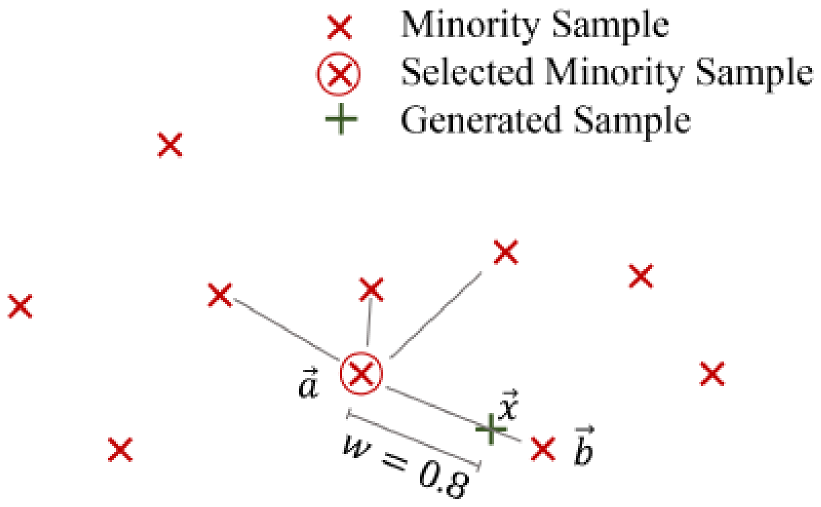
Train In Data

K-Means SMOTE

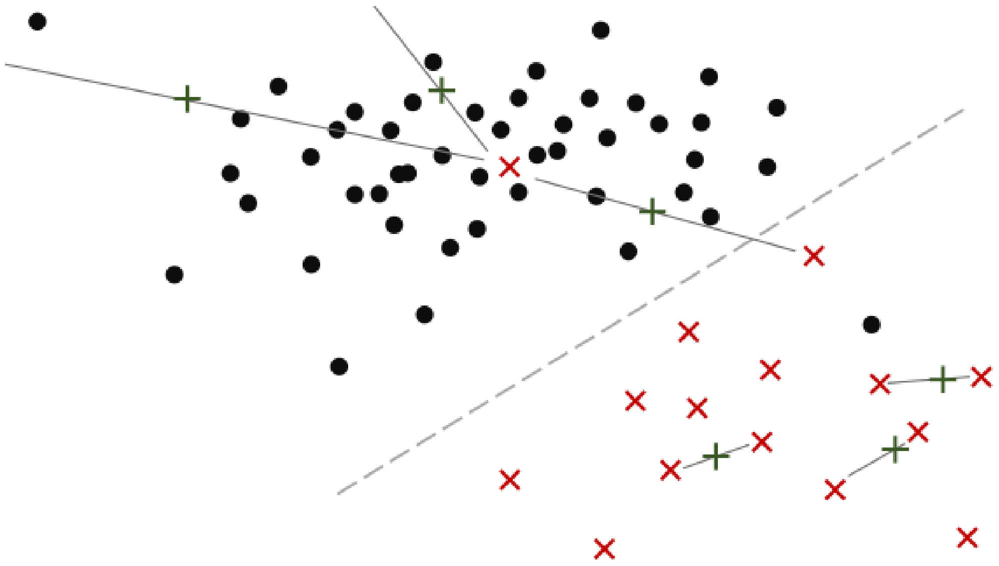
A bit of background: SMOTE



https://arxiv.org/pdf/1711.00837.pdf

SMOTE linearly interpolates a randomly selected minority sample and one of its k = 5 nearest neighbors

