Learning from Imbalanced Data

Leonidas Akritidis[1]

[1]School of Science and Technology, International Hellenic University

14th International Conference on Information, Intelligence, Systems and Applications (IISA 2023)
University of Thessaly, Volos, Greece
10 – 12 July, 2023

About the presenter

- Leonidas Akritidis
- PhD in Electrical and Computer Engineering
- Postdoctoral researcher in the School of Science and Technology, International Hellenic University.
- Adjunct lecturer in the School of Science and Technology, International Hellenic University.
- Research interests: Machine Learning, Deep Learning, Data Mining, Big Data, Information Retrieval.
- Contact: lakritidis@ihu.gr

About this tutorial

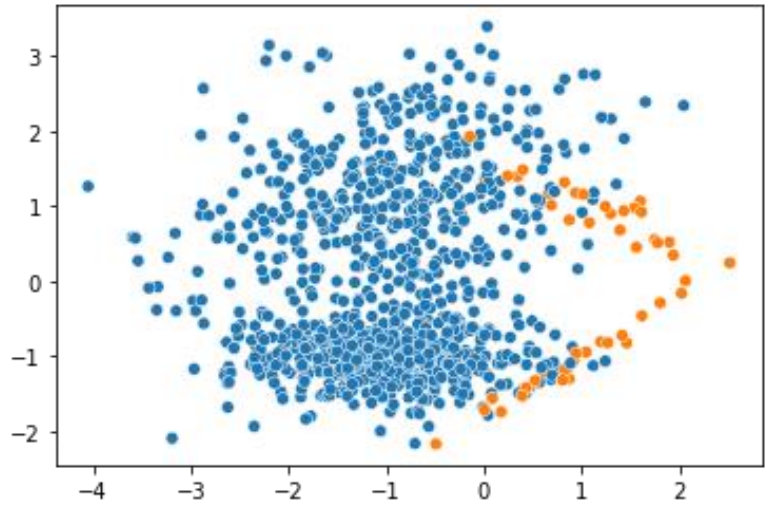
- Methods for confronting the problem of class imbalance in classification tasks.
- This tutorial includes:
  - These slides.
  - Four notebooks with code and descriptions.
- Supported by the survey paper:
  - L. Akritidis, P. Bozanis, "A Multi-Dimensional Survey on Learning from Imbalanced Data", Chapter in Machine Learning Paradigms - Advances in Theory and Applications of Learning from Imbalanced Data, to appear, 2023.
- Supported by the Github repo https://github.com/lakritidis/imbalanced

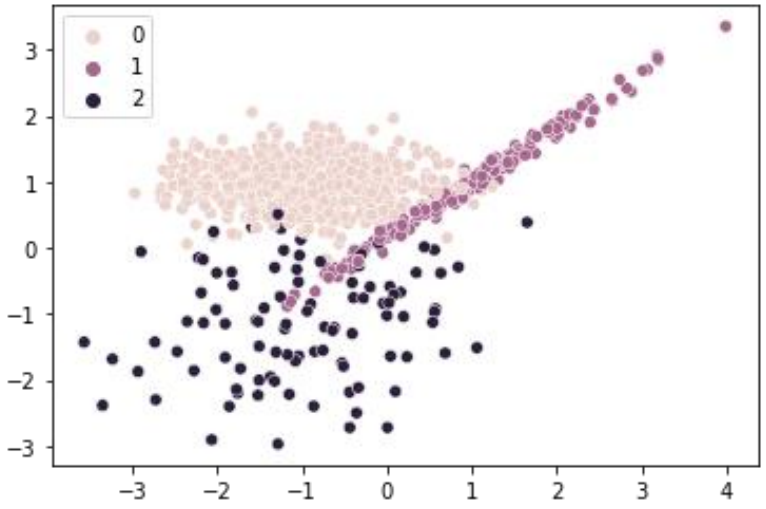
Organization

- The tutorial is organized in 4+2 parts:
  1. **Introduction.**
  2. **Part A:** Imbalanced data: descriptions, problems, performance evaluation measures.
  3. **Part B:** Oversampling, Undersampling, Hybrid sampling.
  4. **Part C:** Oversampling with Generative deep learning methods.
  5. **Part D:** Oversampling & undersampling with Boosting and Bagging.
  6. **Epilogue, Conclusions, Discussion.**
- Parts A-D are accompanied by live demonstrations.

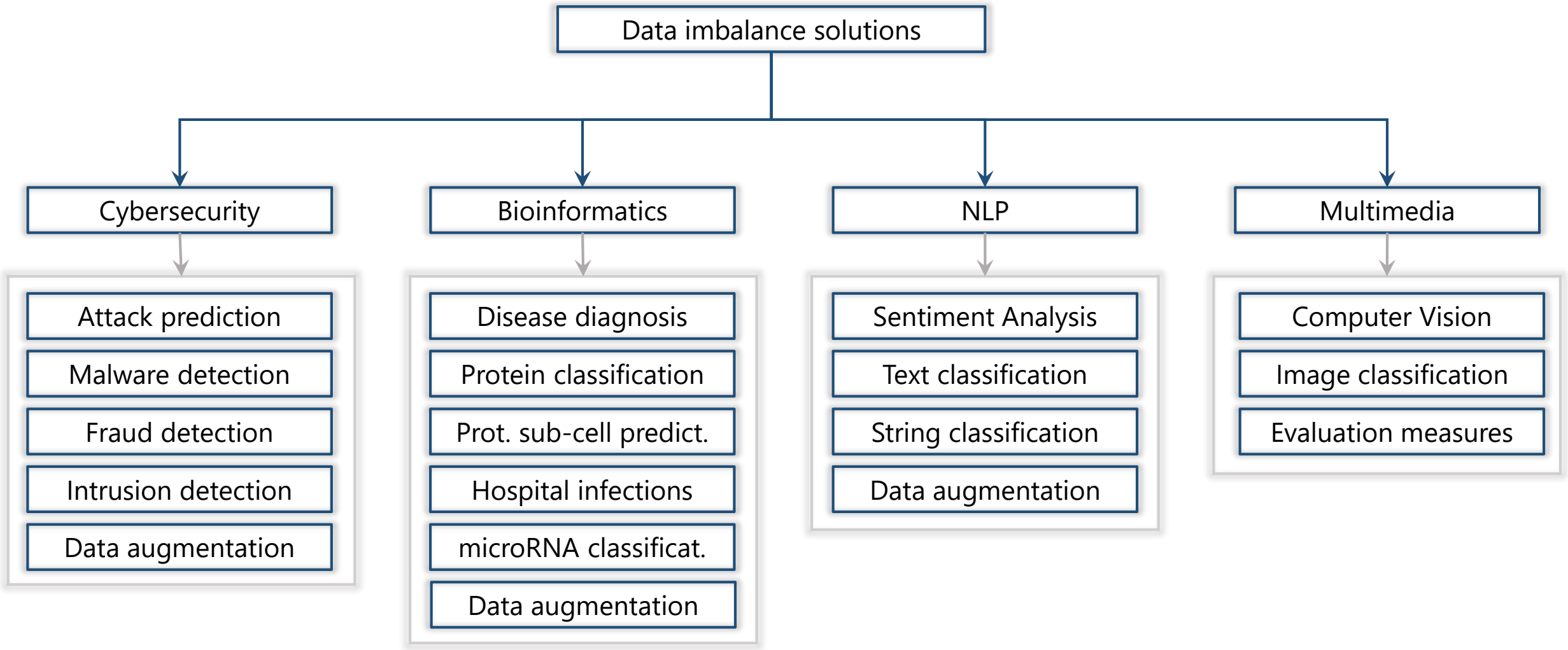
Imbalanced data

- Related to the uneven distribution of the training examples to the involved classes.
- Present in numerous application fields:
  - Both binary and multi-class.
  - Cybersecurity, Bioinformatics, Natural Language Processing, Image Processing,…





Application-based classification

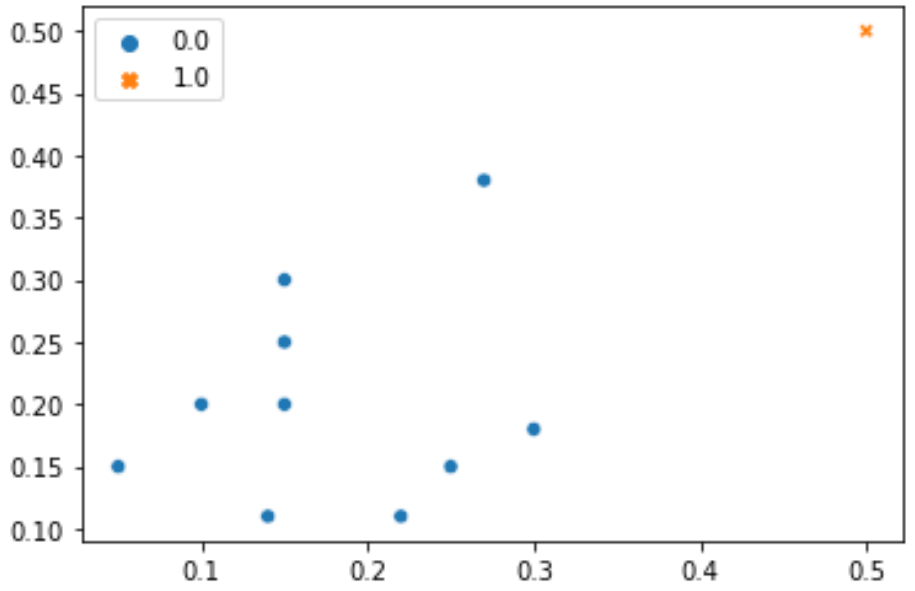


Problems caused by imbalanced data

- Predictors become strongly biased towards the majority class.
- They cannot effectively learn the minority classes.
- Their performance degrades rapidly.
- The accuracy metric is misleading in these kinds of problems (more on this later).
  - There is a requirement to use/introduce other evaluation measures.

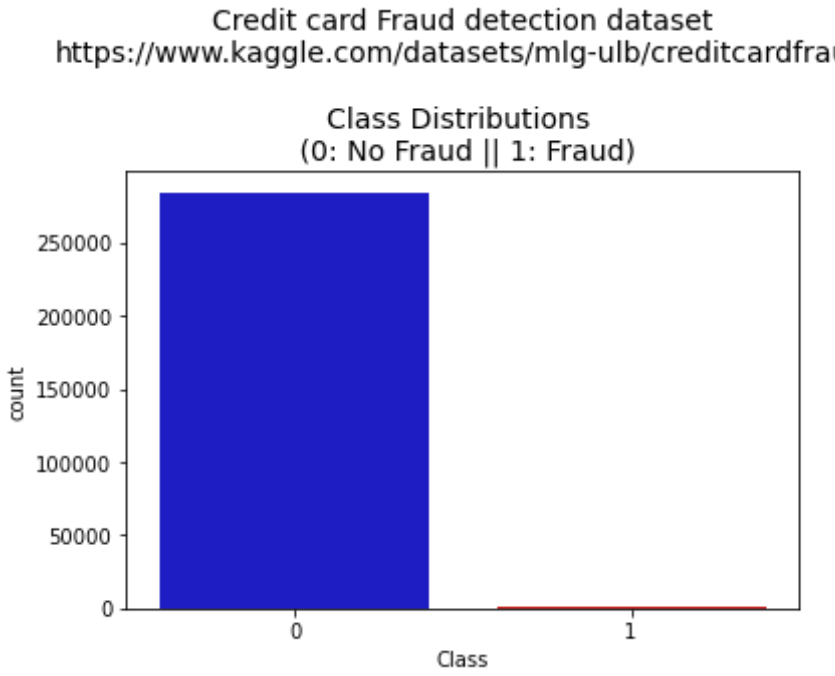
Definitions

- **The class that contains the majority of the samples is called majority class.**
- **The other classes are known as minority classes.**
- Dataset Imbalance Ratio (IR)
- $$IR = \frac{\text{number of samples from the majority class}}{\text{1 sample from the minority class}}$$
  - $IR = 10{:}1$ from 11 random samples, 10 correspond to the majority class and 1 corresponds to the minority class.



Extreme Data Imbalance

- In extreme situations, the majority class vastly outnumbers the minority classes (an imbalance ratio of say, 1000:1, or more), so the problem is often referred as an **extreme data imbalance problem**.

