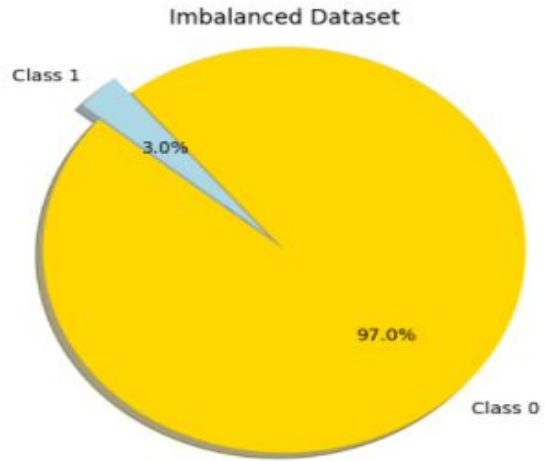
ML with Imbalanced Data

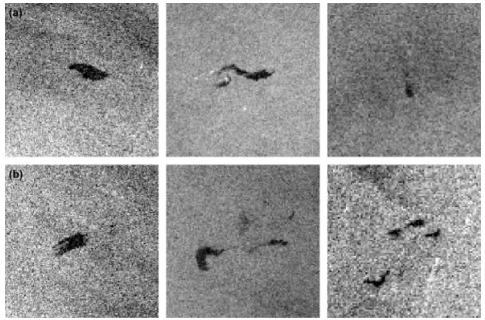
What is Imbalanced Data?

When the class of interest is much rarer (“Minority”) than the other class or classes (“Majority”) in the dataset.



Why is imbalanced data bad?

- The cost of missing a minority Class is much higher than a Majority class
  - Some Use Case examples
    - Cancer detection , where minority of classes are positive
    - Oil spill detection from satellite imagery
    - Fraud Detection : employee, bank, telecommunication
    - Spam emails

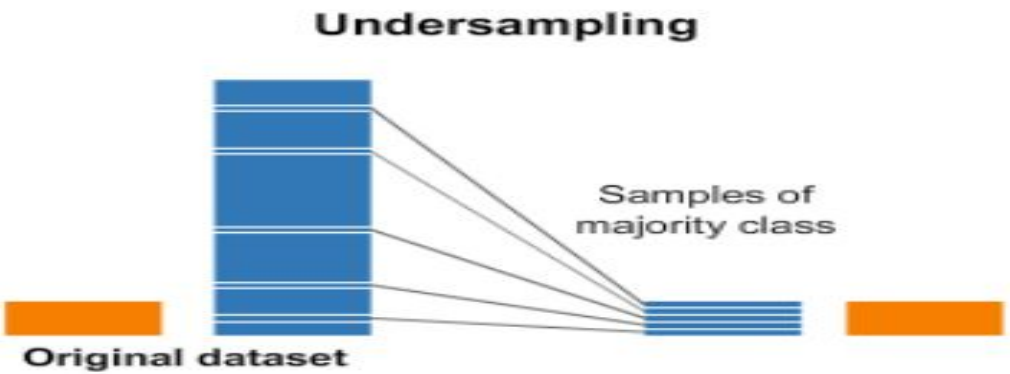


What can be done to help?

- Undersampling
- Oversampling
- SMOTE: Synthetic Minority OverSampling Technique
- ADASYN: Adaptive Synthetic Sampling Method
- Hyper parameter settings, weight class/cost , Cross Validation

