# Oversampling methods introduction

1. Oversampling methods

* Process of increasing number of samples from the minority class
* until a desired balancing ratio

1. Random extraction vs sample generation

* Sample extraction: extract samples at random from minority class
* Random
* Sample generation: creates new samples from existing ones
* SMOTE and its variants
* ADASYN

A close-up of a sample

Description automatically generated

# Random oversampling

1. Overview

* Extracts obs at random from the minority class until a certain balancing ratio is reached
* Naïve technique

1. Duplication

* Extracts observations at random from the minority class, with replacement
* Duplicates samples from the minority class
* Increase likelihood of overfitting

1. Implementation

A black text on a white background

Description automatically generated

1. Multiclass

* 1 minority class: sampling\_strategy = ‘auto’
* Multiple minority classes: sampling\_strategy = ‘not majority’ or dict

# Random oversampling with smoothing

1. Overview

* Extracts obs at random from the minority class and compounds its value with some noise
* The noise is informed by the class distribution
* Creates new examples -> avoids data duplication
* We can choose how disperse we want the new samples -> shrinkage factor (arbitrary)

1. Procedure

* Take the minority samples and determine their distribution -> Std of each variable
* Extract a value at random from N ~(0, 1)
* Extract at random 1 observation
* Add the noise (with the shrinking factor) = shrinking factor x std x random value

1. Implementation

A screenshot of a computer code

Description automatically generated

A graph of a number of dots

Description automatically generated with medium confidence

# SMOTE

1. Overview

* Synthetic Minority Oversampling Technique
* Creates samples by interpolation – a type of estimation where new data points are created within the range of known data points
* Minority class is oversampled by creating synthetic examples instead of extracting data at random
* Prevents duplication

1. How it works

* Looks only at the obs from the minority class
* Finds its k nearest neighbours (typically k=5)
* Determines the distance between the neighbours and the sample we want to generate a new observation from
* Multiplies that distance by a random number and adds it to the original sample to place the new obs in the dataset

A diagram of a star formation

Description automatically generated

1. Numerical example
2. Python implementation

* Isolates minority class samples
* Trains KNN and finds K nearest neighbours to each sample of the minority class
* Determines how many new samples need to be generated
* Selects from which samples a new sample will be generated (random)
* Selects the neighbour that will be used to extrapolate the sample (random)
* Finds a random factor
* Determines new sample value

1. Implementation

A close-up of a math problem

Description automatically generated

# SMOTE-NC

1. Overview

* SMOTE – Nominal Continuous
* Extends functionality of SMOTE to categorical variables
* SMOTE, its variants and ADASYN can’t work with categorical data (unlike random oversampling)

1. How it works

* Calculate the Euclidean distances to find K neighbours
* Calculate standard deviations of numerical variables in minority class -> median of standard deviations. When calculating L2 distance between 2 obs.
* If values in categorical var are the same -> 0
* If values in categorical var are different -> use the median stddev.
* Values of numerical variables are calculated as in SMOTE
* Values of categorical variables are those shown by the majority of neighbours

1. Implementation

A screenshot of a computer code

Description automatically generated

# SMOTE-N

1. Overview

* Nominal variables, only (vs. SMOTE-NC requires dataset with both continuous and categorical variables)
* Extends the functionality of SMOTE to categorical variables

1. How it works

* Looks only at the minority class examples
* Find the k (usually 5) closest neighbors?
* Distance: value difference metric
* Determine the values of the newly created examples?
* Majority vote

1. Distance in numerical vectors
2. Distance in categorical vectors

* **Value Difference Metric (VDM)**
* Calculate the difference between the values of a variable
  + : number of examples in the training set that have value for variable
  + : number of examples in the training set that have value for variable given class (conditional probability)
  + : number of classes
  + : a constant, usually 1 or 2
* E.g.

A screenshot of a computer

Description automatically generated

* **Distance between observations**
* : features
* : typically 1 or 2

1. SMOTE-N procedure

* With the VDM -> distances
* Train a KNN
* Find the K nearest neighbors of each obs from the minority
* Values of the new examples are those shown by the majority of the neighbours

1. Implementation

A computer screen shot of a computer code

Description automatically generated with medium confidence

ADASYN

1. Overview

* Uses a weighted distribution of the minority class according to how difficult the obs are to be learned / classified
* More synthetic data is generated from the samples that are harder to classify
* SMOTE: use all samples from the minority class to create the synthetic data
* ADASYN: use more samples that are harder to classify and less that are easy to classify to create the synthetic data

1. How it works

* Step 1: determine imbalance ratio
* Step 2: determine number of samples to generate
* If and -> G = 800
* Factor is 1 for full balancing, or balancing ratio of 1
* Step 3:
* Train KNN using entire dataset (not only minority class as SMOTE)
* Find K closest neighbors for each sample of minority class
* Determine the weighting r
* Step 4: normalize r (1 r value per observation of the minority class)
* Step 5: calculate number of synthetic examples that need to be generated for each obs of the minority class
* Step 6: for each minority class example , generate synthetic examples
* The neighbor can be from the majority or minority class
* The KNN is trained on the entire dataset