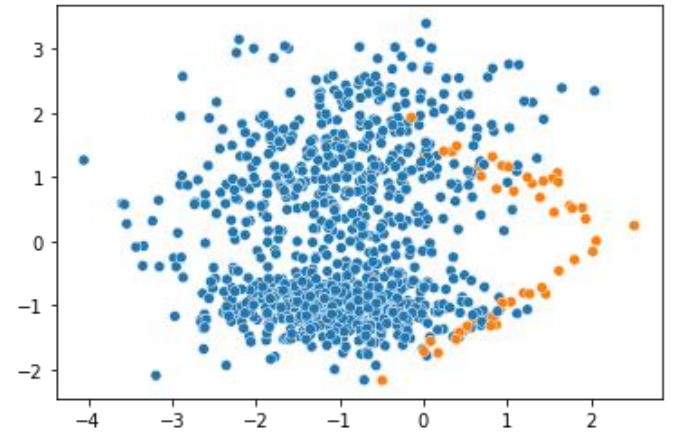


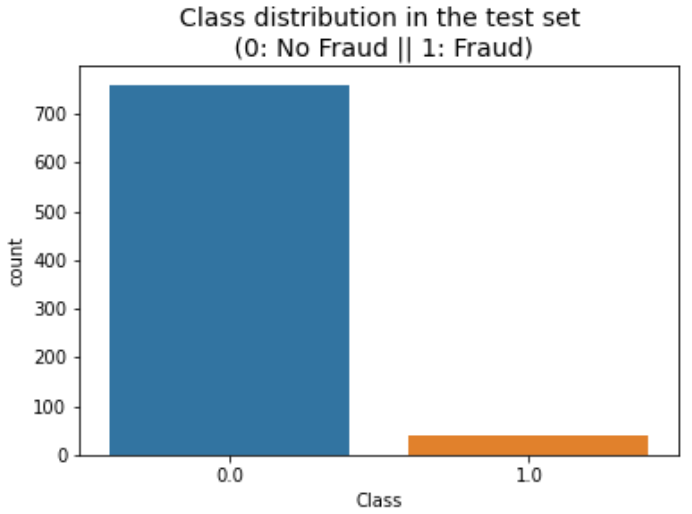
Classification & Accuracy

- Data imbalance is independent of the underlying task:
  - It negatively affects classification, regression, clustering, etc. tasks.
- The scientific literature mainly deals with classification problems:
  - Attack detection, intrusion detection, credit card fraud detection, disease detection, object recognition, etc.
- Accuracy is the most popular metric for evaluating the performance of a classification model.
- But when the dataset is imbalanced, Accuracy has problems:
  - **It erroneously receives very high values even when the classifier is weak.**

An example

- Consider a dataset with 1000 samples, 2 classes, $IR = 19{:}\,1$.
  - 950 samples from class 0, 50 samples from class 1.
- Now let's say that we desire to build a classifier;
- 80% / 20% for training/testing.
- Test set: 200 samples (class 0: 190, class 1: 10).
- Let's use a dummy classifier that always outputs 0:
- $\text{Accuracy} = \frac{\text{correct predictions}}{\text{total predictions}} = \frac{190}{200} = 0{,}95$





Accuracy is misleading

- This dummy classifier achieves an outstanding performance even though it always predicts 0!
- **The greatest the imbalance ratio, the highest the accuracy!**
- Apparently, other metrics are required to measure the classification performance.
- Metrics which are robust to class imbalance.

Performance evaluation measures (1)

- TP **(True Positive):** the actually positive examples that are correctly predicted as positive.
- TN **(True Negative):** the actually negative examples that are correctly predicted as negative.
- FP **(False Positive):** the actually negative examples that are incorrectly predicted as positive.
- FN **(False Negative):** the actually positive examples that are incorrectly predicted as negative.
- Accuracy: For a classification model, it measures the number of correctly predicted examples among all the total possible examples. Simply, the ratio of the correctly predicted examples to the total number of examples present.

Metric	Formula
Accuracy	$\frac{TP+TN}{TP+FP+FN+TN}$
Sensitivity	$\frac{TP}{(TP+FN)}$
Specificity	$\frac{TN}{(TN+FP)}$
Precision	$\frac{TP}{(TP+FP)}$
Recall	$\frac{TP}{(TP+FN)}$
F-measure	$\frac{2*Precision*Recall}{Precision+Recall}$
G-MEAN	$\sqrt{sensitivity * specificity}$
AUC	$\frac{1+TPR-FPR}{2}$

	Actual Positive	Actual Negative
Observed Pos	TP	FP
Observed Neg	FN	TN

Performance evaluation measures (2)

- **Precision:** the ratio of true positives (TP) to the total number of positive examples predicted.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall:** the fraction of true positive (TP) examples to all examples that are actually positive.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F-measure:** the harmonic mean of Precision and Recall.

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Metric	Formula
Accuracy	$\frac{TP+TN}{TP+FP+FN+TN}$
Sensitivity	$\frac{TP}{(TP+FN)}$
Specificity	$\frac{TN}{(TN+FP)}$
Precision	$\frac{TP}{(TP+FP)}$
Recall	$\frac{TP}{(TP+FN)}$
F-measure	$\frac{2*Precision*Recall}{Precision+Recall}$
G-MEAN	$\sqrt{sensitivity * specificity}$
AUC	$\frac{1+TPR-FPR}{2}$

	Actual Positive	Actual Negative
Observed Pos	TP	FP
Observed Neg	FN	TN

Performance evaluation measures (3)

- **Sensitivity, or True Positive Rate (TPR):** the positive samples that are correctly predicted by a model.

$$TPR = \text{Recall} = \frac{TP}{TP + FN}$$

- **Specificity, or True Negative Rate (TNR):** the negative examples that are correctly predicted by a model.

$$TNR = \frac{TN}{TN + FP}$$

- **False Positive Rate (FPR):**

$$FPR = \frac{FP}{FP + TN}$$

- **G-measure:**

$$G = \sqrt{TPR \cdot TNR}$$

Metric	Formula
Accuracy	$\frac{TP+TN}{TP+FP+FN+TN}$
Sensitivity	$\frac{TP}{(TP+FN)}$
Specificity	$\frac{TN}{(TN+FP)}$
Precision	$\frac{TP}{(TP+FP)}$
Recall	$\frac{TP}{(TP+FN)}$
F-measure	$\frac{2*Precision*Recall}{Precision+Recall}$
G-MEAN	$\sqrt{sensitivity * specificity}$
AUC	$\frac{1+TPR-FPR}{2}$

	Actual Positive	Actual Negative
Observed Pos	TP	FP
Observed Neg	FN	TN

Performance evaluation measures (4)

- **A ROC curve** (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots TPR vs FPR.

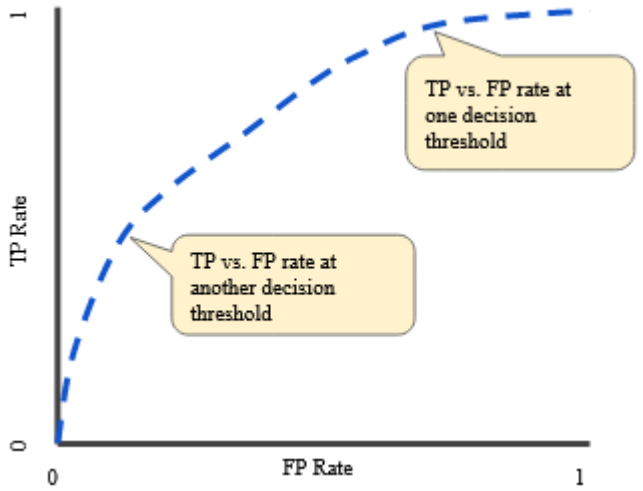


Image Source: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

Metric	Formula
Accuracy	$\frac{TP+TN}{TP+FP+FN+TN}$
Sensitivity	$\frac{TP}{(TP+FN)}$
Specificity	$\frac{TN}{(TN+FP)}$
Precision	$\frac{TP}{(TP+FP)}$
Recall	$\frac{TP}{(TP+FN)}$
F-measure	$\frac{2*Precision*Recall}{Precision+Recall}$
G-MEAN	$\sqrt{sensitivity * specificity}$
AUC	$\frac{1+TPR-FPR}{2}$

	Actual Positive	Actual Negative
Observed Pos	TP	FP
Observed Neg	FN	TN

Performance evaluation measures (5)

- **Area Under the Curve (AUC):** measures the surface of the two-dimensional area underneath the entire ROC curve from (0,0) to (1,1).
- Provides an aggregate measure of performance across all possible classification thresholds.
- AUC is the probability that the model ranks a random positive example higher than a random negative example.

Metric	Formula
Accuracy	$\frac{TP+TN}{TP+FP+FN+TN}$
Sensitivity	$\frac{TP}{(TP+FN)}$
Specificity	$\frac{TN}{(TN+FP)}$
Precision	$\frac{TP}{(TP+FP)}$
Recall	$\frac{TP}{(TP+FN)}$
F-measure	$\frac{2*Precision*Recall}{Precision+Recall}$
G-MEAN	$\sqrt{sensitivity * specificity}$
AUC	$\frac{1+TPR-FPR}{2}$

	Actual Positive	Actual Negative
Observed Pos	TP	FP
Observed Neg	FN	TN

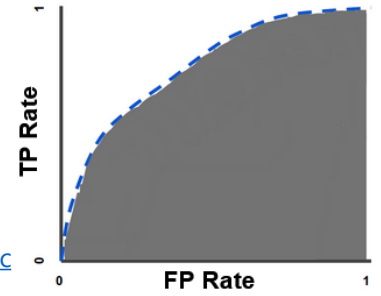


Image Source: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

Balanced Accuracy

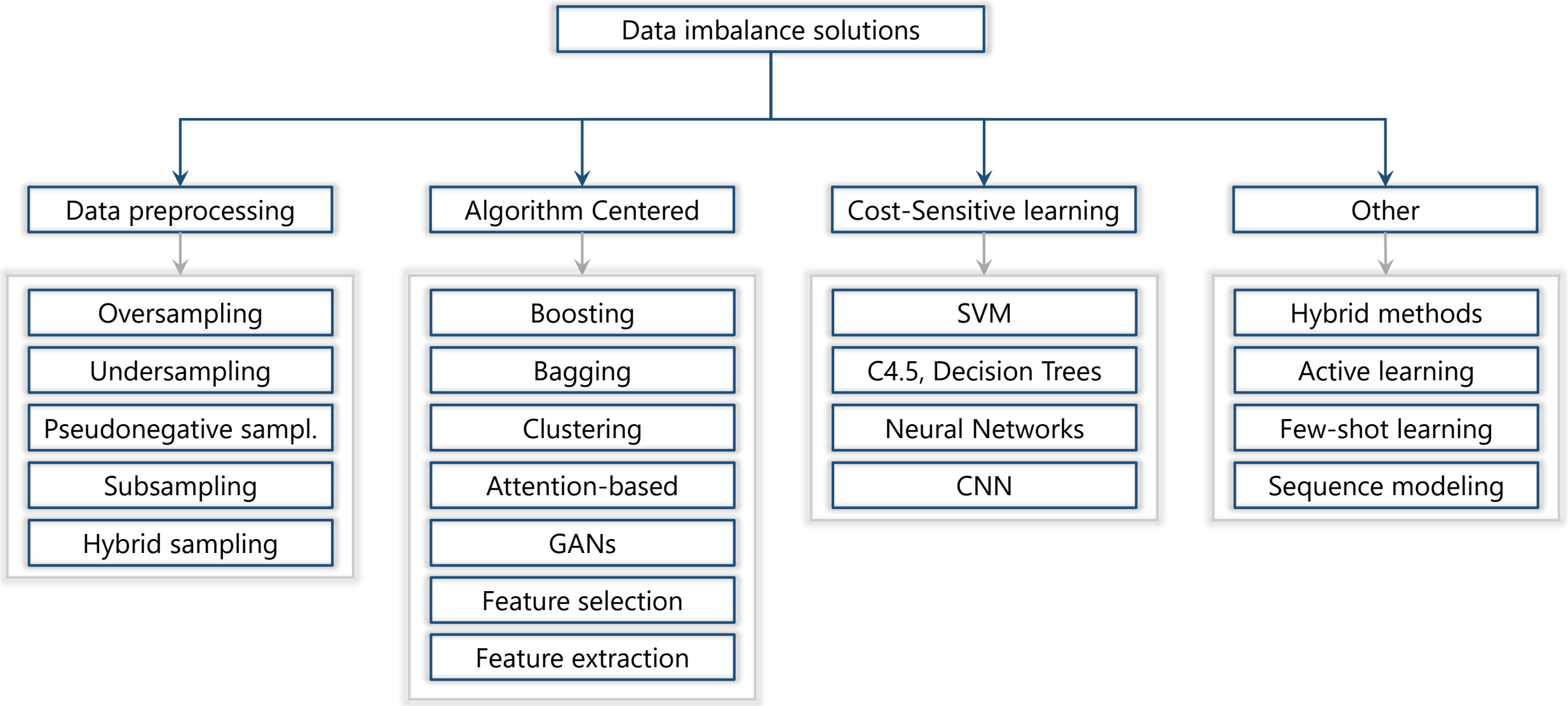
- Balanced Accuracy: computes the geometric mean of accuracy of minority and majority class.
- Or else, it is the average of the sum of Sensitivity and Specificity:

$$\text{Balanced Accuracy} = \frac{\text{sensitivity} + \text{specificity}}{2} = \frac{TPR + TNR}{2}$$

- For our previous dummy classifier:

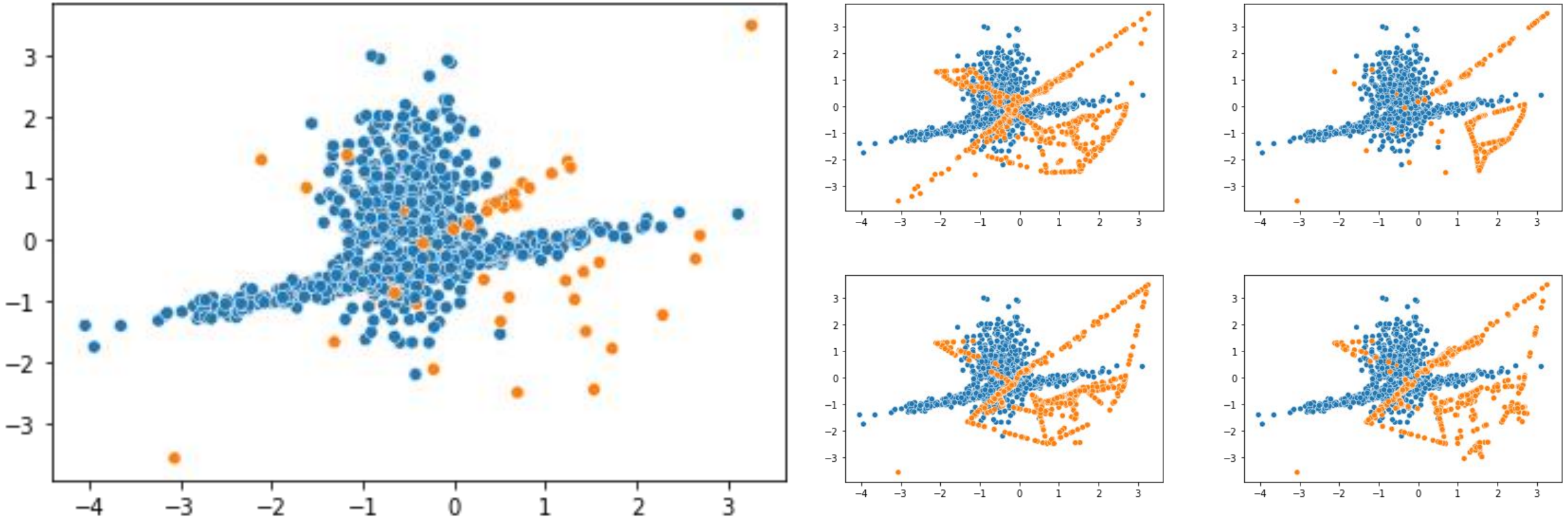
$$\text{Balanced Accuracy} = \frac{TPR + TNR}{2} = \frac{1 + 0}{2} = 0.5$$

Method-based classification

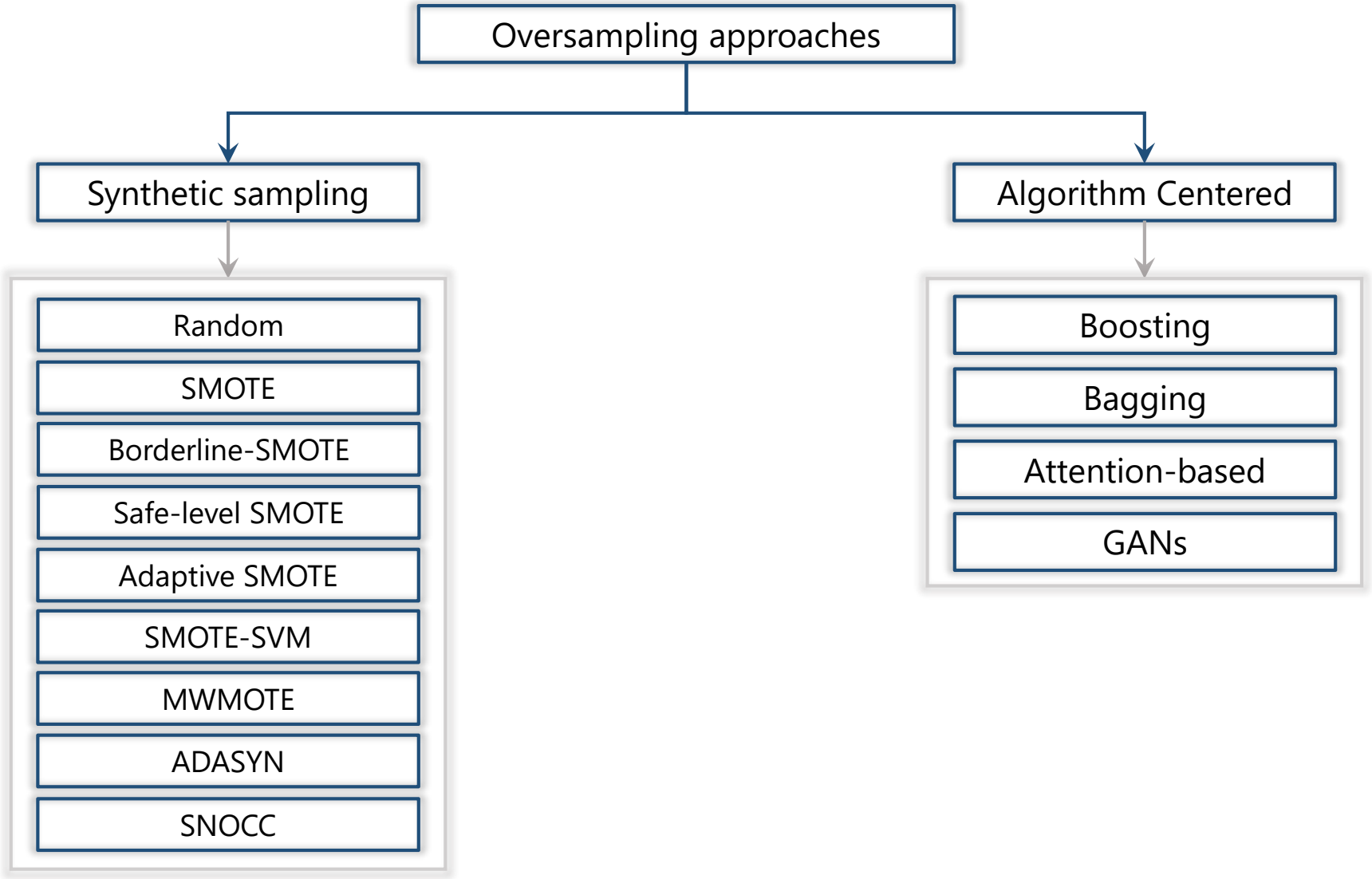


Oversampling

- Confront data imbalance by creating synthetic samples for the minority class/es.



Oversampling Approaches



Random Oversampling

- This naïve method simply replicates a portion of minority class samples to increase their weights.
- In other words, it re-creates some existing examples in the original minority class.
- Main drawback: it causes overfitting.

SMOTE – Synthetic Minority Oversampling Technique

- N.V. Chawla, K.W. Bowyer, L. O Hall, W.P. Kegelmeyer. "SMOTE: Synthetic Minority Over-sampling Technique", *Journal of Artificial Intelligence Research*, 16:321–357, 2002.
- Choose a random sample x_i from the minority class.
- Find its k nearest neighbors (typically $k = 5$).
- Pick a neighbor x_i^n and **generate a synthetic sample in a random point of the line that connects x_i and x_i^n**:

$$x_{syn} = x_i + \lambda(x_i - x_i^n)$$

- This procedure can be used to create as many synthetic examples for the minority class as are required.

SMOTE – Synthetic Minority Oversampling Technique

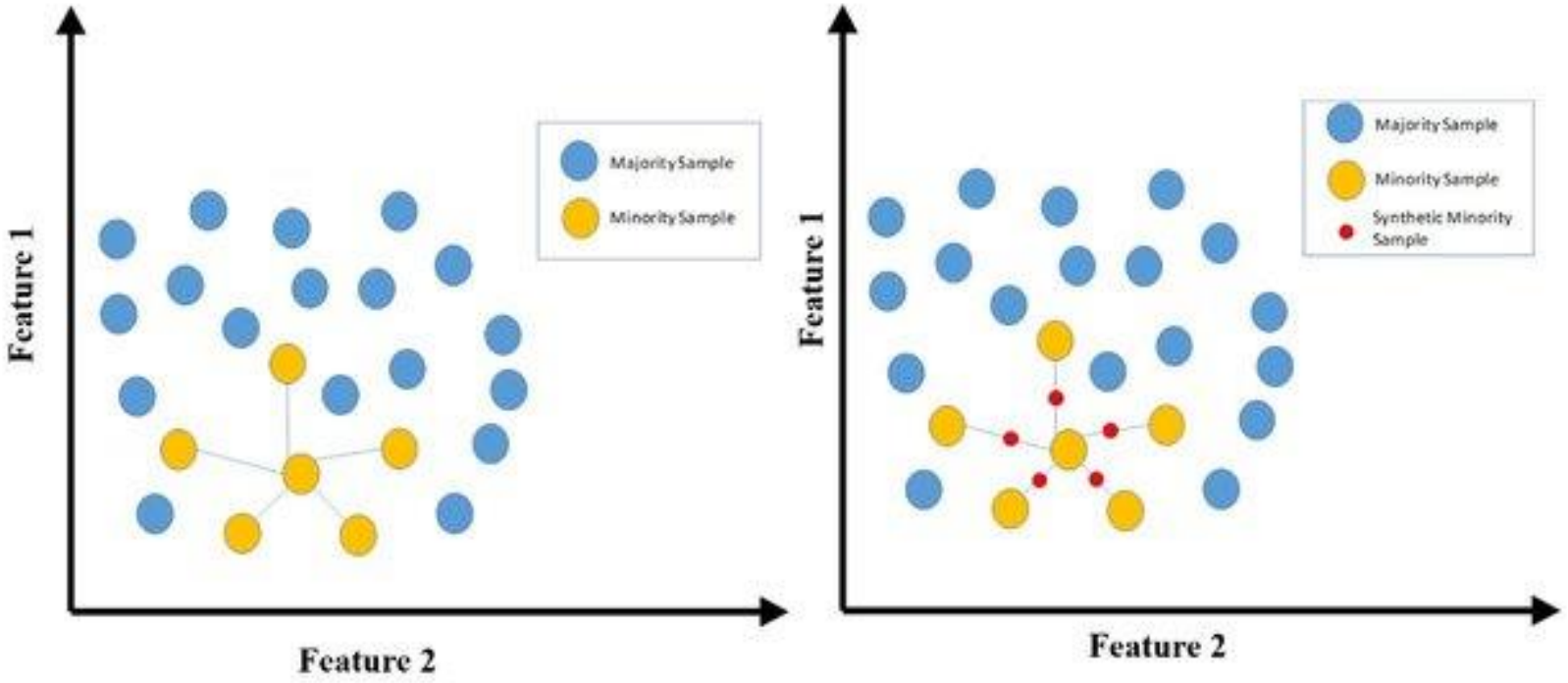


Image Source: Human Knee Abnormality Detection from Imbalanced sEMG Data
https://www.sciencedirect.com/science/article/abs/pii/S174680942100001X?via%3Dihub

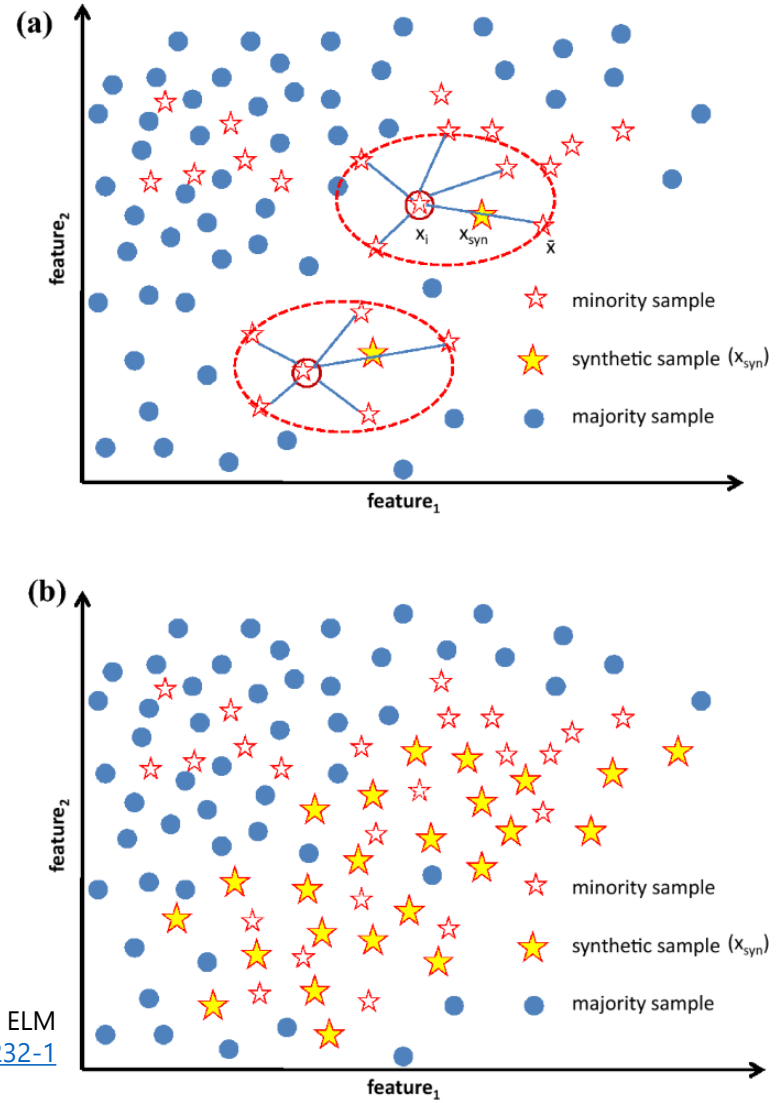


Image Source: Classifying imbalanced data using SMOTE based class-specific kernelized ELM
https://link.springer.com/article/10.1007/s13042-020-01232-1

SMOTE – Synthetic Minority Oversampling Technique

- **Strong point:** SMOTE is effective because the synthetic minority examples are relatively close in feature space to existing examples from the minority class.
- **The class distribution is not affected much.**
- **Weak point:** the synthetic examples are created without considering the majority class.
- There is a significant **possibility of synthesizing ambiguous examples** if there is a strong overlap for the classes.

Borderline SMOTE (1)

- SMOTE does not care about the "nature" of the minority samples to be used for synthetic data generation.
- Therefore, it may even use outliers for synthetic data generation.
  - Outliers: minority class samples inside a large area of majority class samples.