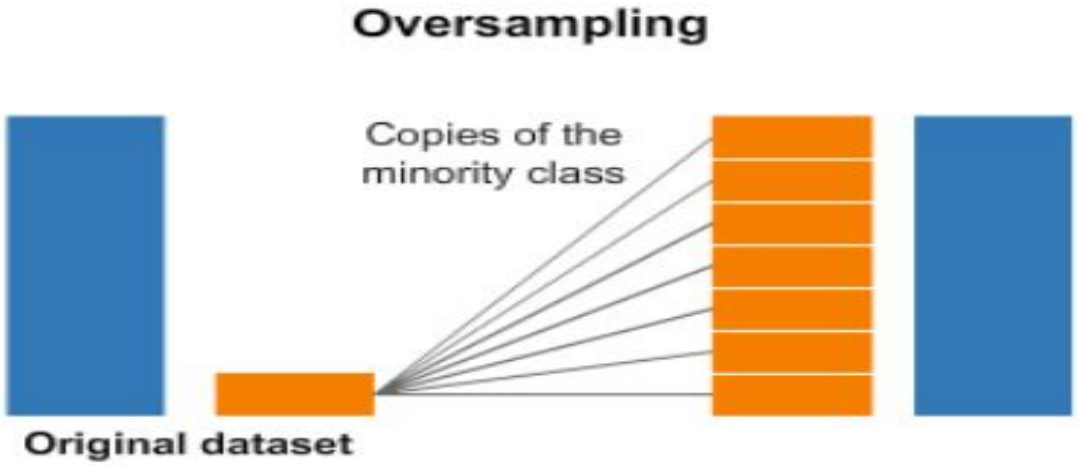
Undersampling

- Create random sample of the Majority Class to Match the Minority proportion.
- Drawback:
  - We lose lot of good data from majority Class
  - Model may not be generalized enough
- Workaround :create *Ensemble* models , trained separately , take Vote



Oversampling

- Generate more minority class random samples from existing minority set
- Drawback:
  - May be computationally expensive & time depending on dataset
  - May not be random enough depending on method used

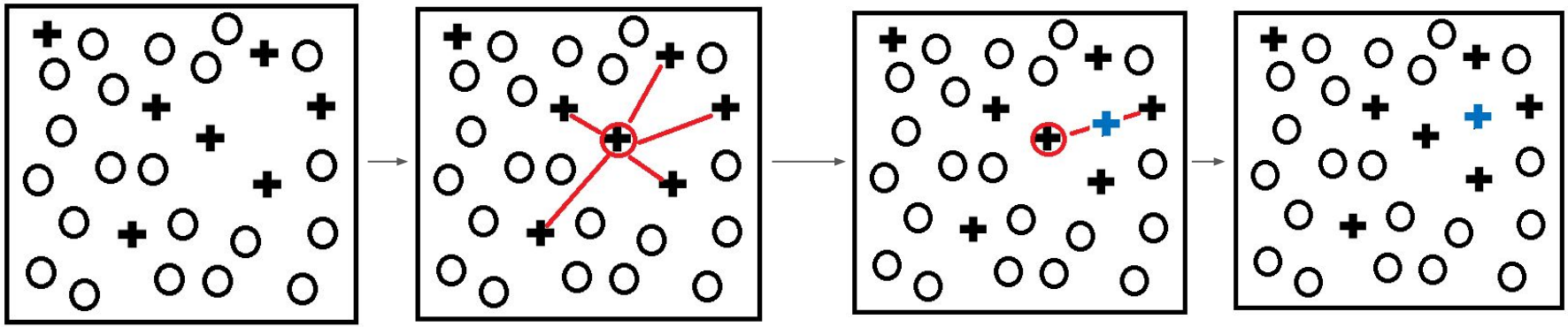


SMOTE

Synthetic Minority Oversampling Technique

The SMOTE algorithm at a high level:

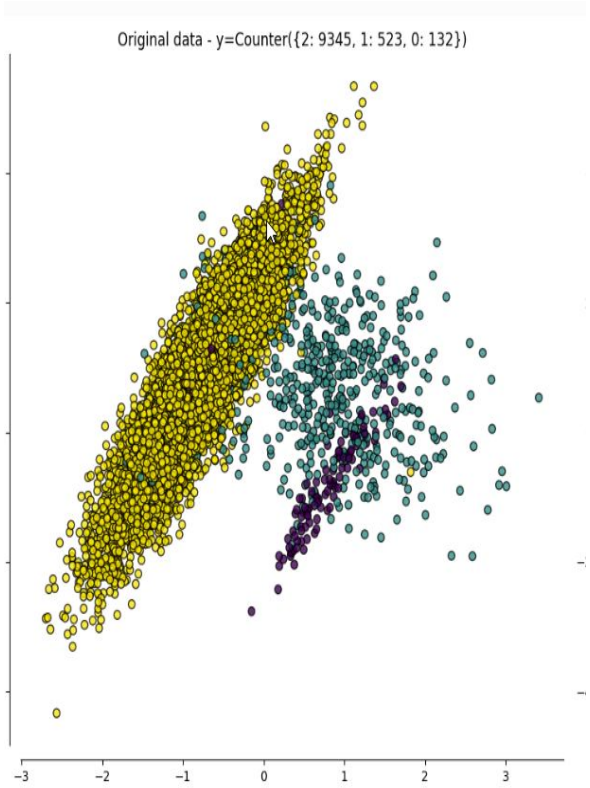
1. Randomly select a data point from minority class +
2. Find 5 K-nearest neighbours of that point
3. Randomly select one of the 5 neighbours
4. Generate a new synthetic value between + and the lucky neighbour