1. Implementation

A white background with text

Description automatically generated

A diagram showing a number of dots

Description automatically generated with medium confidence

# Borderline SMOTE

1. Overview

* Extension of SMOTE which creates synthetic examples only from the observations in the minority class closer to the boundary with the majority class(es)
* Presents 2 variants (1 & 2)

1. Procedure

* Fits KNN with all the dataset
* Finds and ignores observations from the minority class which K neighbours belong to the majority class -> noise & irrelevant
* Finds and ignores observations from the minority class if most of the neighbours are from the minority class -> safe and easy to classify
* Selects the observations from the minority class if most of their neighbours are from the **majority** class
* Fits KNN to minority class examples
* Variant 1: interpolates synthetic samples as SMOTE, between the observations in the DANGER group and its neighbours from the minority class
* Variant 2: Variant 1 + interpolates synthetic examples between the observations in the DANGER group and its neighbours in the majority class but closer to the DANGER group
  + Factor can take values between 0 and 0.5 instead of 0 and 1

1. Implementation

A screen shot of a computer code

AI-generated content may be incorrect.

# SVM SMOTE

1. Oversampling recap

A diagram of a sample

AI-generated content may be incorrect.

1. Templates for synthetic data

* ADASYN: samples from the minority if some of their closest neighbours are from the opposite classes. The more neighbours from the opposite class, the more likely it is to be used as template
* Borderline SMOTE: observations from the minority for which the majority of the neighbours are from the opposite class
* SVM SMOTE: observations from the minority that are the support vectors of a SVM

1. Sample for interpolation

* ADASYN: Interpolate from template to closest neighbour from minority class only
* Borderline SMOTE:
* Variant 1: Interpolate from template to closest neighbour from minority class only
* Variant 2: interpolate from template to closest neighbour from minority or majority but with half the distance in the latter
* SVM SMOTE: Inter or extrapolates to neighbours from the minority
* If most neighbours are from **minority** class -> **extrapolation** (to expand the boundary)
* If most neighbours are from **majority** class -> **interpolation** (keep the boundary as is)

1. Support vectors

* Step 1
* With a SVM find the support vectors
* Select the support vectors from the minority class
* These are the templates
* Inter vs. extrapolation

A diagram of a person's relationship

AI-generated content may be incorrect.

* Train a KNN algorithm on the entire dataset
* Usually to find the 10 closest neighbours
* If most neighbours from the minority -> extrapolation
* If most neighbours from the majority -> interpolation (new sample created within 2 existing samples, stays within boundary)
* Support vector and neighbour are from minority class
* Factor from 0 to 1
* Differently from SMOTE, the neighbour is not chosen at random, it selects from closest to furthest neighbour in order
* Another KNN is trained on the minority group
* Usually to find the 5 closest neighbours -> for creating new synthetic data
* In total, we fit 1 SVM and 2 KNNs (could be very costly in practice)

1. Implementation

A computer screen shot of text

AI-generated content may be incorrect.

# K-means SMOTE

1. Overview

* SMOTE linearly interpolates a randomly selected minority sample and one of its k=5 neighbours
* SVM, Borderline SMOTE: create samples at the boundary is better than in areas that are safe
* SMOTE might create noisy examples (e.g. surrounded by majority class)
* SMOTE does not contemplate intra-class clusters (e.g. fraudulent credit card transactions)
* K-means SMOTE – the idea:
* Boost minority class regions by creating samples within naturally occurring clusters of the minority class
* Contemplate intra-class clusters
* Avoid introducing noise

1. Procedure

* Cluster data with k-means -> need to know k or treat k as a hyperparameter
* Select clusters that will be oversampled (i.e. those with high proportion of minority class)
* By default select those clusters where 50% of obs are minority class
* We can increase IR -> another hyperparameter
* Oversample selected clusters
* Determine how many samples to create in each cluster
* Assign weights to clusters, more weights to clusters with less minority obs
  + Determine Euclidean distance between all samples from the minority
  + Determine mean Euclidean distance (L2-mean)
* Calculate number of synthetic examples that need to be generated for each cluster
* SMOTE with samples in cluster
  + 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
* Implementation not straightforward
* A lot of parameters to adjust
* Potentially some EDA to corroborate those parameters

1. Implementation

A screenshot of a computer code

AI-generated content may be incorrect.

# How to set up a classifier with oversampling

1. Generalization

* Important to determine performance in data with class imbalance – test set
* Resample the train set only
* Train the model on the resampled train set
* Then evaluate on the test set

1. Cross-validation

* Train set divided into k folds
* Resample k-1 folds
* Train model on resampled k-1 folds
* Test model on the kth fold (not resampled)
* Repeat k times
* Need to set up resampling within the cross-validation pipeline