SVM, Borderline SMOTE

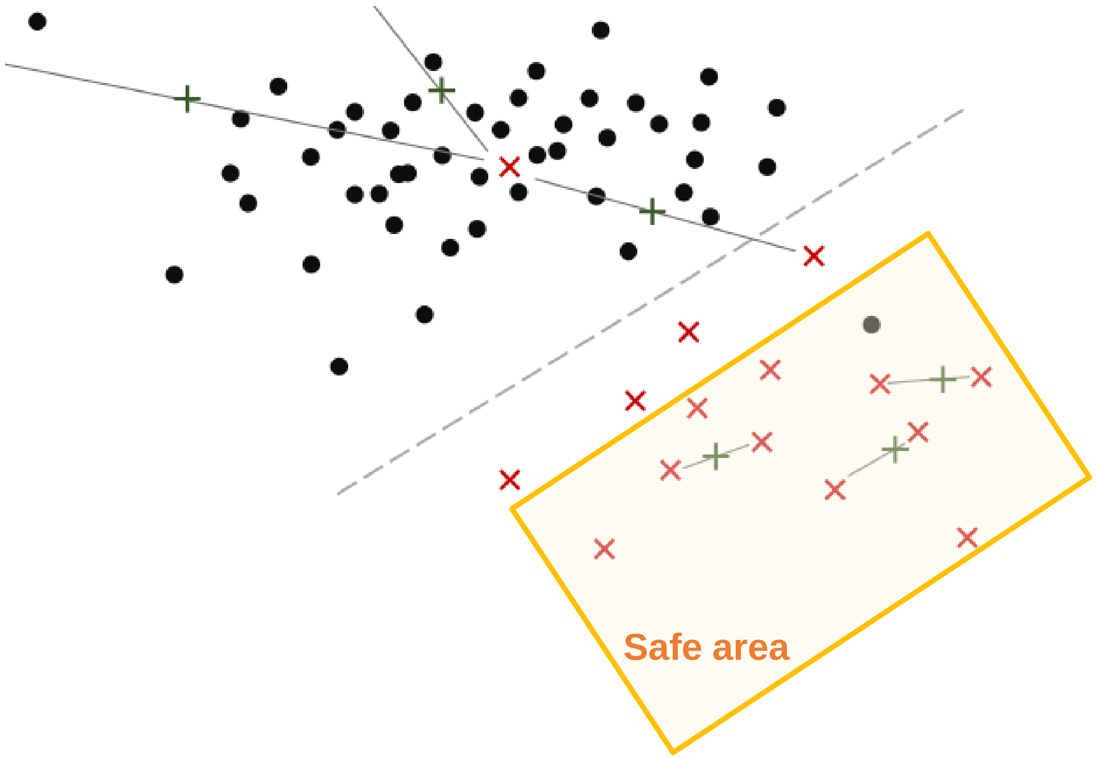


https://arxiv.org/pdf/1711.00837.pdf

- We should not create samples in areas that are safe

- We should create samples at the boundary

SVM, Borderline SMOTE

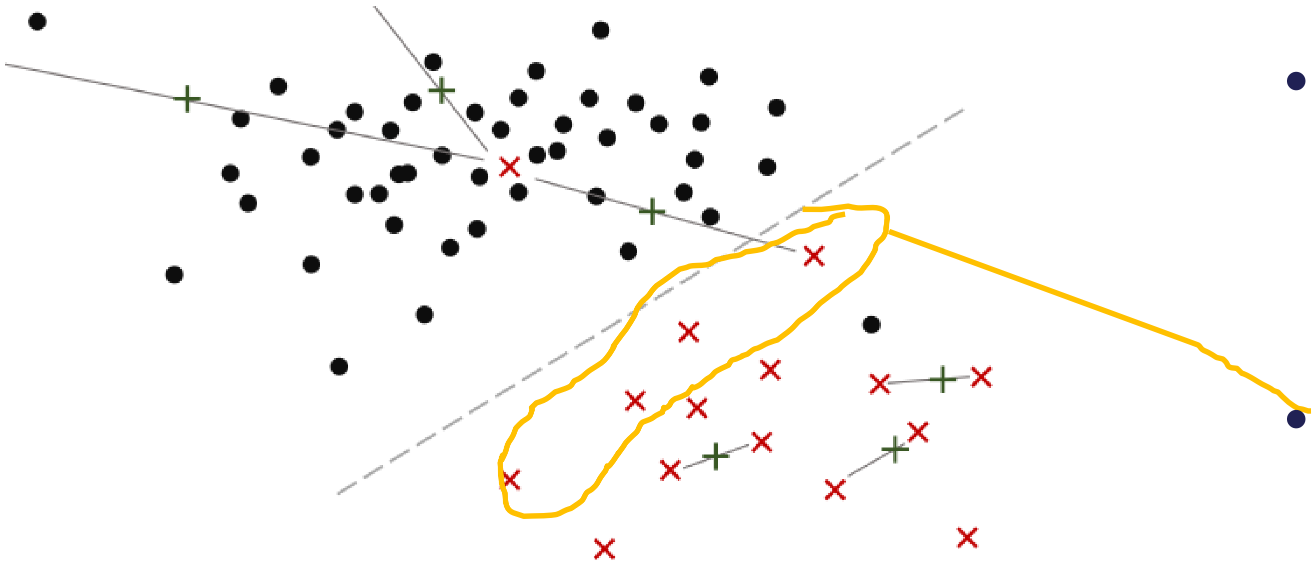


https://arxiv.org/pdf/1711.00837.pdf

- We should not create samples in areas that are safe

- We should create samples at the boundary

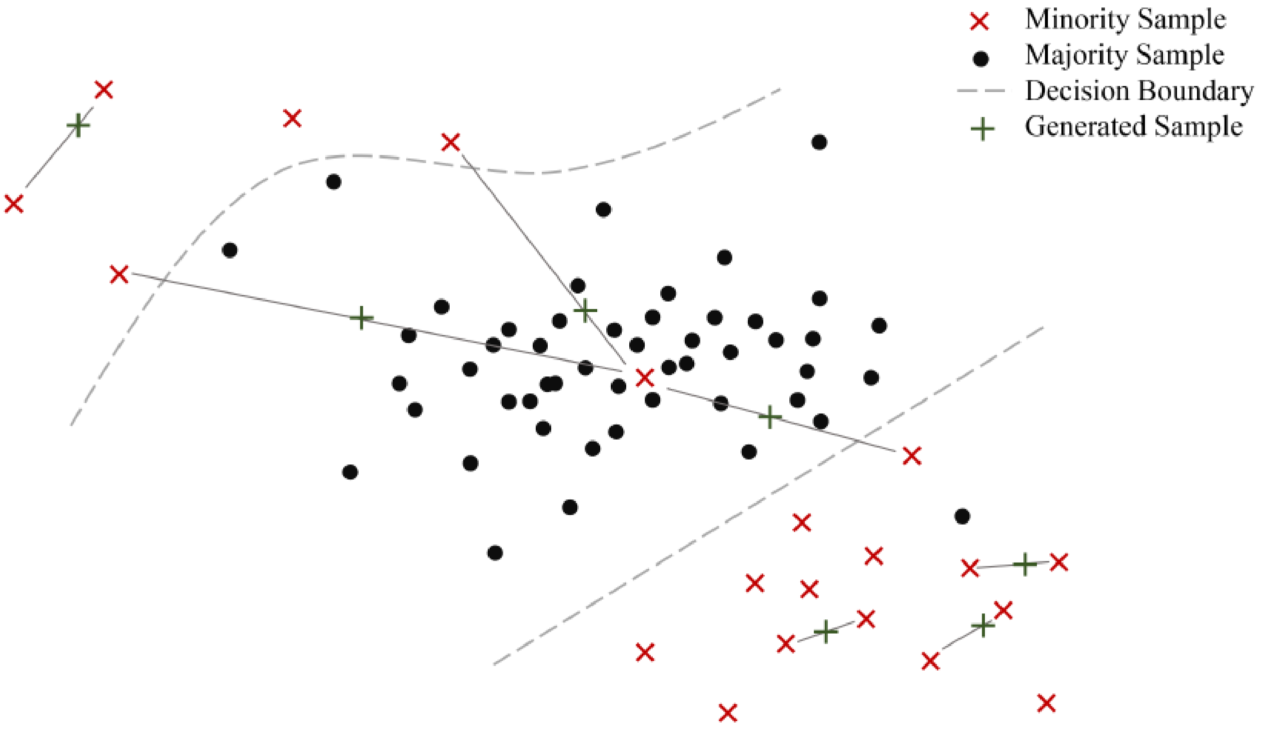
SVM, Borderline SMOTE



