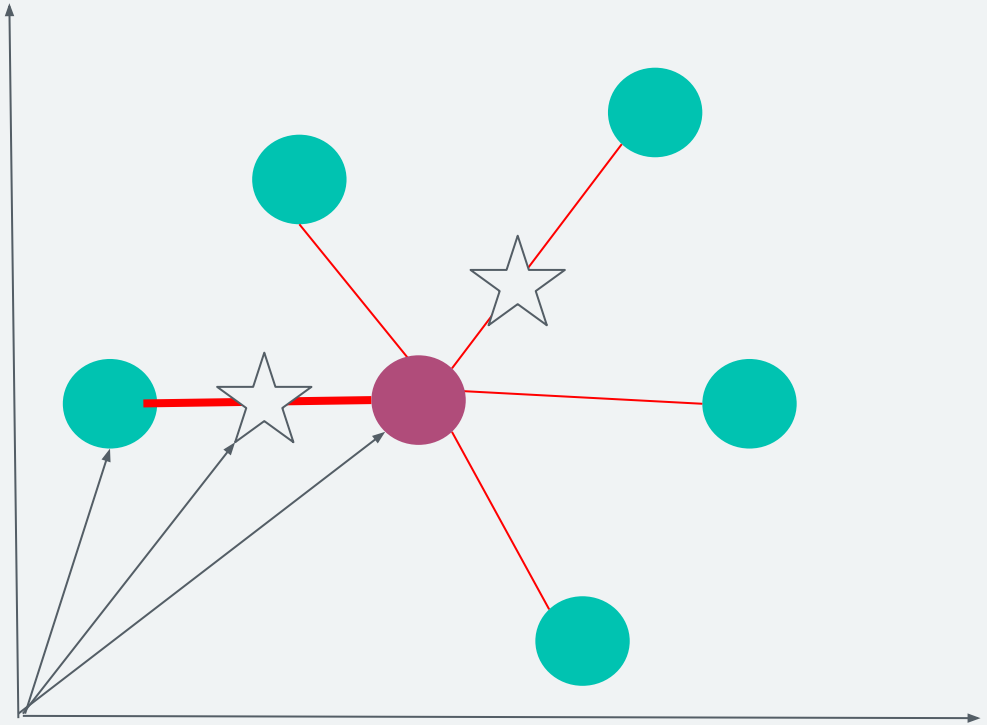




- Useful in datasets such as fraud detection / cancer detection.
- Since it uses NearestNeighbors, it's not inherently suited to Big Data.

How does it work?

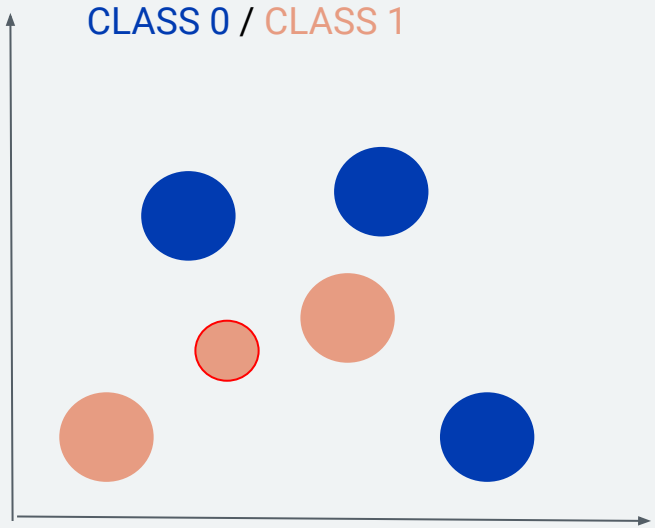
Creates synthetic points by randomly sampling the feature space between every point and a random choice of its k nearest neighbors



Variants of SMOTE

We can apply SMOTE by itself, or combine it with other algorithms such as Edited Nearest Neighbors (ENN) to help remove noise.

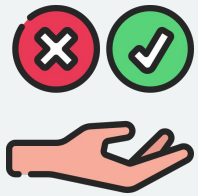
ENN removes samples from the dataset that are misclassified by their nearest neighbors



What have we implemented?

Local Solution

Applies the imblearn implementation of SMOTE to partitions of the dataset.