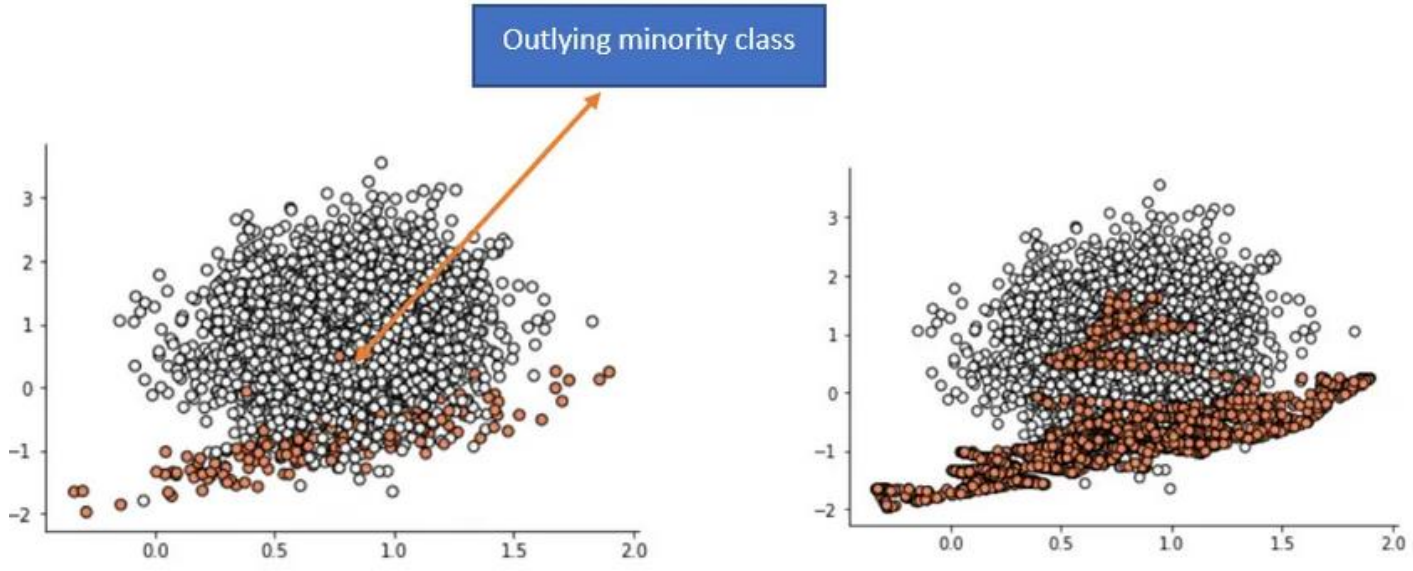
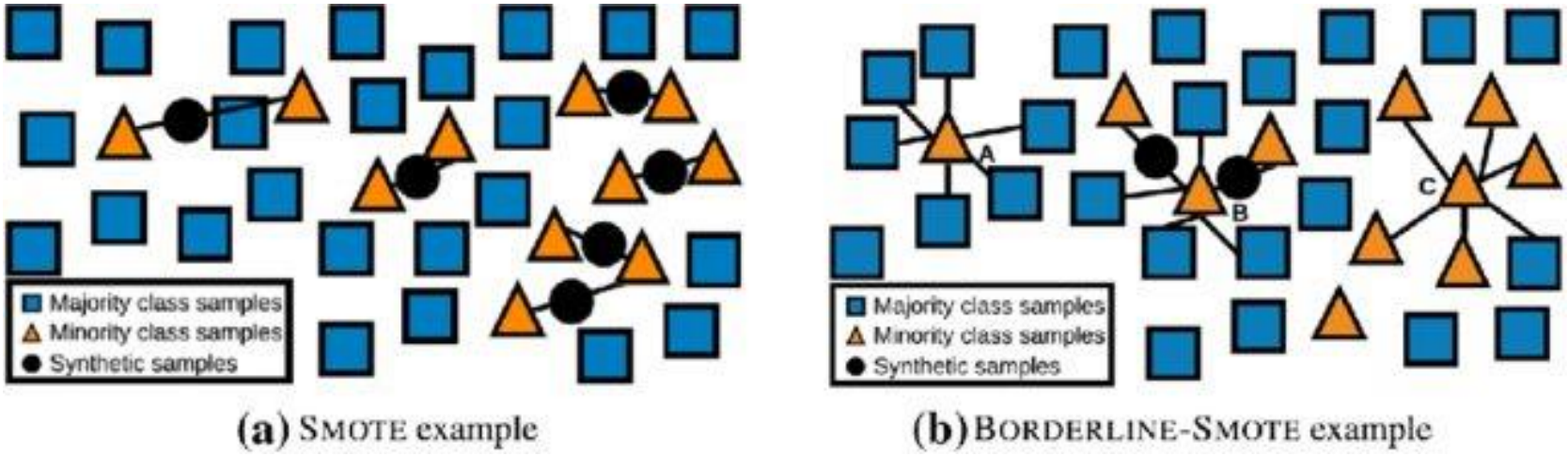


Image Source: Class Imbalance, SMOTE, borderline SMOTE, ADASYN
https://towardsdatascience.com/class-imbalance-smote-borderline-smote-adasyn-6e36c78d804

Borderline SMOTE (2)

- H. Han, Wen-Yuan Wang, Bing-Huan Mao. "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning", *International Conference on Intelligent Computing*, 878–887. Springer, 2005.
- Each sample x_i from the minority class is categorized as:
  - **Noise:** all its nearest-neighbors belong to a different class,
  - **In danger:** at least half of its nearest neighbors belong to the same class,
  - **Safe:** all nearest neighbors are from the same class.
- **Borderline SMOTE uses only samples from the "in danger" group to generate new samples.**
- **It finds their k-nearest neighbors and generates samples over the line that connects them.**
  - **The neighboring points must belong to the same class as the sample "in danger".**

Borderline SMOTE (3)



(a) SMOTE example (b) BORDERLINE-SMOTE example

Image Source: A. Bernardo, E. Della Valle, "VFC-SMOTE: very fast continuous synthetic minority oversampling for evolving data streams", *Data Mining and Knowledge Discovery*, 35:2679–2713, 2021

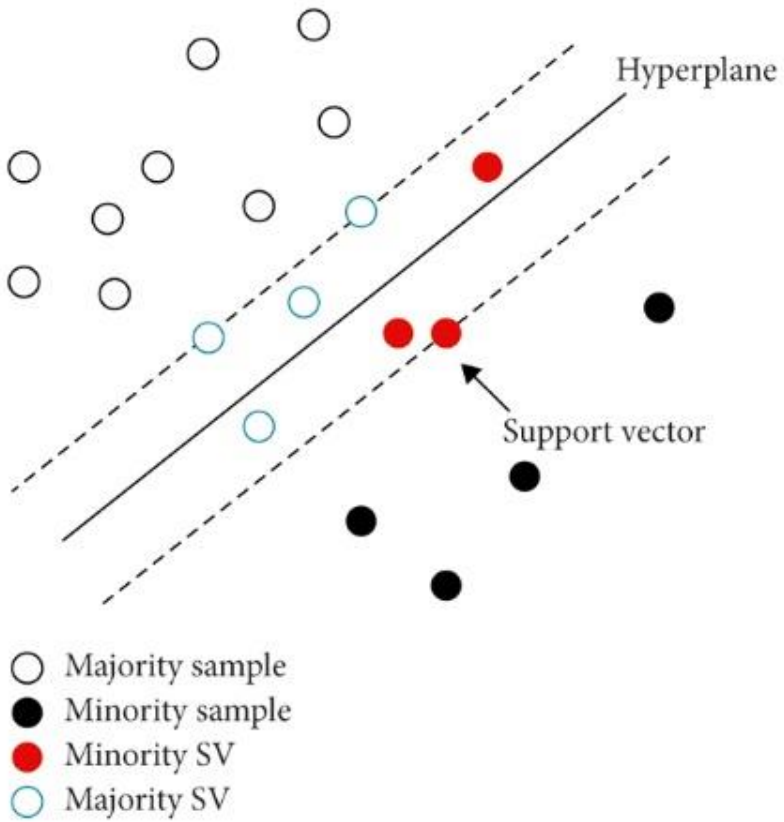
Borderline SMOTE version 2

- There is a number of variations of Borderline SMOTE.
- **Borderline SMOTE uses only samples from the "in danger" group to generate new samples.**
- **A variant method finds the k-nearest neighbors and generates samples over the line that connects them.**
  - **k-nearest neighbors from ANY class.**

SMOTE-SVM (1)

- H.M. Nguyen, E.W. Cooper, K. Kamei. "Borderline Over-Sampling for Imbalanced Data Classification", *International Journal of Knowledge Engineering and Soft Data Paradigms*, 3(1): 4–21, 2011.
- Instead of using the k-nearest neighbors algorithm to generate synthetic samples, this technique uses an SVM classifier.
- The SVM classifier is trained on the original training set.
- **Synthetic data will be randomly created along the lines joining each minority class support vector with a number of its nearest neighbors.**
- More data points are synthesized away from the region of class overlap. SMOTE-SVM focuses more on the region where the data is separated.

SMOTE-SVM (2)


Hyperplane
Support vector
Majority sample
Minority sample
Minority SV
Majority SV

k-Means SMOTE (1)

- G. Douzas, F. Bacao, F. Last, "Improving imbalanced learning through a heuristic oversampling method based on k-Means and SMOTE", *Information Sciences*, 465: 1-20, 2018.
- k-Means SMOTE employs the k-Means clustering algorithm in conjunction with SMOTE to rebalance the dataset.
- **It generates instances according to the cluster density.**
- **In this way it confronts the problem of intra-class imbalance.**
- **This method avoids the generation of noise in the safe area.**

k-Means SMOTE (2)

- Clustering step: Cluster the instances into k groups by using k-Means.
- Filtering step: Select the clusters to be oversampled and determine the number of instances to be generated in each cluster.
  - The cluster selection is based on the proportion of minority and majority instances in each cluster.
  - Any cluster with at least 50% of minority instances can be selected for oversampling.
  - Weights are assigned to the selected clusters to determine the instances to be created.
- Oversampling step: Use SMOTE to oversample each selected cluster.
  - Choose a random minority instance x_i within a cluster.
  - Find one of its random minority neighbors, x_i^n.
  - Determine a new instance x_{syn} by interpolating x_i and x_i^n.
  - The process is repeated until the desired number of minority instances is created.

ADASYN – Adaptive Synthetic Sampling (1)

- H. He, Y. Bai, E.A. Garcia, S. Li. "ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning". In *2008 IEEE International Joint Conference on Neural Networks*, 1322–1328. IEEE, 2008.
- ADASYN algorithm decides the number of synthetic examples that need to be generated for each minority example by the amount of its majority nearest neighbors.
- **The more majority nearest neighbors, the more synthetic examples will be created.**
- ADASYN uses a density distribution Γ_i as a criterion to decide the number of synthetic examples.
  - In contrast, in SMOTE, each minority sample has an equal chance of being selected for the synthetic process.

ADASYN – Adaptive Synthetic Sampling (2)

- G: The total number of synthetic examples to be created:

$$G = \beta\left(\left|S_{maj} - S_{min}\right|\right), \qquad \beta \in [0,1]$$

- For each minority instance x_i, find the k-nearest neighbors.
- Define the weight of x_i for synthetic process, as the density distribution:

$$\Gamma_i = \frac{\Delta_i / k}{Z}$$

- Δ_i: the number of majority examples in the k-nearest neighborhood of x_i,
- Z: a normalized constant so that Γ_i is a probability mass function, i.e., $\sum \Gamma_i = 1$
- The number of synthetic examples to be generated, for x_i:

$$g_i = G \cdot \Gamma_i$$

- Finally, generate g_i synthetic examples for each x_i by using SMOTE.