https://arxiv.org/pdf/1711.00837.pdf

- We should not create samples in areas that are safe
- We should create samples at the boundary

SMOTE, noise and intra-class clusters

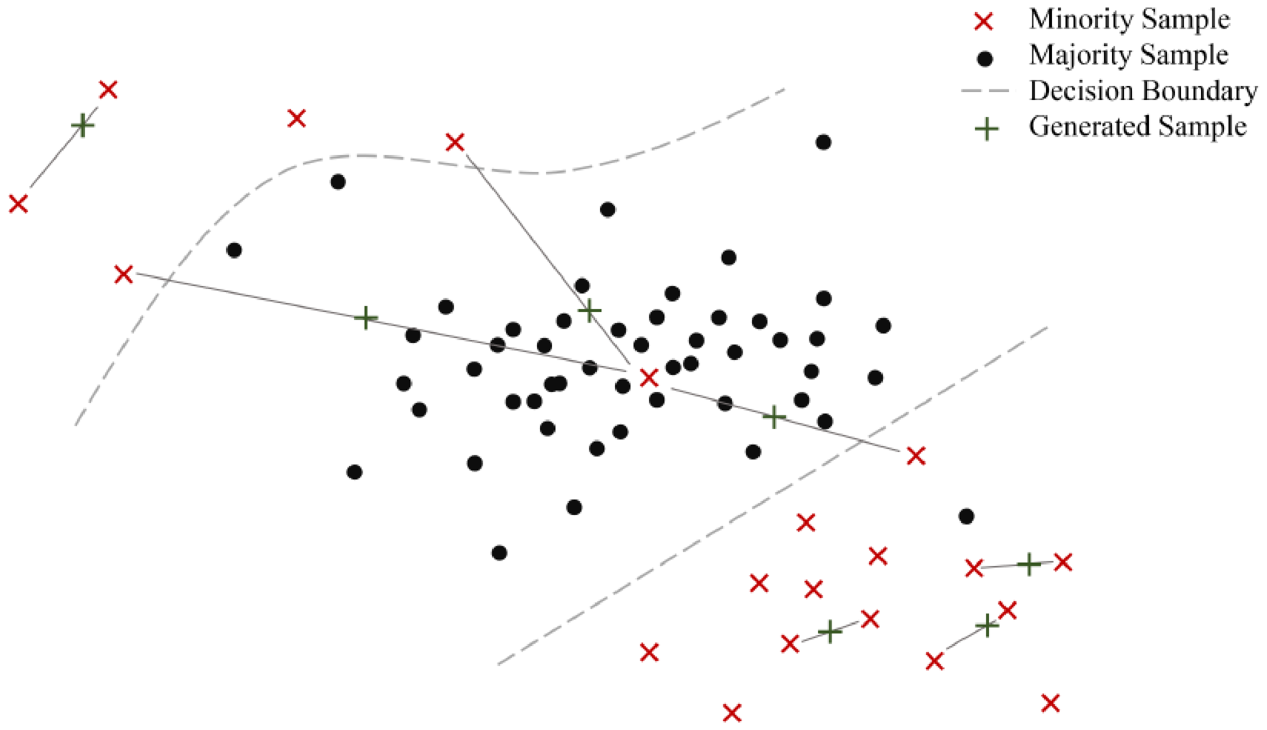


https://arxiv.org/pdf/1711.00837.pdf

- SMOTE might create noisy examples

- SMOTE does not contemplate intra-class clusters

Examples of intra-class clusters



https://arxiv.org/pdf/1711.00837.pdf

Fraudulent credit card transactions:

- Transactions in foreign countries
- High value transactions

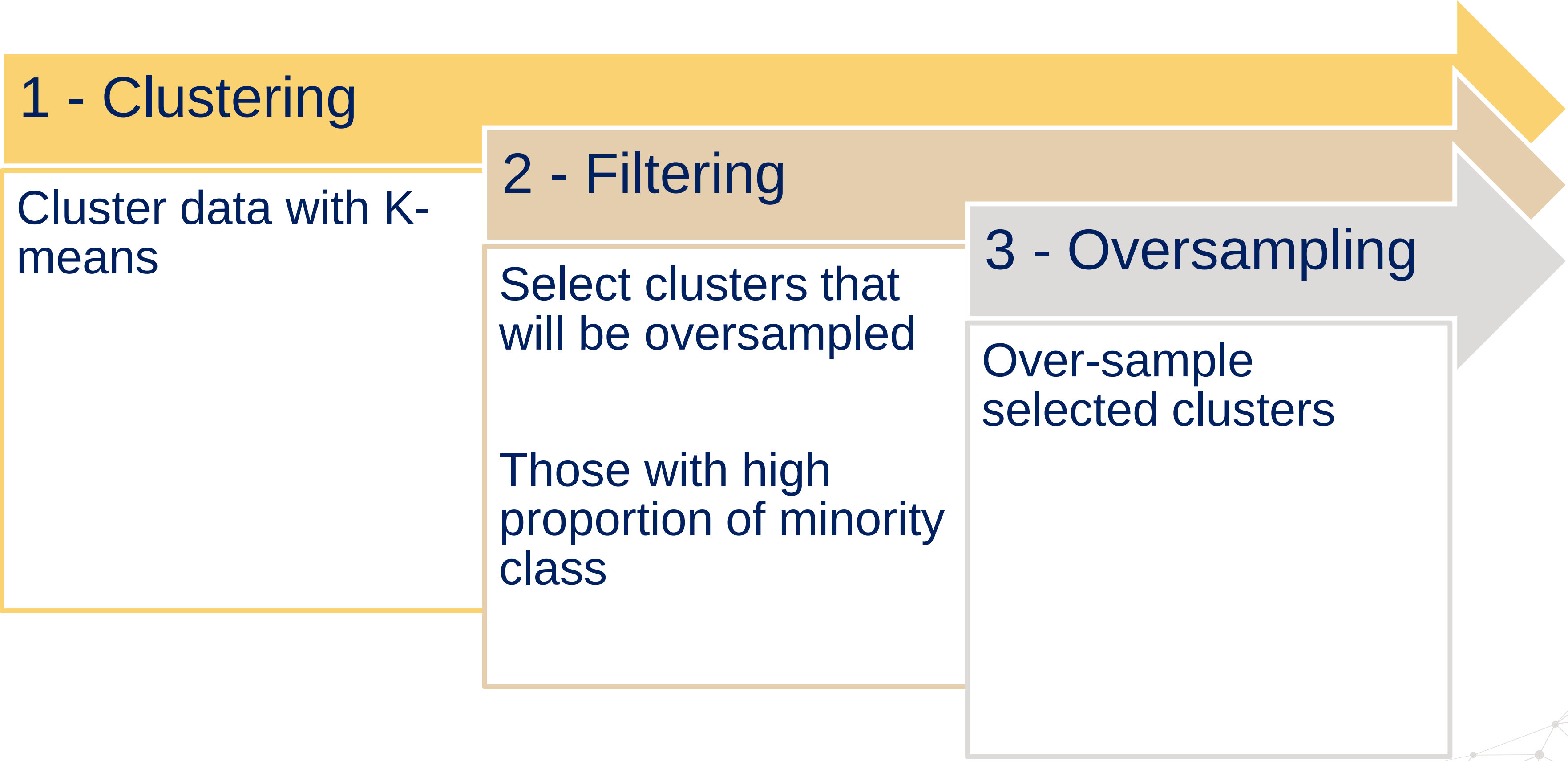
K-Means SMOTE – the idea

- Boost minority class regions by creating samples within naturally occurring clusters of the minority class.