- Optimal implementation from python library
- Doesn't take into account entire dataset when looking at neighbors

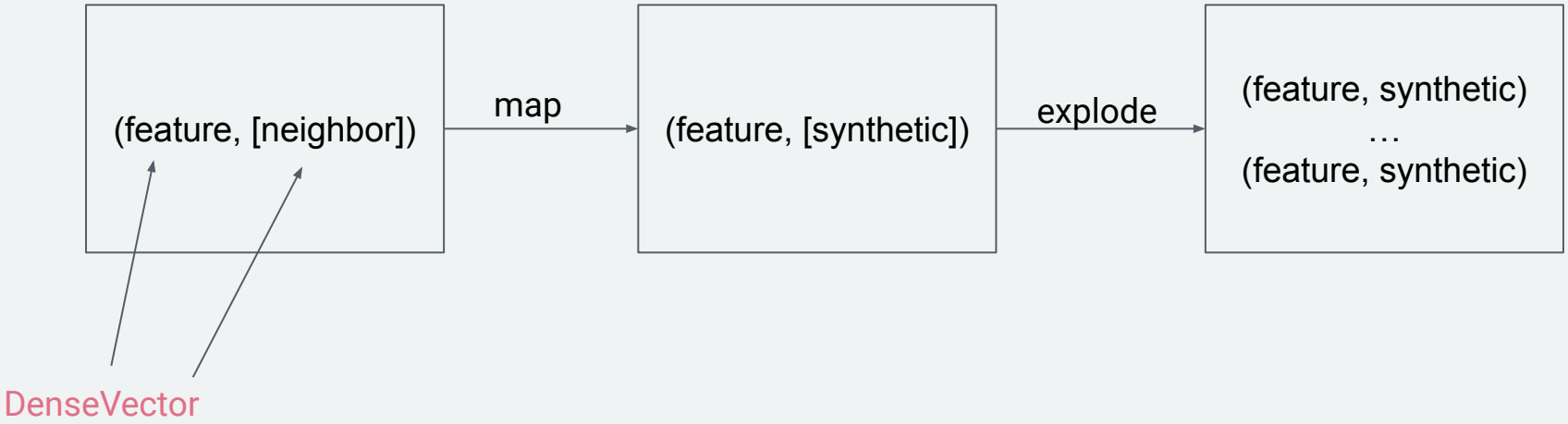
Global Solutions

Approx-SMOTE

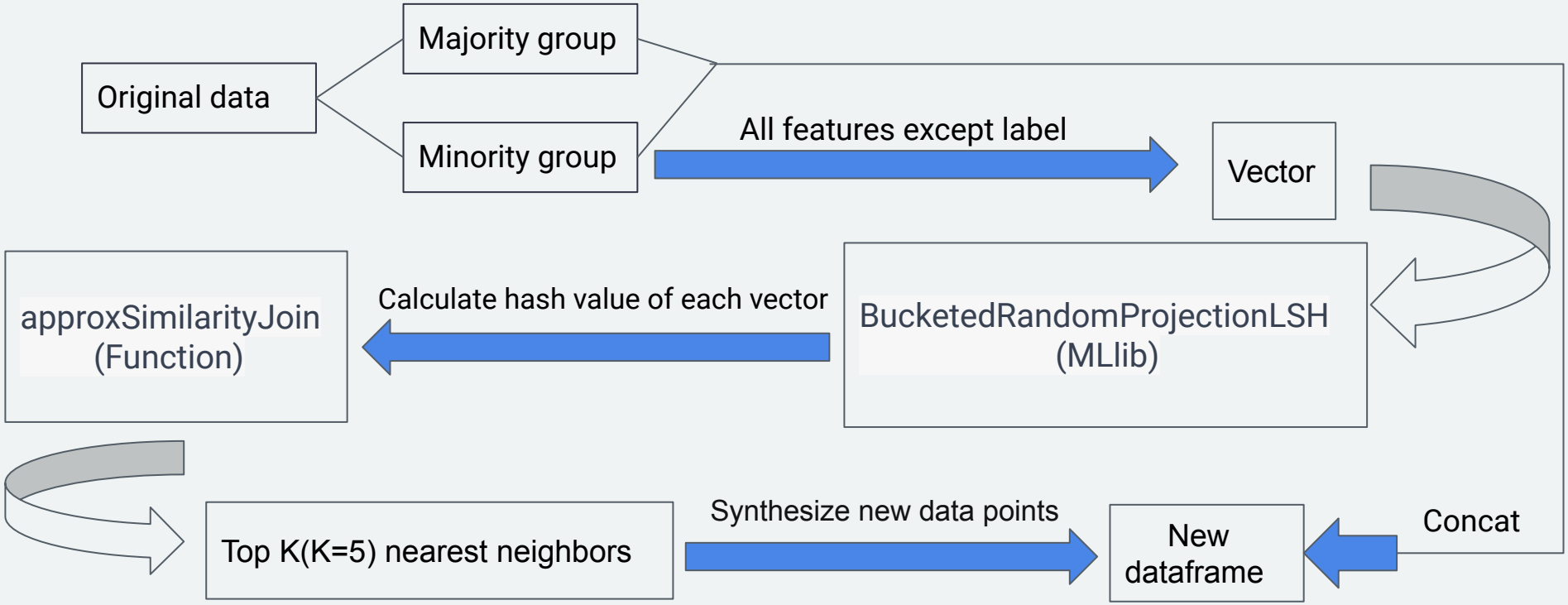
Exact-SMOTE
+
Exact-ENN

Synthesising the new data

We perform a map step to synthesize the new data.