ADASYN – Adaptive Synthetic Sampling (2)

Algorithm 2 ADASYN(X, β, K)

Input:

X: the original training set

$\beta \in [0, 1]$: desired balanced level

K: number of nearest neighbors

Output: the oversampled training set

S_{maj} majority class S_{min} minority class

n_{maj} # of majority observations n_{min} # of minority observations

$G \leftarrow (n_{maj} - n_{min}) \times \beta$

$r_{1 \times n_{min}}$ percentage of nearest neighbors in majority class

for $i \leftarrow 1$ to n_{min} **do**

for each i, compute k nearest neighbors and store the indices in the nn

$r[i] \leftarrow \frac{|nn[i] \cap S_{maj}|}{K}$

for $i \leftarrow 1$ to n_{min} **do**

$\hat{r}[i] \leftarrow \frac{r[i]}{\sum_i r[i]}$

$g[i] \leftarrow int(\hat{r}[i] \times G)$

$Syn_{(G) \times m} \leftarrow$ empty array for synesthetic samples

$j = 1$

for $i \leftarrow 1$ to n_{min} **do**

$newindex \leftarrow g[i]$

while $newindex \neq 0$ **do**

$K_c \leftarrow$ random number between 1 and K

$diff \leftarrow= S_{min}[nn[i][K_c]] - S_{min}[i]$

$Syn[j][] \leftarrow S_{min}[i][] + diff \times uniform(0, 1)$

$j+ = 1$

$n- = 1$

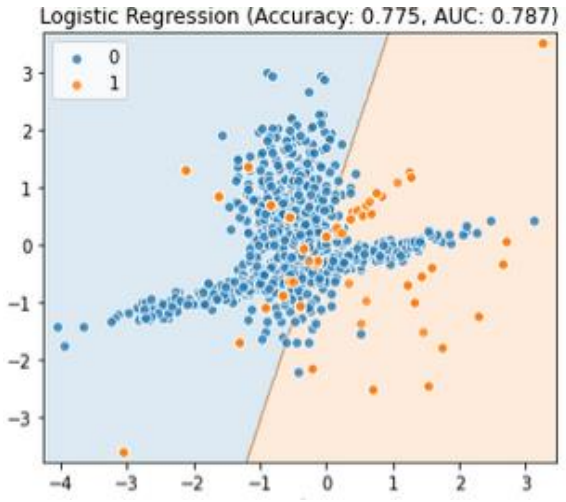
end while

end for

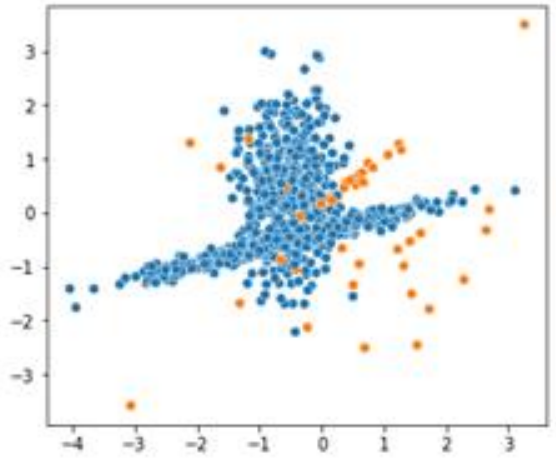
Return *Dataframe (X & Syn)*

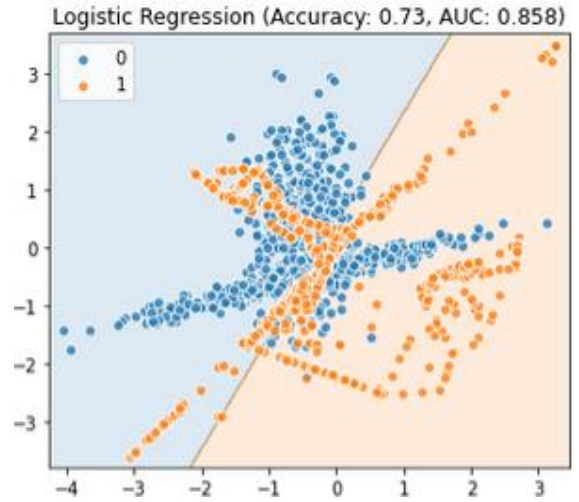
PJ Huang, "Classication of Imbalanced Data Using Synthetic Over-Sampling Techniques", Master Thesis, University of California, Los Angeles, 2015.

Comparison (classification with LogisticRegression)

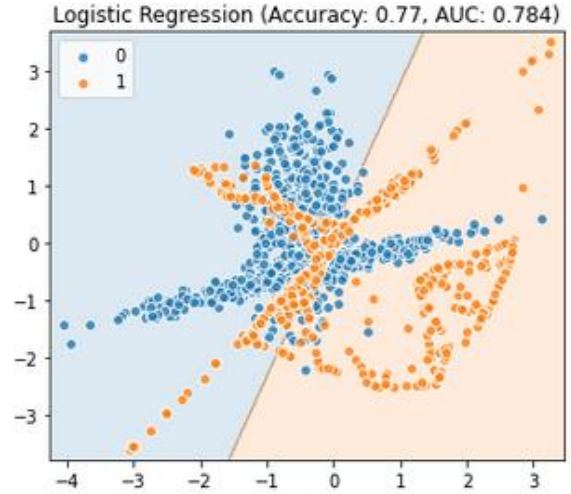


Random oversampling

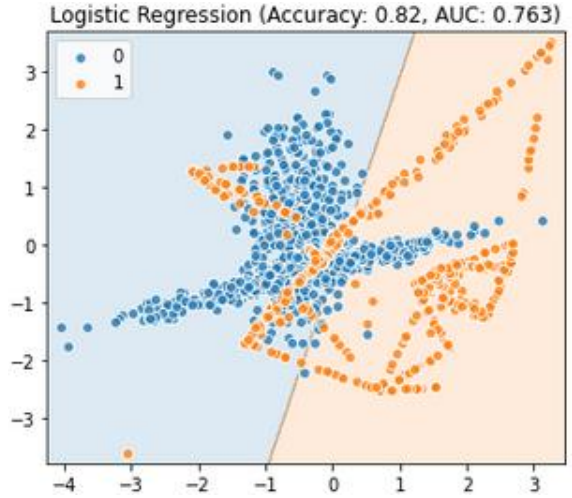




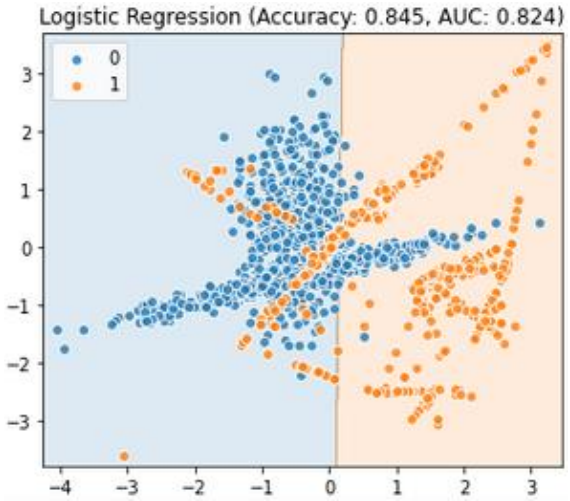
ADASYN



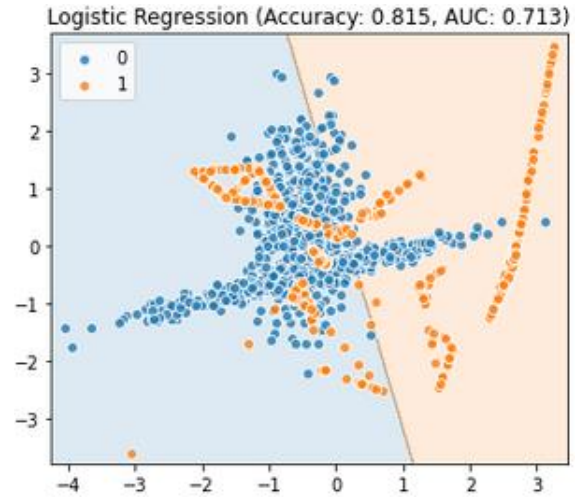
SMOTE



Borderline SMOTE

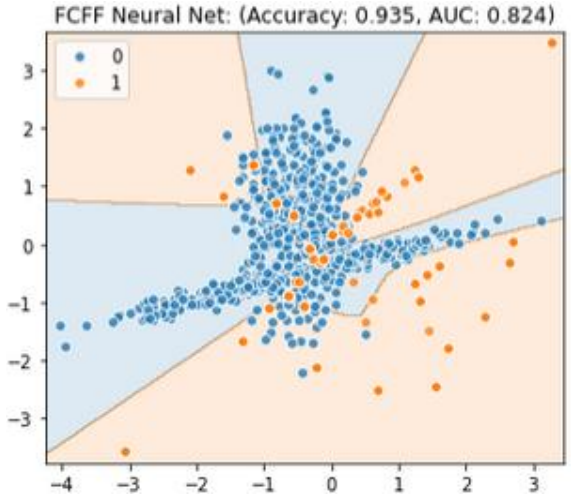


SMOTE-SVM

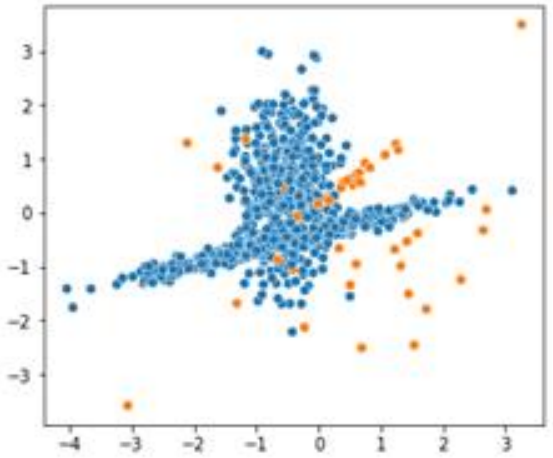


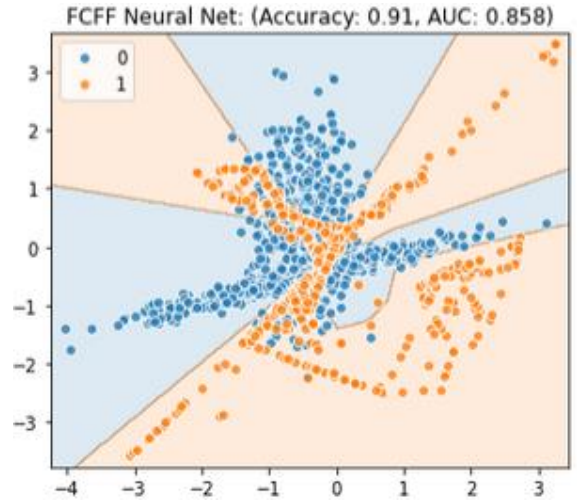
K-Means SMOTE

Comparison (classification with Neural Nets)

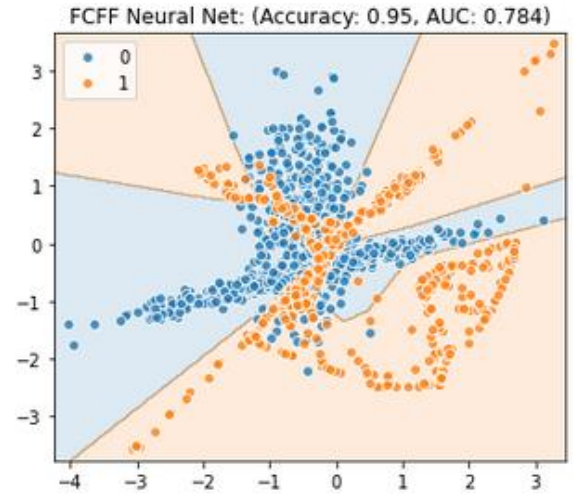


Random oversampling

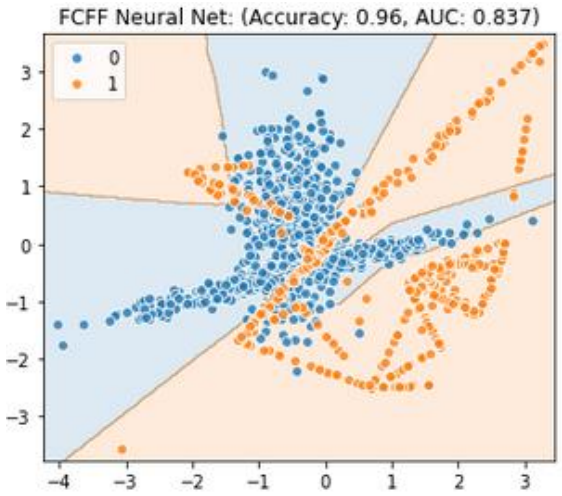




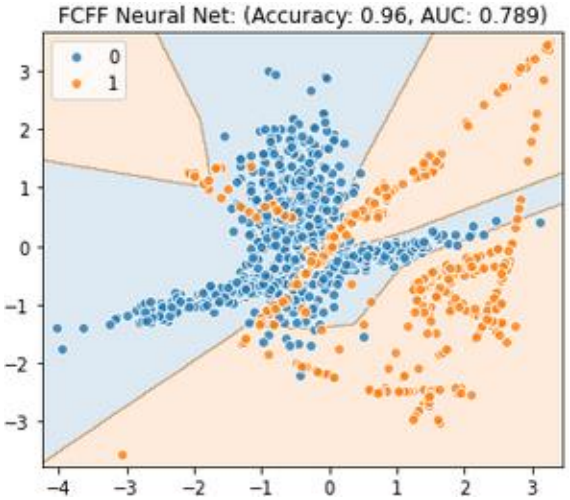
ADASYN



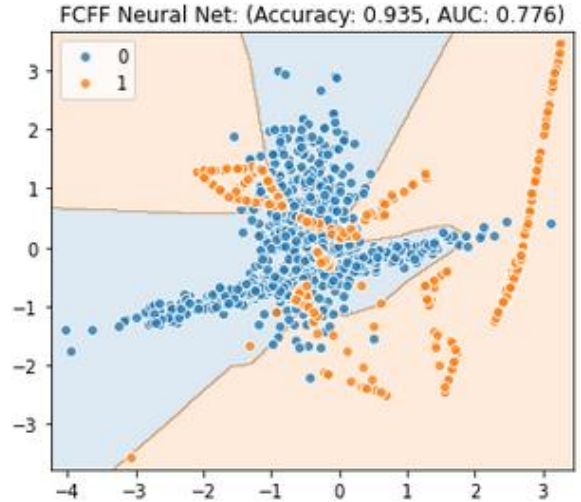
SMOTE



Borderline SMOTE



SMOTE-SVM



K-Means SMOTE

Generative deep learning models

- Generative modeling is an unsupervised learning task that automatically discovers patterns in the input data.
- In this way the model can generate new examples that plausibly could have been drawn from the original dataset.
- Deep Generative Models:
  - Variational Autoencoders (VAEs).
  - Generative Adversarial Networks (GANs).
  - Deep Belief Networks (DBNs).
  - Generative Transformers.
  - Restricted Boltzmann Machines (RBMs).