- Cotemplate intra-class clusters

- Avoid introducing noise

K-Means SMOTE - 3 steps



1 - Clustering

Cluster data with K-means

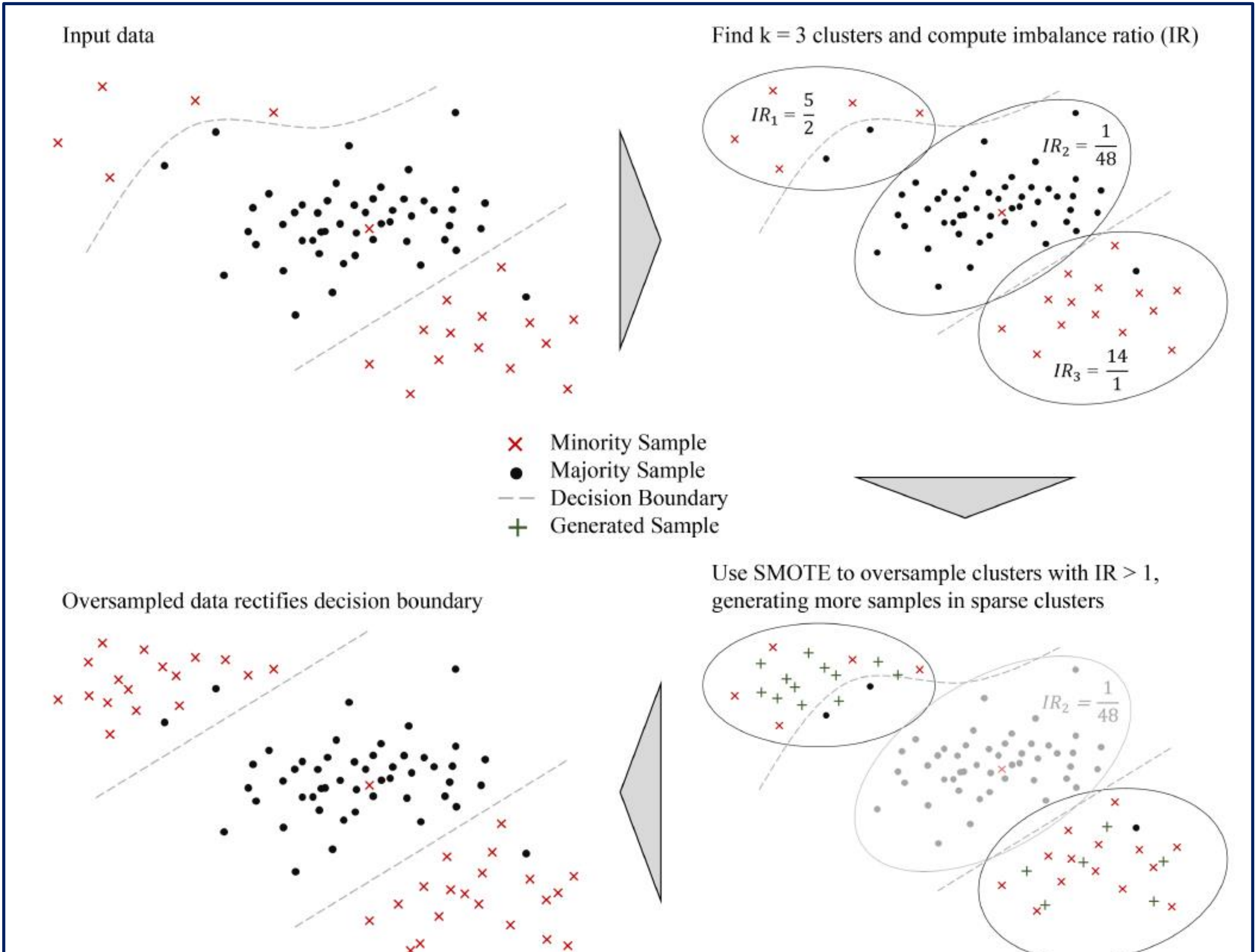
2 - Filtering

Select clusters that will be oversampled

Those with high proportion of minority class

3 - Oversampling

Over-sample selected clusters

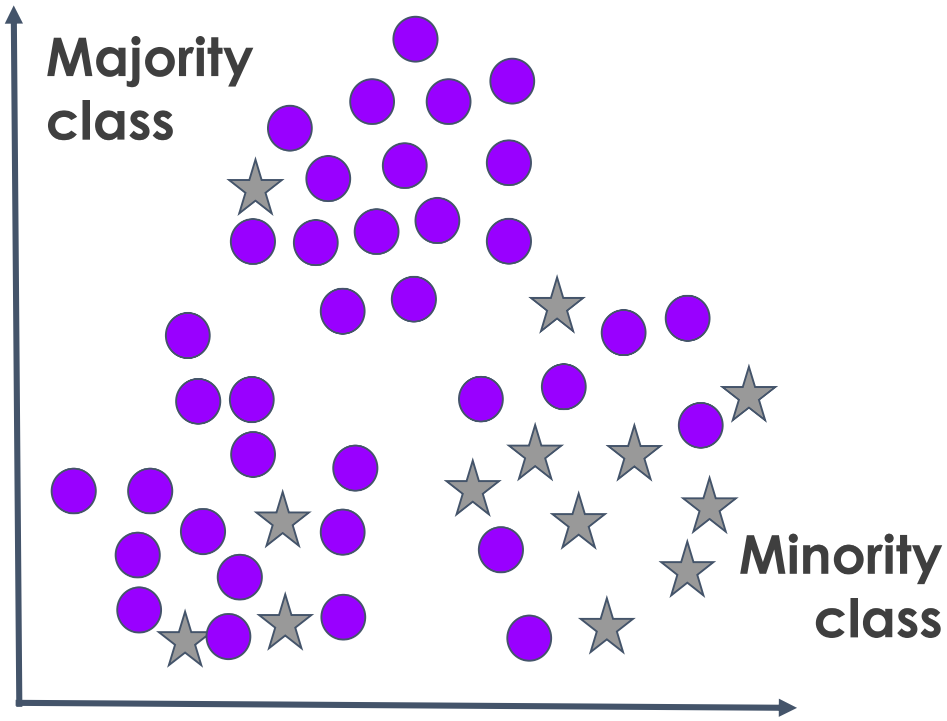

Input data
Find k = 3 clusters and compute imbalance ratio (IR)
$IR_1 = \frac{5}{2}$
$IR_2 = \frac{1}{48}$
$IR_3 = \frac{14}{1}$
Minority Sample
Majority Sample
Decision Boundary
Generated Sample