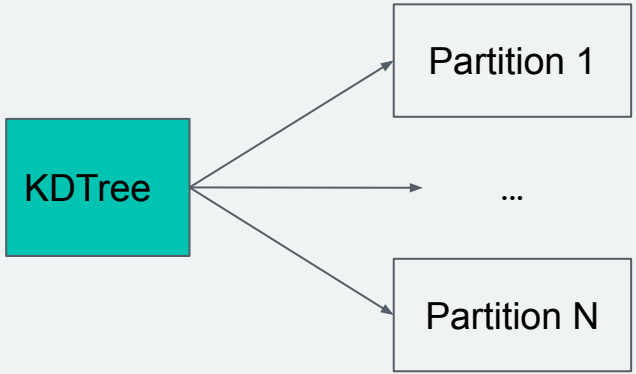


Approximate SMOTE



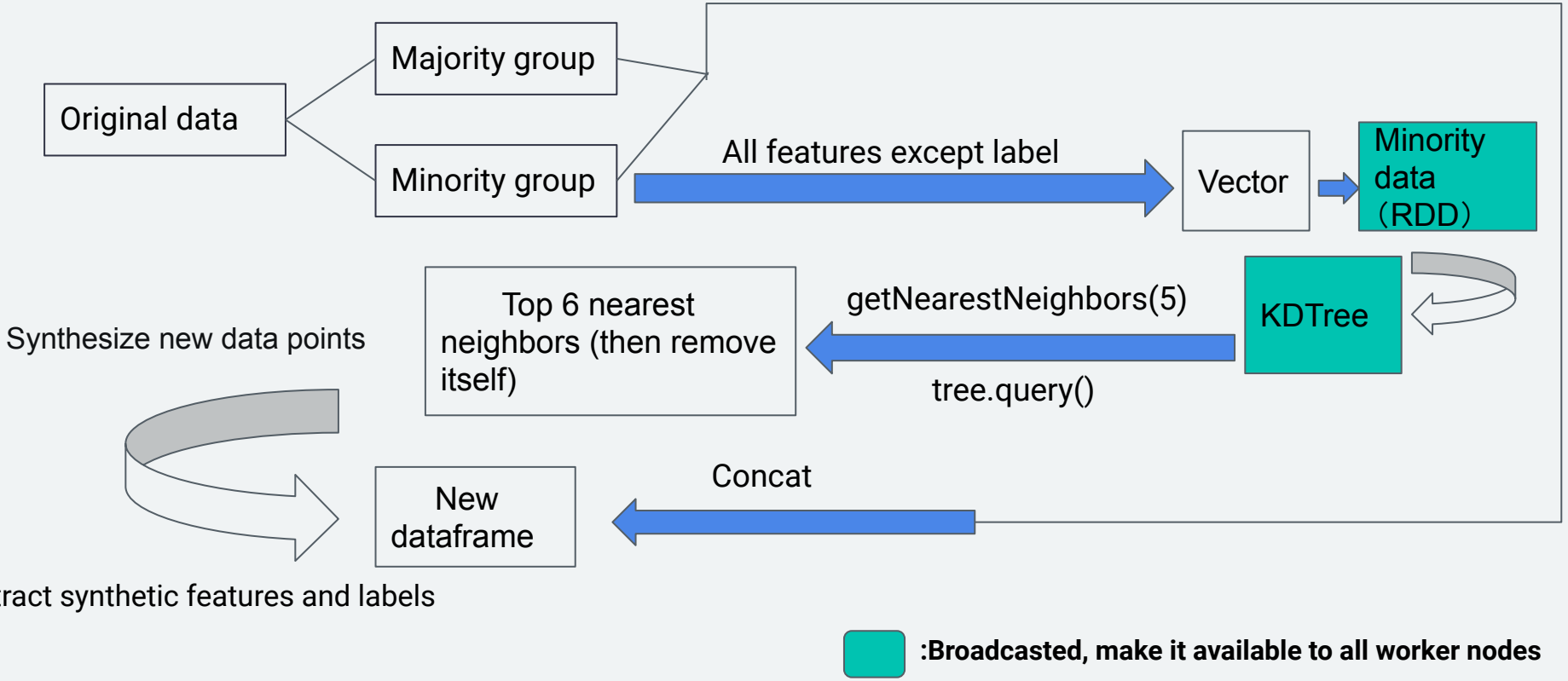
KDTree

- Faster than simply comparing each point.
- Works by recursively splitting the feature space in smaller and smaller regions.
- When querying, we start at the root node and descend the tree, always choosing the subtree closest to our point.Means we only have to search half of the possible points.
- Can be constructed in O(k log n) time.

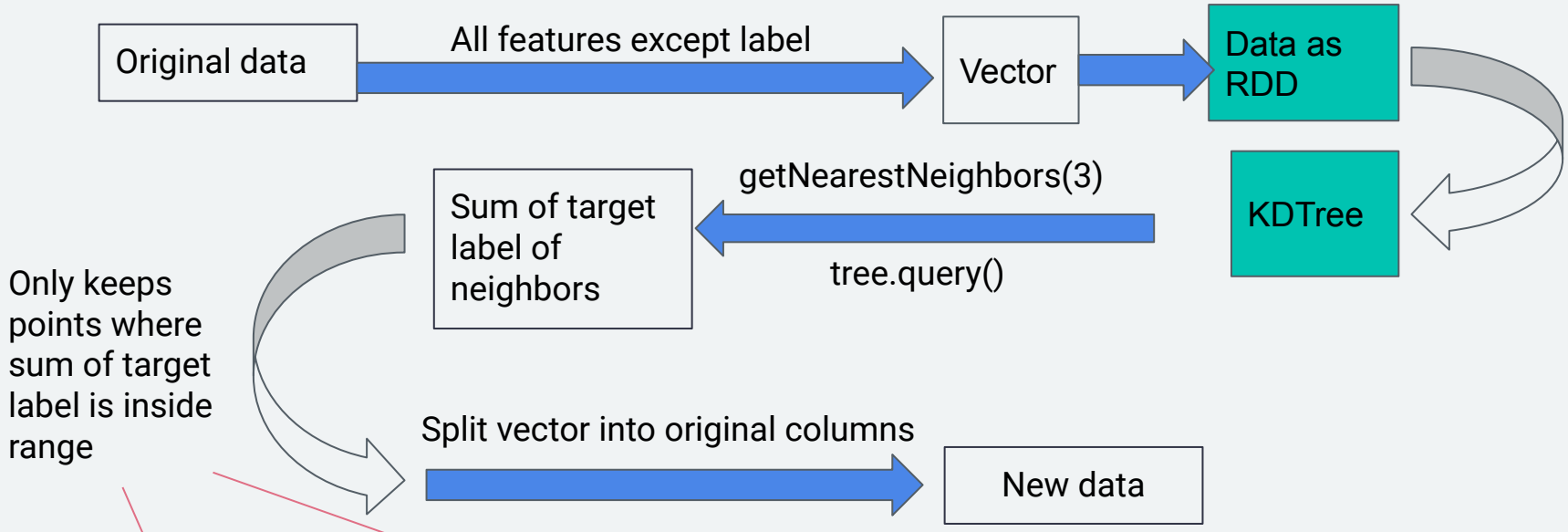


https://www.youtube.com/watch?v=BK5x7IUTIyU

Exact SMOTE


Original data
Majority group
Minority group
All features except label
Vector
Minority data (RDD)
KDTree
getNearestNeighbors(5)
tree.query()
Top 6 nearest neighbors (then remove itself)
Synthesize new data points
Concat
New dataframe
Extract synthetic features and labels
:Broadcasted, make it available to all worker nodes

Exact-ENN



When removing noise from minority group (label = 1), we want the sum of target labels of the neighbors to be greater than or equal to 2