Autoencoders

- The Autoencoder (AE) is topologically identical to a standard, fully-connected feed-forward neural network.
- It is an unsupervised learning model, trained to reproduce its input $\mathbf{x}$ as accurately as possible.
- If $\hat{\mathbf{x}}$ is the output of the network, then the Autoencoder attempts to minimize the reconstruction error $L(\mathbf{x}, \hat{\mathbf{x}})$ that quantifies the distance between $\mathbf{x}$ and $\hat{\mathbf{x}}$.
  - The encoder part compresses the data from the initial space to the **latent space.**
  - The decoder part decompresses it.

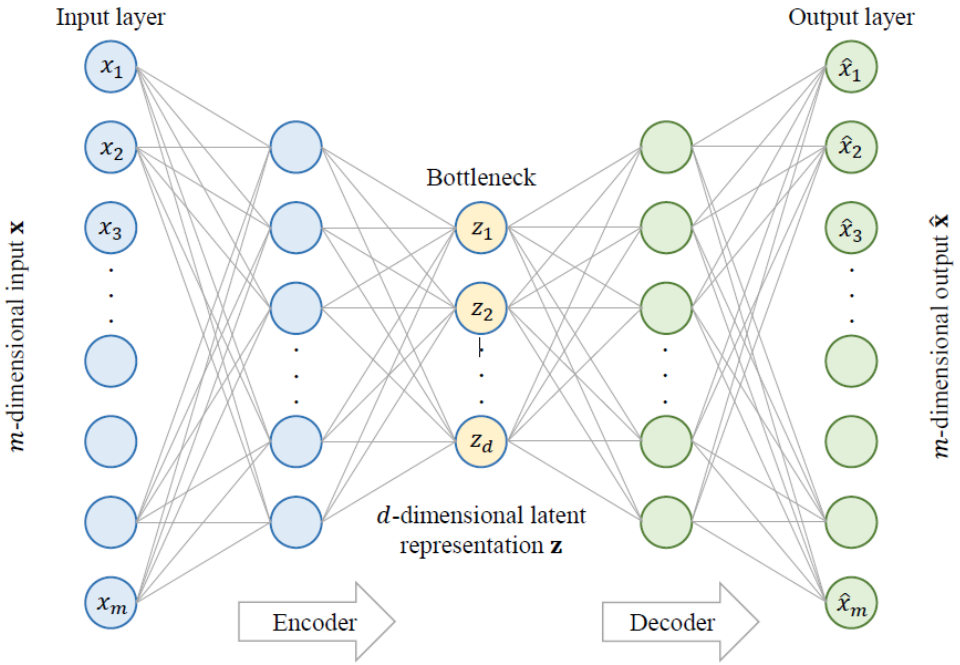


Image Source: L. Akritidis, P. Bozanis, "Low Dimensional Text Representations for Sentiment Analysis NLP Tasks", Springer SN Computer Science, 2023.

Autoencoders are not generative

- The latent space **z** produced by the Autoencoder is sparse.
- It is difficult to predict the distribution of values in that space.
- Finding a latent value for which the decoder will produce meaningful data is almost impossible.
- The architecture lacks the necessary knowledge to generate meaningful samples.

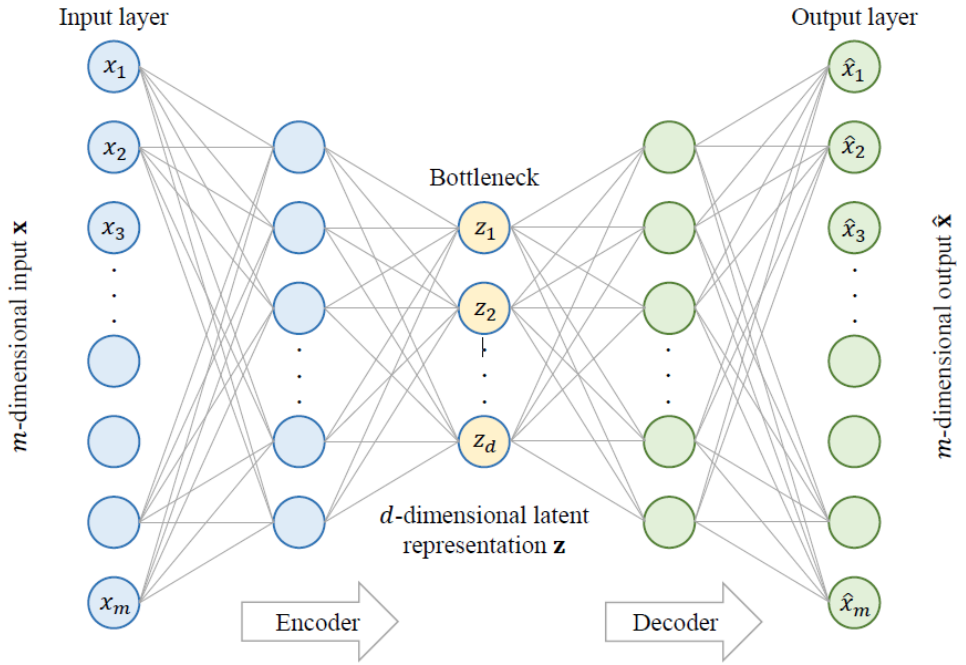


Image Source: L. Akritidis, P. Bozanis, "Low Dimensional Text Representations for Sentiment Analysis NLP Tasks", Springer SN Computer Science, 2023.

Variational Autoencoders (VAEs, 1)

- VAE is an Autoencoder whose encoded distribution is regularized during training to ensure that the latent space has good properties.
- These properties will allow us to generate meaningful data.
- The term "variational" comes from the **variational inference** statistical method that allows the estimation of a probability distribution.
- **VAEs encode an input sample as a probability distribution over the latent space (instead of encoding it as a single point like AEs)**.

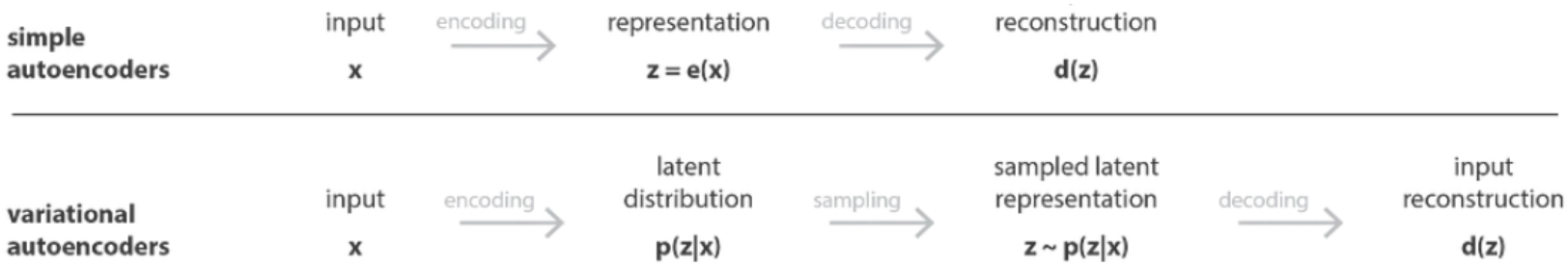


Image Source: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Variational Autoencoders (VAEs, 2)

- In practice, the encoded distributions are chosen to be normal.
- Therefore, an input sample $\mathbf{x}$ is mapped to a mean vector $\boldsymbol{\mu}(\mathbf{x})$ and a vector of standard deviations $\boldsymbol{\sigma}(\mathbf{x})$.
- $\boldsymbol{\mu}(\mathbf{x})$ and $\boldsymbol{\sigma}(\mathbf{x})$ parametrize a diagonal Gaussian distribution $N(\boldsymbol{\mu}, \boldsymbol{\sigma})$, from which we then sample a latent vector $\mathbf{z} \sim N(\boldsymbol{\mu}, \boldsymbol{\sigma})$.

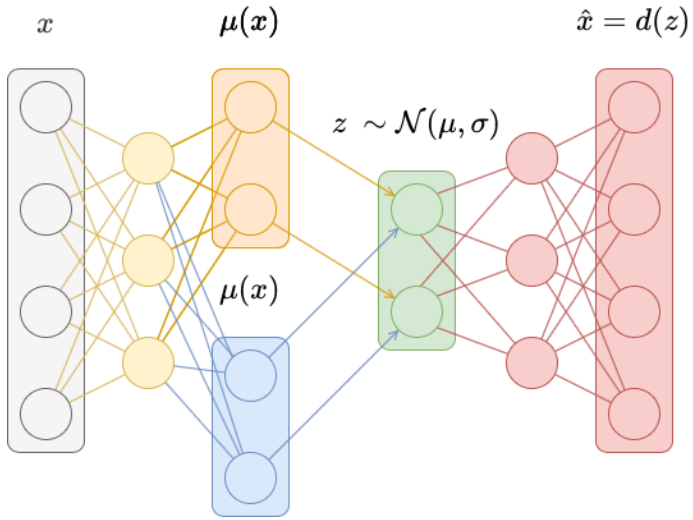


Image Source: https://avandekleut.github.io/vae/

Variational Autoencoders (VAEs, 3)

- The distributions returned by the encoder are enforced to be close to a standard normal distribution.
- VAEs produce latent spaces that are more compact and smooth than those learned by traditional autoencoders.

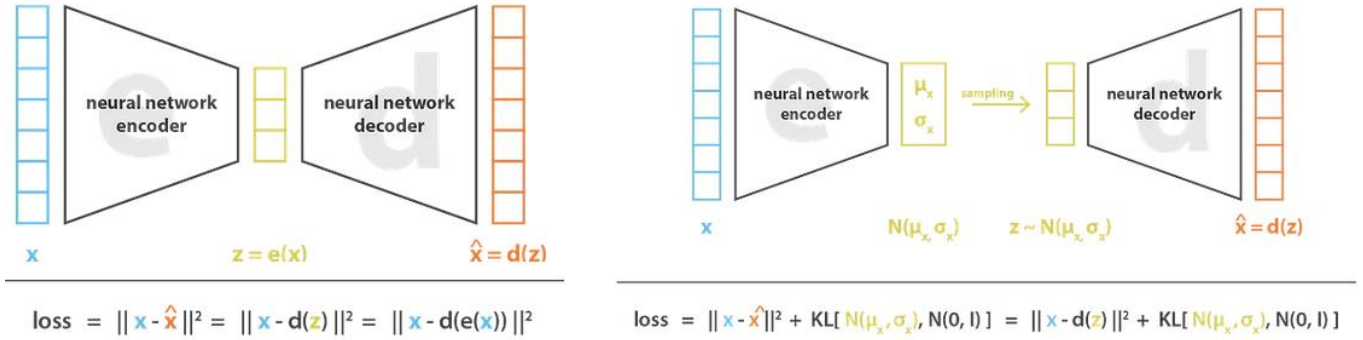


Image Source: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Generative Adversarial Networks (GANs)

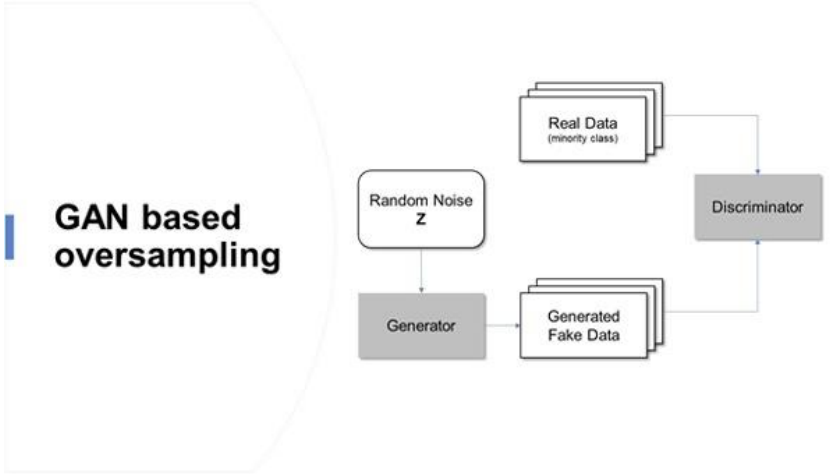
- Generative Adversarial Networks (GANs) are an approach to generative modeling using deep learning methods, such as Convolutional Neural Networks.
- Inside a GAN, two Neural Networks compete against each other to learn the target distribution and generate artificial data:
- **A discriminator network D receives the artificial training examples from G and tries to understand whether they are fraud or not.**
- **A generator network G generates training samples with the aim of fooling the discriminator D.**

GAN: Discriminator network D

- D is a classifier that receives data provided by the generator G.
- Its goal is to correctly identify if the input data is fake or real.
- The training process uses:
  - real data instances used as positive examples, and
  - fake data instances (coming from G), used as negative examples.
- The employed loss function must penalize the misclassifications (real data identified as fake, or fake data identified as real).
- At each iteration, D updates its parameters using backpropagation, thus improving its ability to identify fake data instances.

GAN: Generator network G

- G learns to create fake data by using the output of D.
- Its goal is to cause the discriminator D to classify its output as real.
- G is trained simultaneously with D. The training involves the following phases:
  - Construct an initial random noise sample and use it to produce the output of G.
  - Feed the output of G to the discriminator D.
  - Compute the discriminator loss.
  - Backpropagate using both G and D to compute the gradients.
  - Use these gradients to update only the generator's weights.