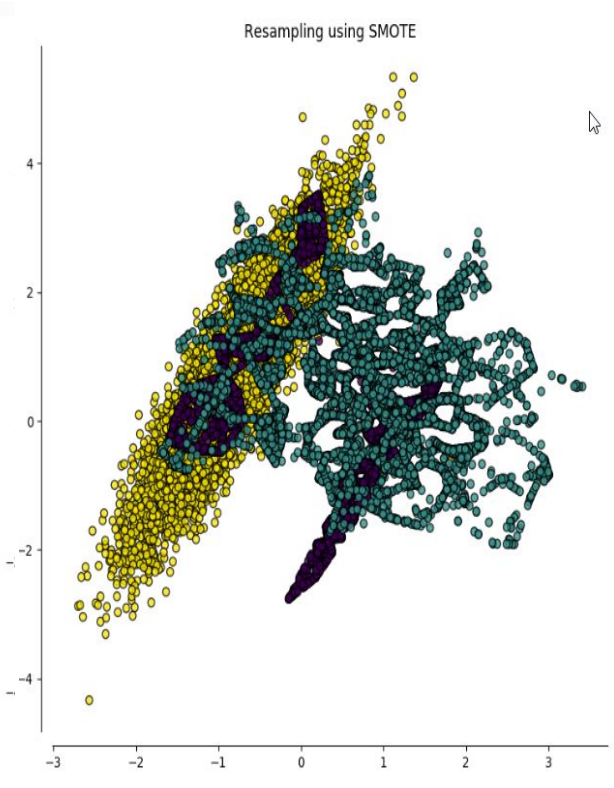


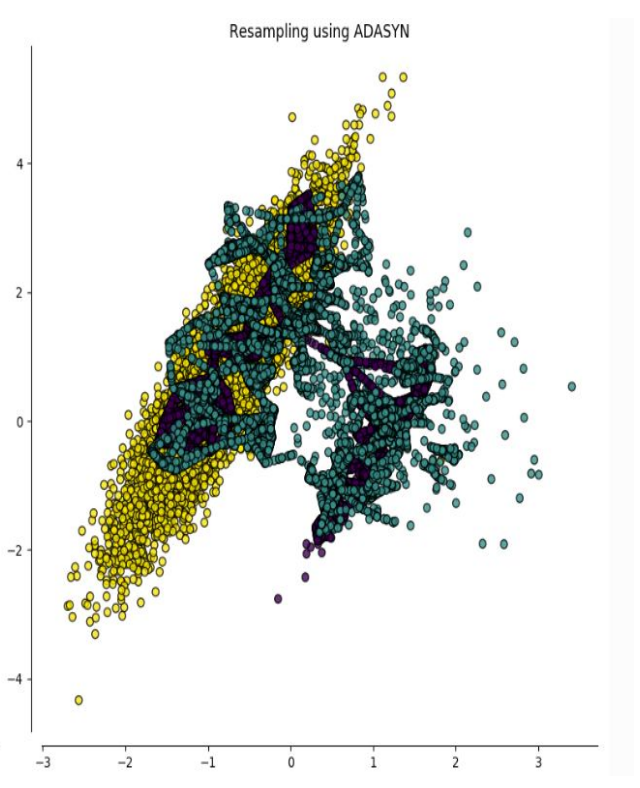
ADASYN

- Adaptive Synthetic Sampling Method
- An improved version of SMOTE
- After creating the samples it adds a random small values to the points thus making it more realistic.
- In other words instead of all the sample being linearly correlated to the parent they have a little more variance in them i.e they are bit scattered.
- ADASYN generally focuses on the samples which are difficult to classify using KNN where as SMOTE doesn’t.

SMOTE VS ADASYN







Hyper Parameters

- Most classifier models have Hyperparameters built in
- Assign a cost to weights
- Usually involve oversampling methods

class_weight : *dict or 'balanced', default=None*

Weights associated with classes in the form `{class_label: weight}`. If not given, all classes are supposed to have weight one.

References

https://imbalanced-learn.readthedocs.io/en/stable/index.html