When removing noise from majority group (label = 0), we want the sum of target labels to be less than or equal to 1

Dataset

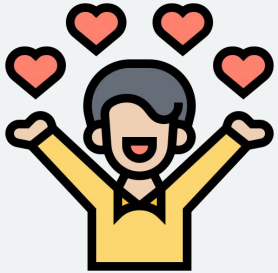
Two datasets:

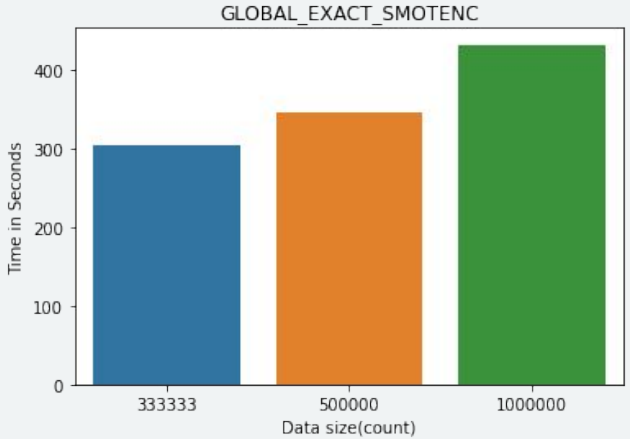
- 380,000 records with a 4.9:1 imbalance ratio
- 123,000,000 records with 50:1 imbalance ratio
    - We used 1,000,000 and 5,000,000 randomly selected records from this dataset.

Results

5,000,000 records sequentially

```
org.apache.spark.SparkException: Job aborted
due to stage failure: Total size of serialized
results of 36 tasks (4.0 GiB) is bigger than
spark.driver.maxResultSize 4.0 GiB.
```

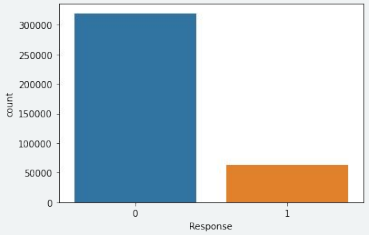


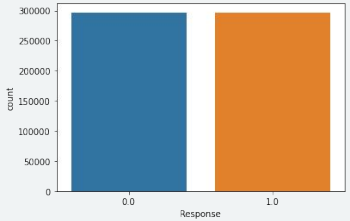


5,000,000 took 310 seconds.

More detailed results

- The original dataset has 380 thousand samples with only 16% of positive response.
- We produced a scalable version of the exact SMOTE algorithm for use in pySpark.
- We are able to improve the recall and precision rate of the minority class when training the oversampled dataset on a Decision Tree





Metrics	Baseline	Exact-SMOTE
Balance Score	0.64	1.00
Precision (for minority)	0.00	0.36
Recall (for minority)	0.00	0.07
F1-score	0.76	0.77
AUC	0.50	0.83

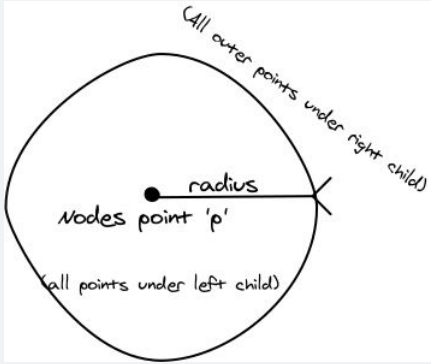
Future Scope

Improved efficiency / synthesis

- Use broadcast effectively
- Cluster-SMOTE, SVM-SMOTE, Gaussian-SMOTE, …
- ADASYN (Adaptive Synthetic)

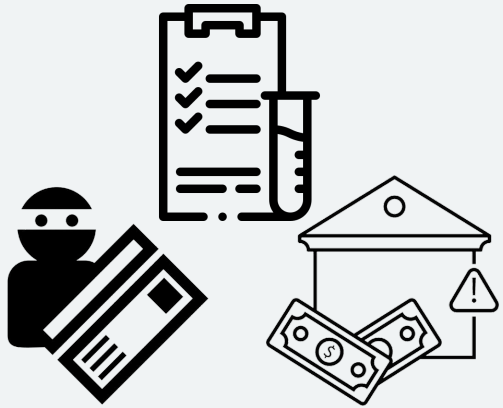
Vantage Point Tree (VP Tree)

- Possibly Faster than KDtree
- Define our own distance function



Application

- Credit Card Fraud
- Medical Tests
- Loan Default Prediction



Thanks!