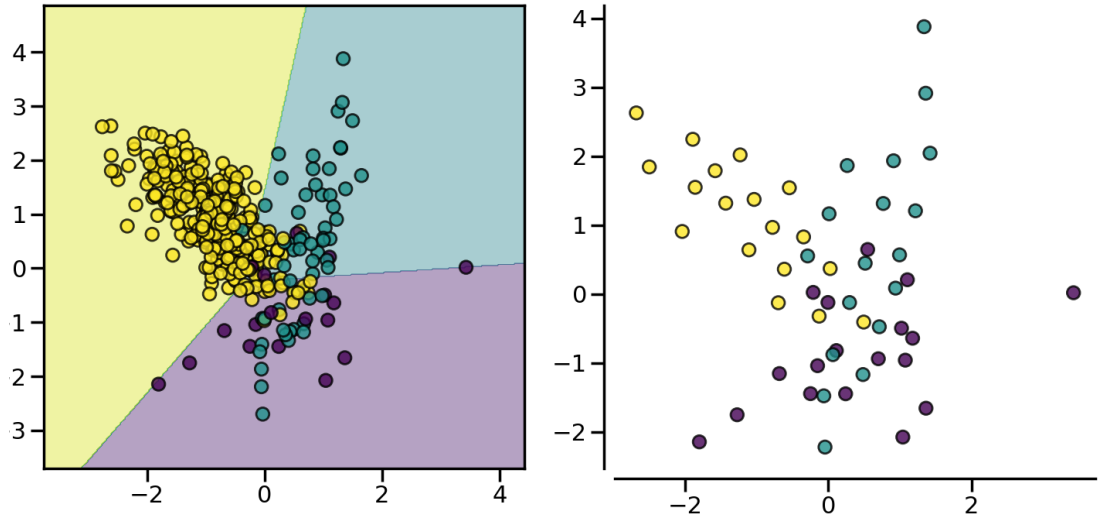


Generative Adversarial Networks (GANs)

- Standard: GAN, DCGAN.
- Conditional: cGAN, SS-GAN, InfoGAN, ACGAN.
- Loss: WGAN, WGAN-GP, LSGAN.
- Image Translation: Pix2Pix, CycleGAN.
- Advanced GANs: BigGAN, PG-GAN, StyleGAN.
- Other: StackGAN, 3DGAN, BEGAN, SRGAN, DiscoGAN, SEGAN.
- V. Sampath, I. Maurtua, J.J. Aguilar Martin, A. Gutierrez, "A survey on Generative Adversarial Networks for imbalance problems in computer vision tasks", *Journal of Big Data* 8:1–59, 2021.

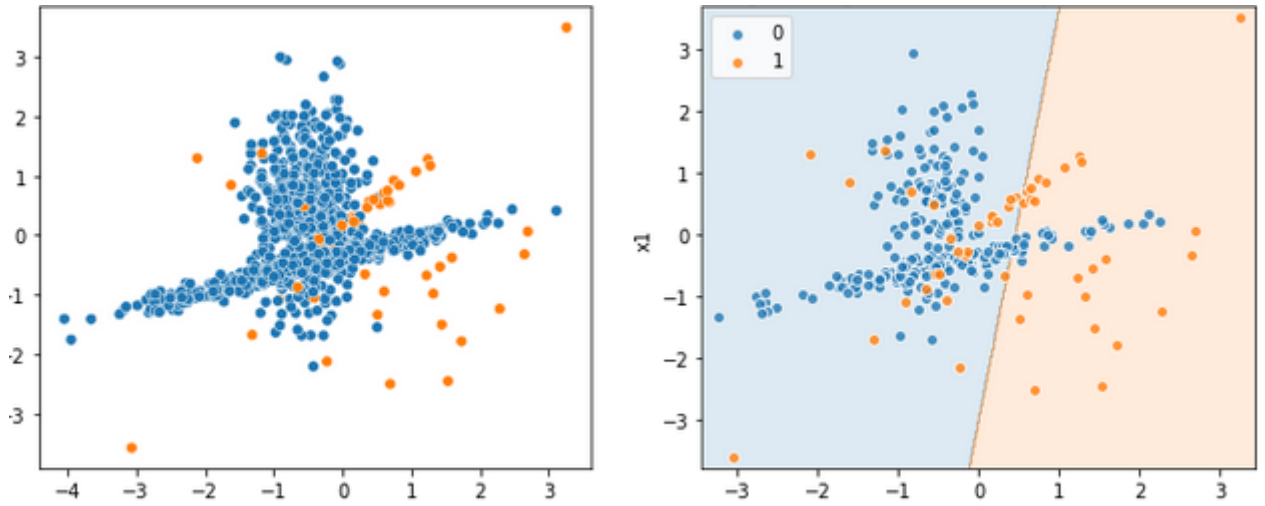
Undersampling

- A preprocessing technique that brings balance to a dataset by reducing the population of the majority class.
- Two approaches:
- **Prototype selection:** Selects the samples to be removed from the targeted classes.
- **Prototype generation:** Reduces the number of samples in the targeted classes; the remaining samples are generated -not selected- from the original set.



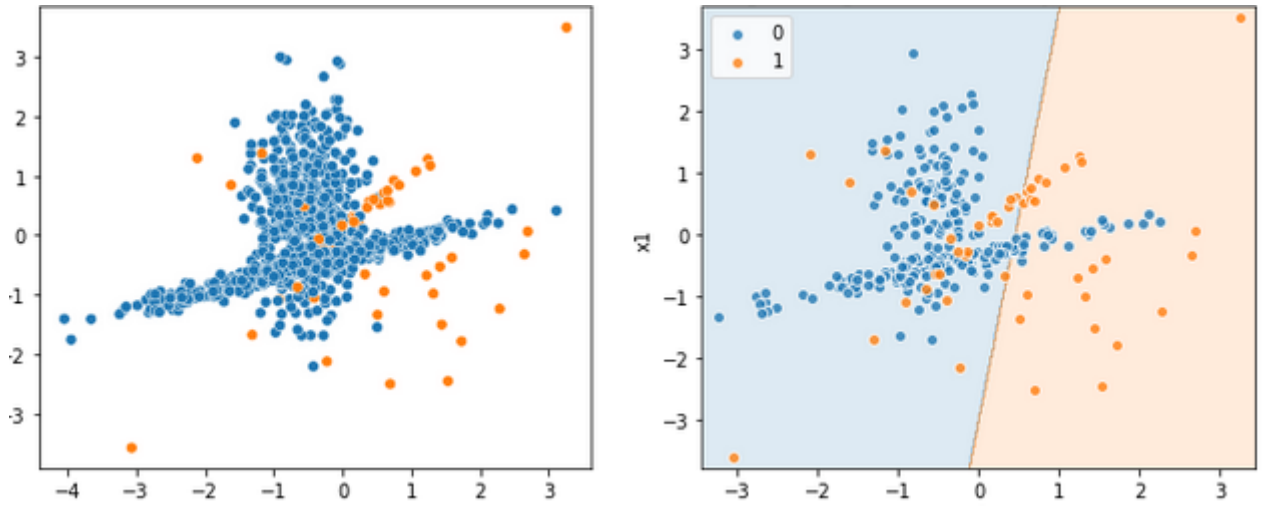
Random Undersampling

- The simplest Prototype Selection undersampling technique.
- Randomly selects a subset of data for the targeted classes until the desired imbalance ratio is achieved.



Random Undersampling

- The simplest Prototype Selection undersampling technique.
- Randomly selects a subset of data for the targeted classes until the desired imbalance ratio is achieved.



NearMiss

- I. Mani, I. Zhang. "kNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction". In *Proceedings of Workshop on Learning from Imbalanced Datasets*, *ICML* 126:1-7, 2003.
- A set of heuristic rules based on nearest neighbors algorithm.
- Let *positive samples* be the samples belonging to the majority class (to be under-sampled).
- *Negative sample* refers to the samples from the minority class (i.e., the most under-represented class).

NearMiss-1

- NearMiss-1 selects the positive samples for which the average distance to the k closest samples of the negative class is the smallest.

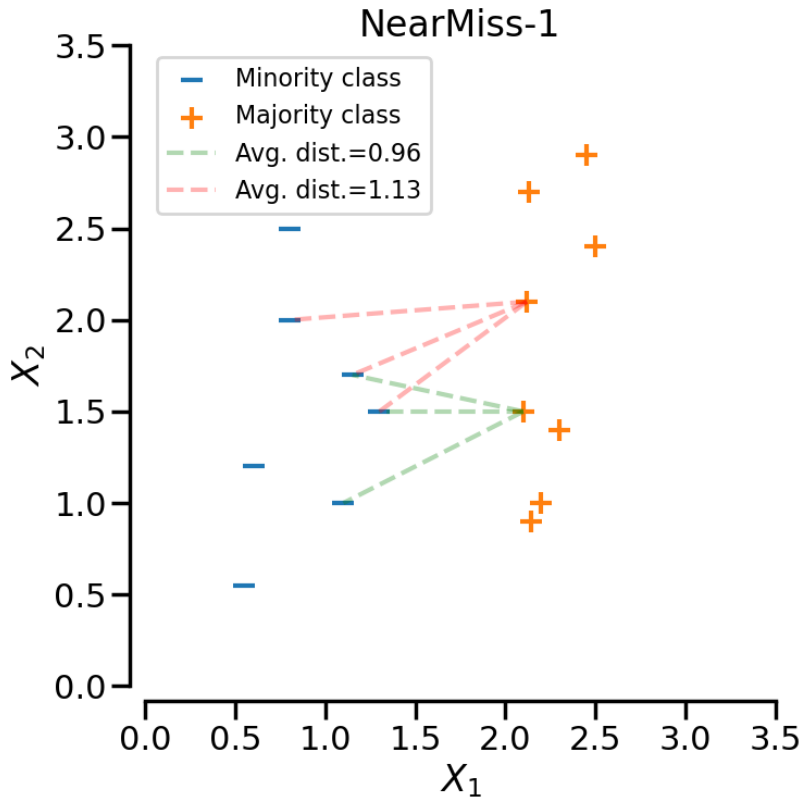


Image Source: https://imbalanced-learn.org/stable/under_sampling.html#prototype-selection

NearMiss-2

- NearMiss-2 selects the positive samples for which the average distance to the k farthest samples of the negative class is the smallest.

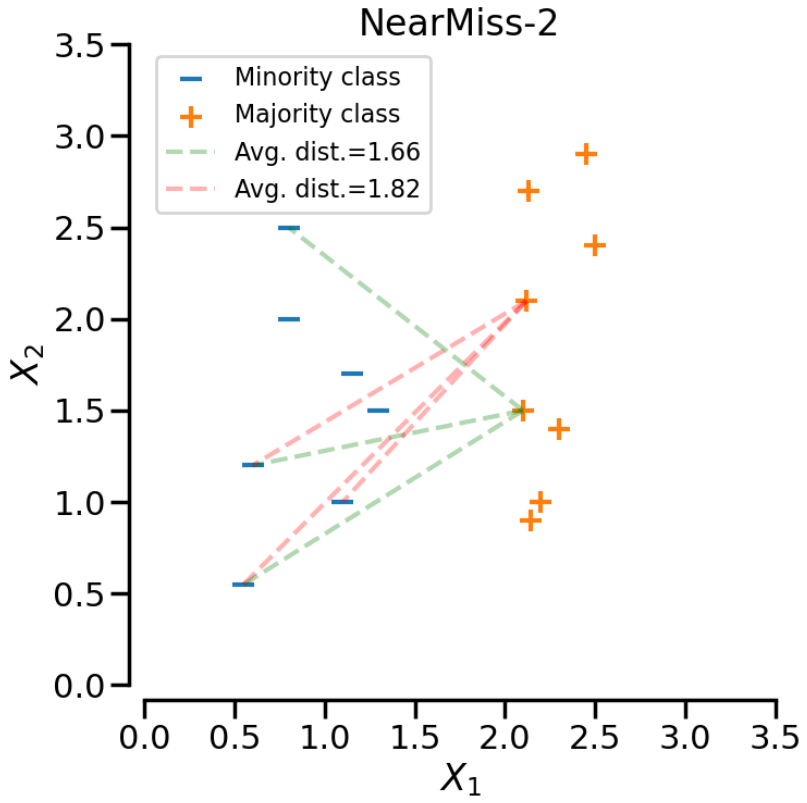


Image Source: https://imbalanced-learn.org/stable/under_sampling.html#prototype-selection

NearMiss-3

- NearMiss-3 is a 2-step algorithm.
- Step 1: identify the k-nearest neighbors of the negative samples.
- Step 2: select the positive samples for which the average distance to the nearest-neighbors is the largest.