Oversampled data rectifies decision boundary
Use SMOTE to oversample clusters with IR > 1, generating more samples in sparse clusters
$IR_2 = \frac{1}{48}$

https://arxiv.org/pdf/1711.00837.pdf

Diagram showing the 3 steps of K-means SMOTE

By default clusters where 50% are minority are selected by default

Clustering step

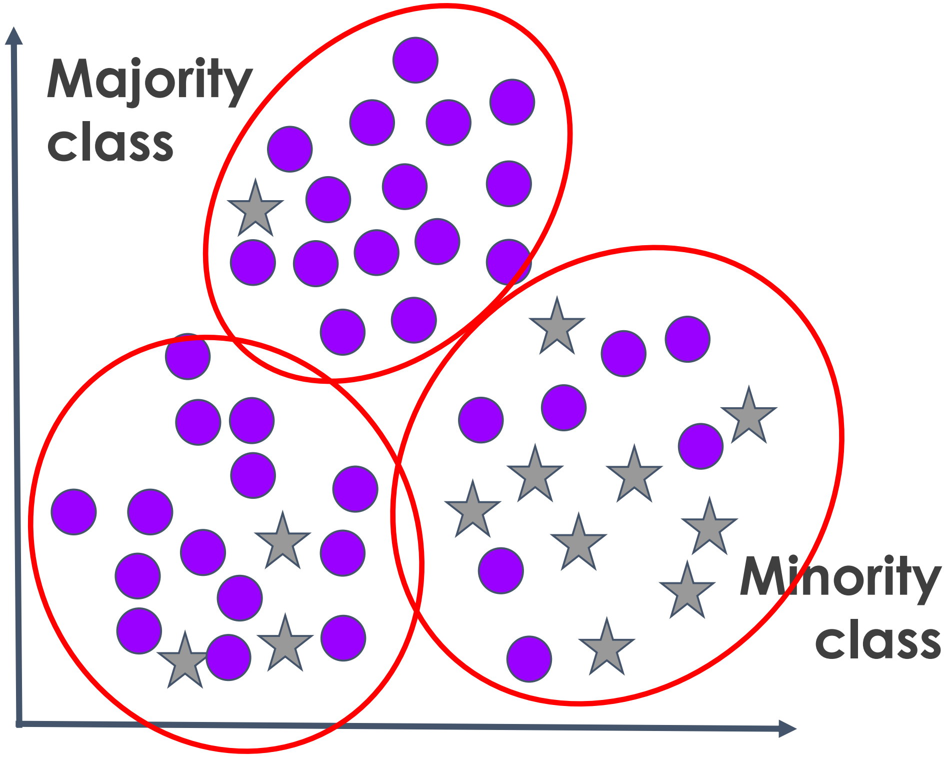


Find clusters ➔ K means algorithm with entire dataset

We need to know K, or treat K as a hyperparameter

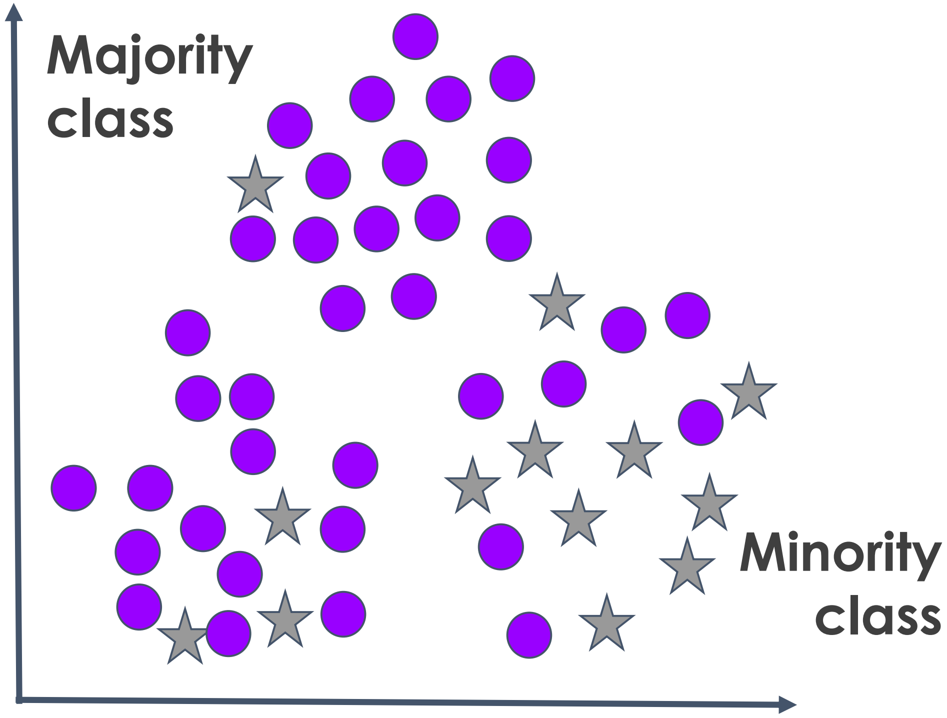
K-means may be slow to converge in huge datasets