Do you have any questions?

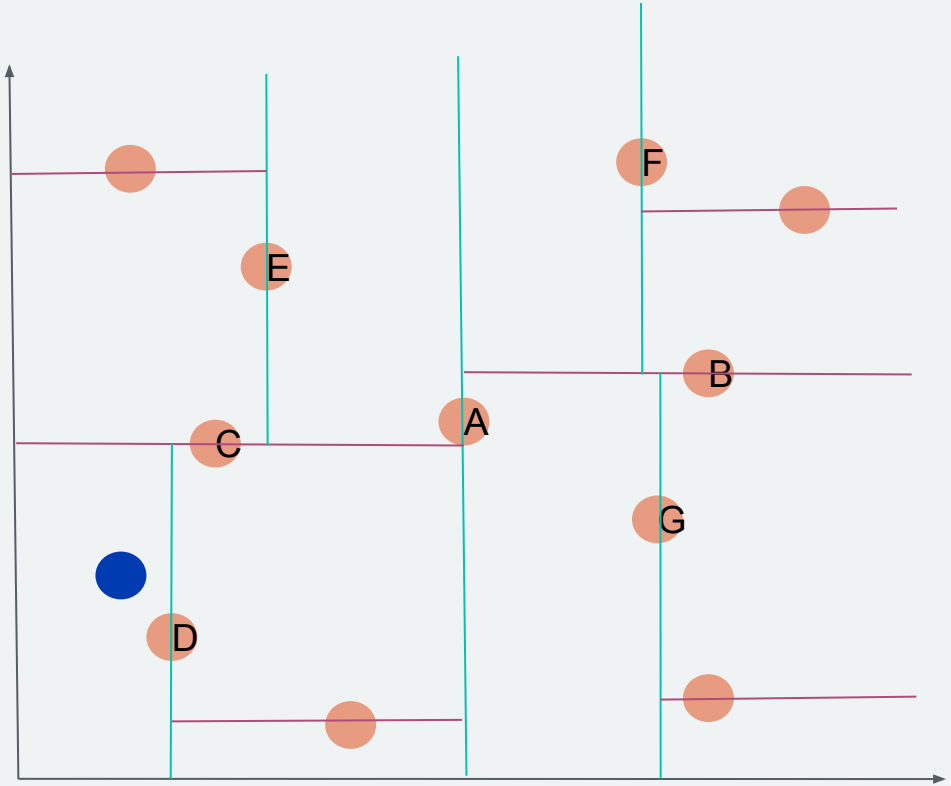
Shannon Entropy

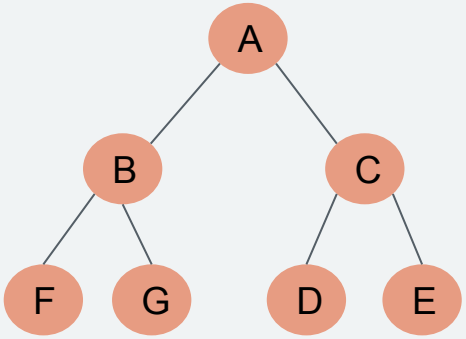
Shannon Entropy is a measure of the amount of uncertainty or information content in a probability distribution. On a dataset of ***n*** instances, if you have ***k*** classes of sizes $\boldsymbol{c_i}$, Shannon Entropy can be computed as equation (1). This will tend towards 0 when the dataset is very imbalanced, and will tend towards ***log k*** when the classes are balanced. Therefore, we can calculate a balanced score as equation (2), which will tend towards 1 for a balanced dataset.

(1) $H = \Sigma_{i=1}^{k} (c_i/n) \log (c_i/n)$

(2) $\text{Balanced} = H / \log k$

KDTrees





BucketedRandomProjectionLSH

https://spark.apache.org/docs/2.1.0/ml-features.html#locality-sensitive-hashing

Hash Function ->

$$h^{x,b}(\vec{v}) = \lfloor \frac{\vec{v} \cdot \vec{x} + b}{w} \rfloor$$

V - feature vector
X - random noise

Values are then grouped by similar hash values. It's easier to compare hash values than vectors.

More detailed results

- A larger dataset (50:1)

Metrics	Baseline	Exact-SMOTE
Balance Score	0.13	0.48
Precision (for minority)	0.944	1.00
Recall (for minority)	0.999	1.00
F1-score	0.998	0.99
AUC	0.999	1.00