Filtering step



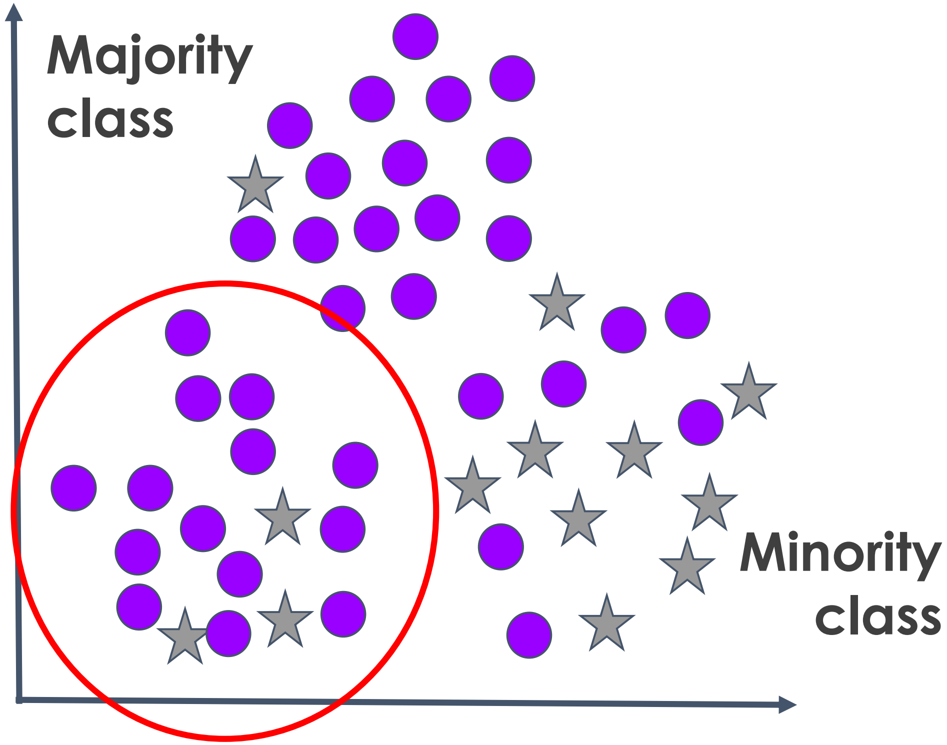
- By default select those clusters where 50% of observations belong to minority
- IR = 1 = # minority / # majority
- We can increase IR, thus we select clusters with higher proportion of minority class
- IR becomes another hyperparameter

Over-sampling



- Determine how many samples to create in each cluster
- Asign weights to clusters, more weights to clusters with less minority observations.

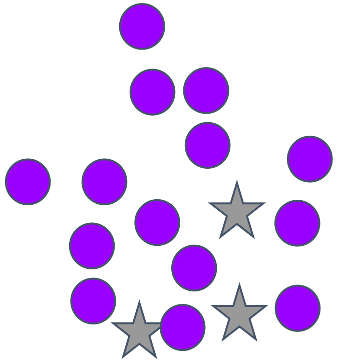
Over-sampling



We take this cluster to proceed with the demo

- Determine how many samples to create in each cluster
- Asign weights to clusters, more weights to clusters with less minority observations.

Cluster weight



1. Determine Euclidean distance between all samples from minority

2. Determine Mean Euclidean distance

3. density = $Xmin / L2mean^{number\ of\ features}$

3. Sparsity = 1 / density

4. Cluster sparsity = Sparsity /sum(Sparsity all clusters)

Cluster weight

1. Determine Euclidean distance between all samples from minority

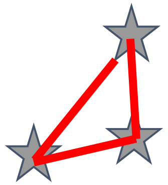
2. Determine Mean Euclidean distance

3. density = ${}^{Xmin}/{}_{L2mean^{number\ of\ features}}$

3. Sparsity = 1 / density

4. Cluster sparsity = Sparsity /sum(Sparsity all clusters)

Cluster weight



1. Determine Euclidean distance between all samples from minority

2. Determine Mean Euclidean distance (L2-mean)

3. density = $^{\# minority}/_{L2mean^{number\ of\ features}}$

3. Sparsity = 1 / density

4. Cluster sparsity = Sparsity /sum(Sparsity all clusters)

K-Means SMOTE

- Calculate the number of synthetic examples that need to be generated for each cluster

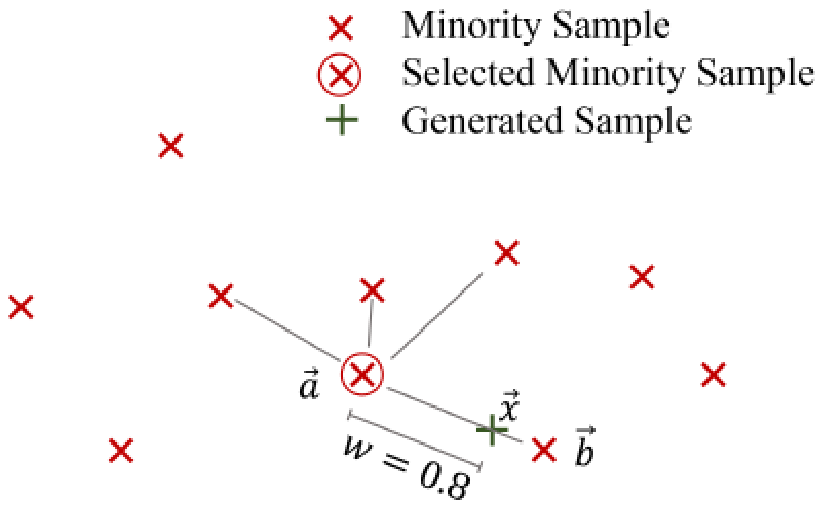
$$g_i = cs_i \text{ X } G$$

csi = cluster sparsity

G = total number of samples to generate

gi = number of samples to generate from cluster i

SMOTE with samples in cluster



https://arxiv.org/pdf/1711.00837.pdf

- Linearly interpolate between sample and neighbour chosen at random

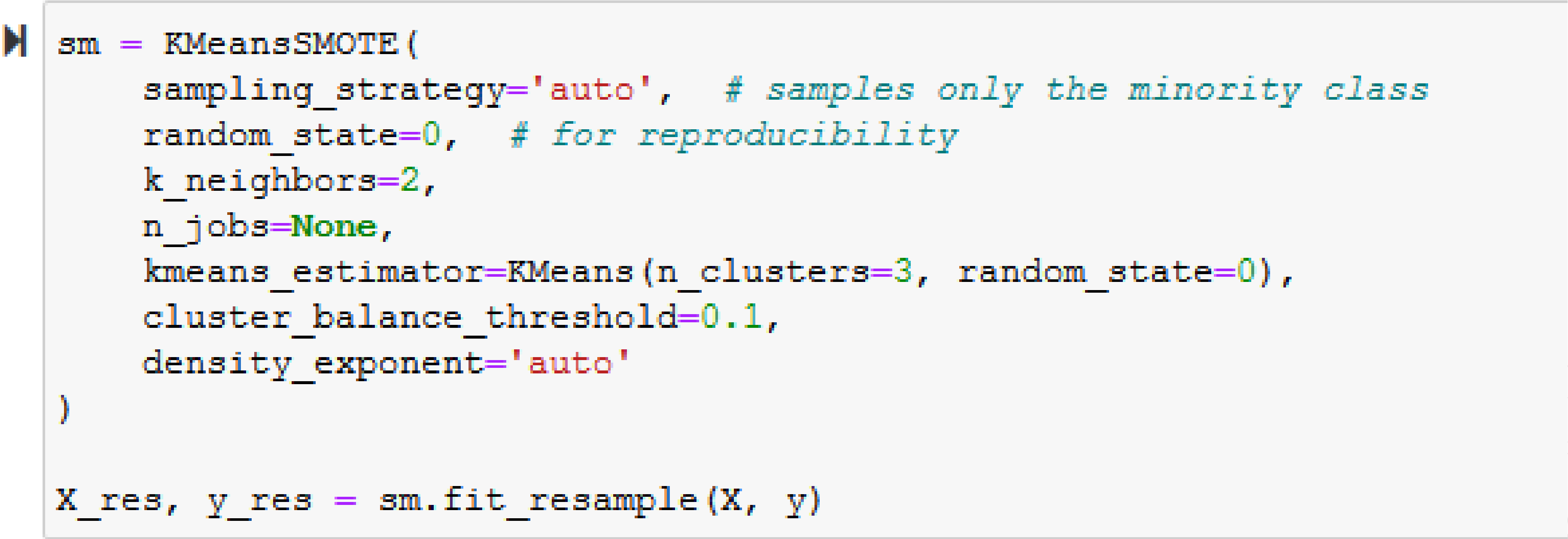
- If cluster has few samples, we may not have enough neighbours

- The number of neighbours becomes a hyperparameter

K-mean SMOTE – my thoughts

- The rationale is well thought, it makes sense
- The implementation of the algorithm is not super straight forward
- A lot of parameters to adjust
- Potentially some EDA to corroborate those parameters

Imbalanced-learn: KMeansSMOTE



```
sm = KMeansSMOTE(
    sampling_strategy='auto',  # samples only the minority class
    random_state=0,  # for reproducibility
    k_neighbors=2,
    n_jobs=None,
    kmeans_estimator=KMeans(n_clusters=3, random_state=0),
    cluster_balance_threshold=0.1,
    density_exponent='auto'
)

X_res, y_res = sm.fit_resample(X, y)
```

Train In Data

THANK YOU

www.trainindata.com