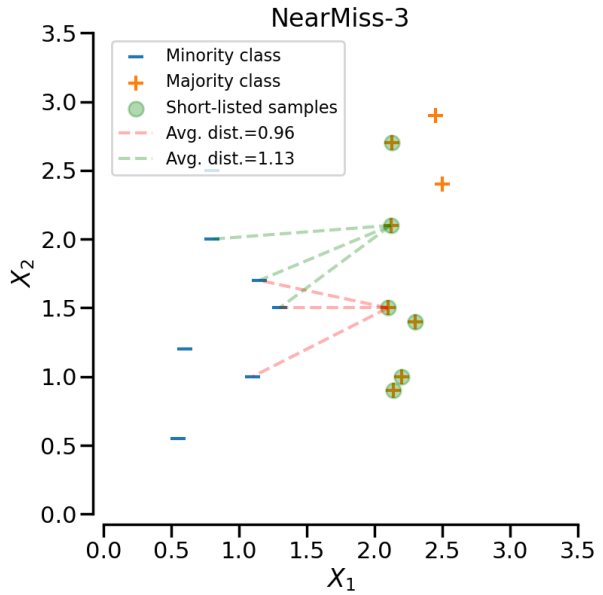


Image Source: https://imbalanced-learn.org/stable/under_sampling.html#prototype-selection

NearMiss comparison

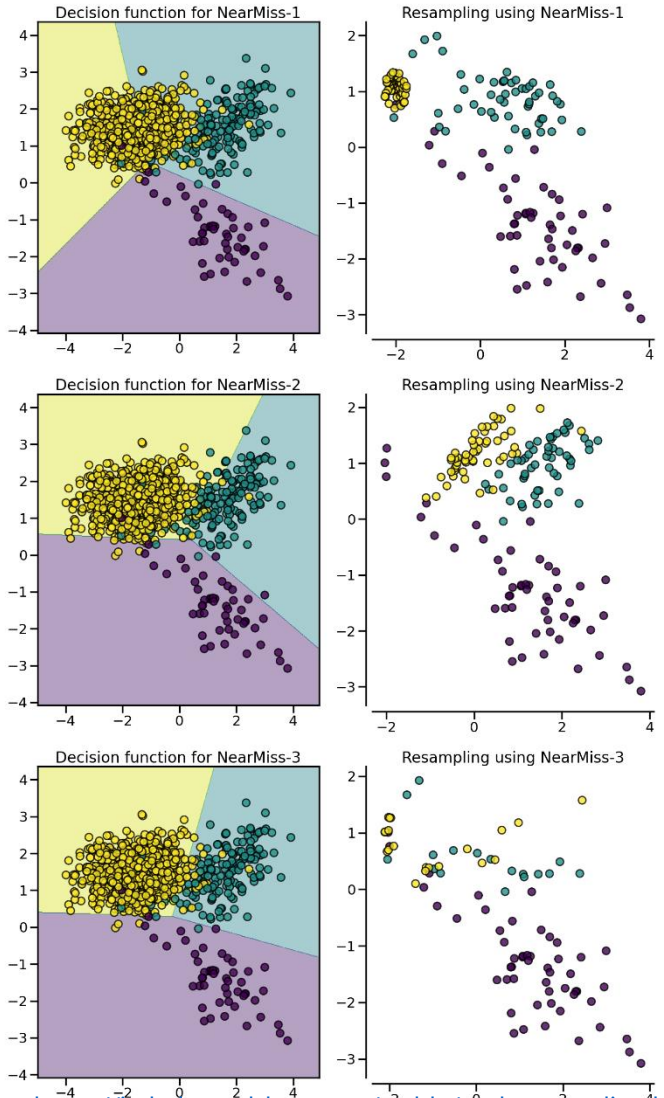


Image Source: https://imbalanced-learn.org/stable/under_sampling.html#prototype-selection

Undersampling with Clustering (1)

- Clustering is another effective way for undersampling the majority class.
- An entire cluster of elements can be effectively replaced by a single representative element, e.g., cluster center, centroid, clustroid, etc.
- Lin et al. proposed two clustering strategies for undersampling, both based on the well-known k-Means algorithm.
- The number of clusters to be constructed is determined by the population of the minority samples.
- In the first strategy, the centroids are used to replace the majority class.
- The second strategy uses the nearest neighbor of each centroid.

Undersampling with Clustering (2)

- Lin, W.C., Tsai, C.F., Hu, Y.H., Jhang, J.S.: "Clustering-based undersampling in class imbalanced data", *Information Sciences* 409:17–26, 2017.

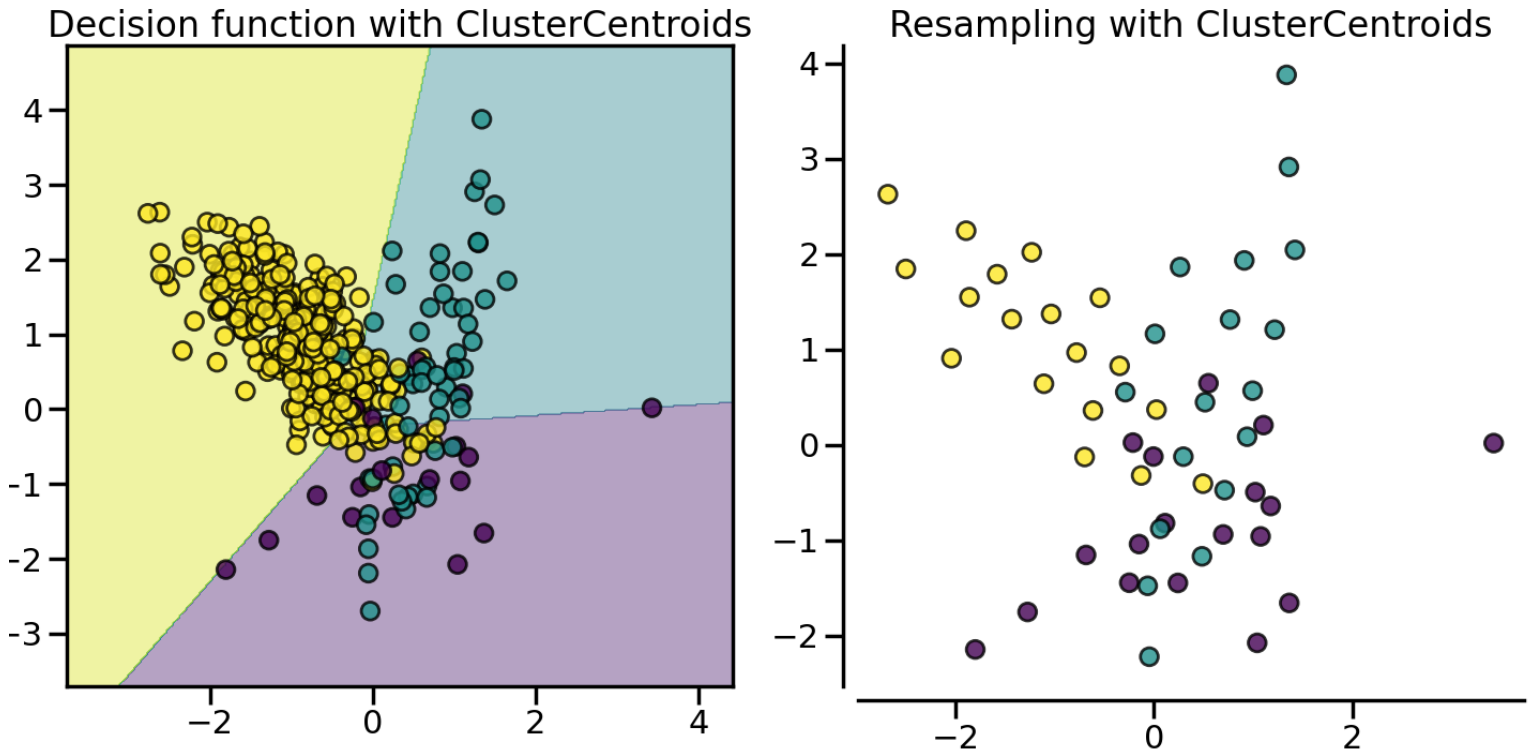
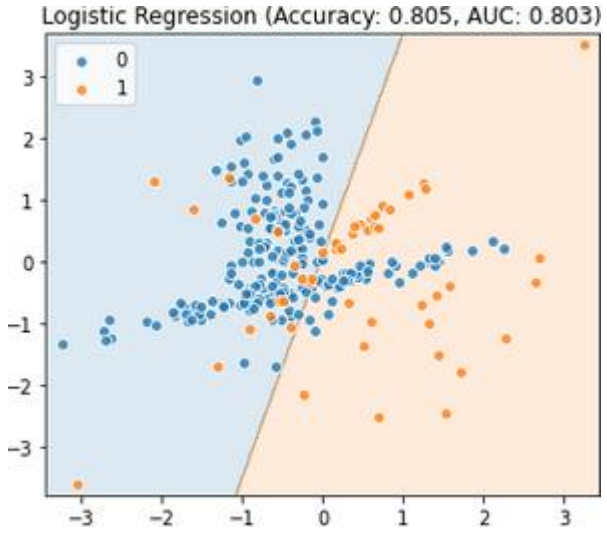
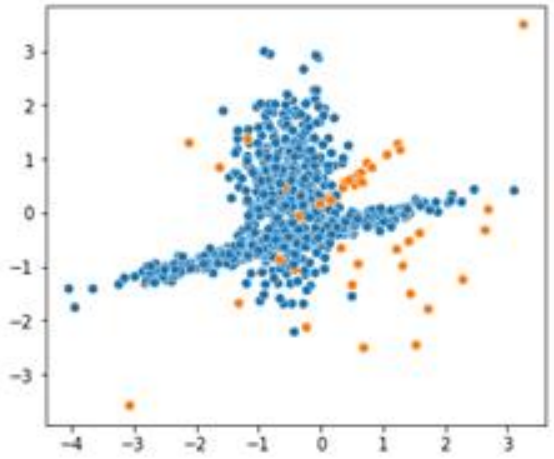


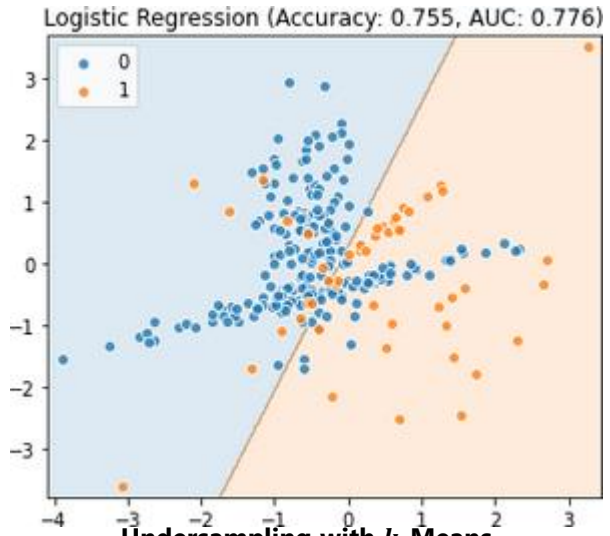
Image Source: https://imbalanced-learn.org/stable/under_sampling.html#prototype-selection

Comparison (classification with LogisticRegression)

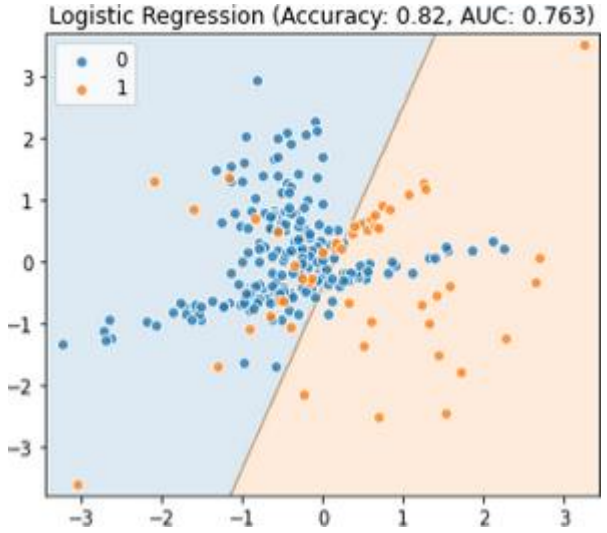


Random undersampling

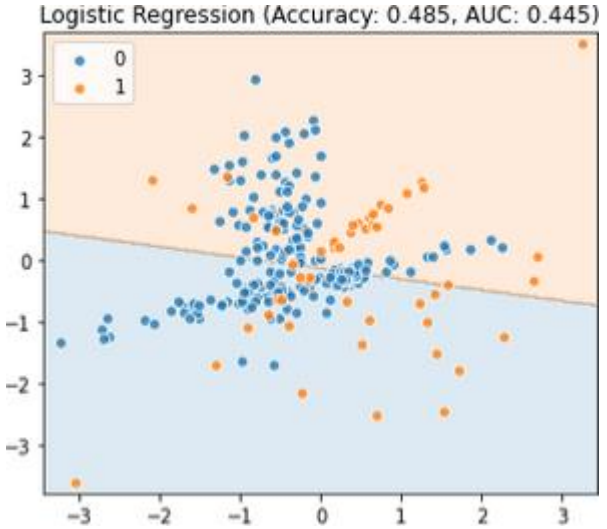




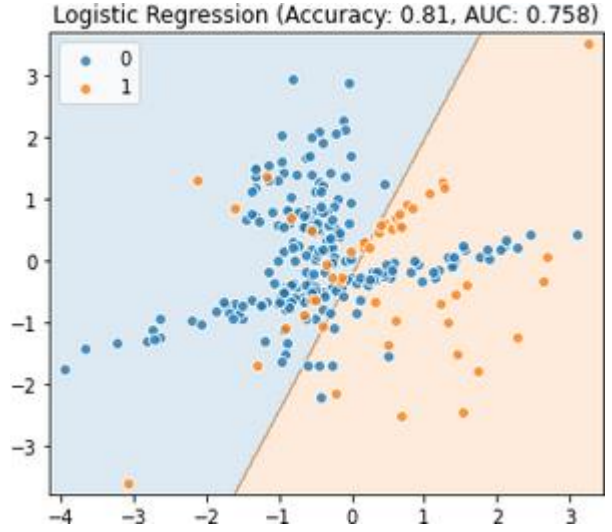
Undersampling with k-Means



Undersampling with NearMiss-1

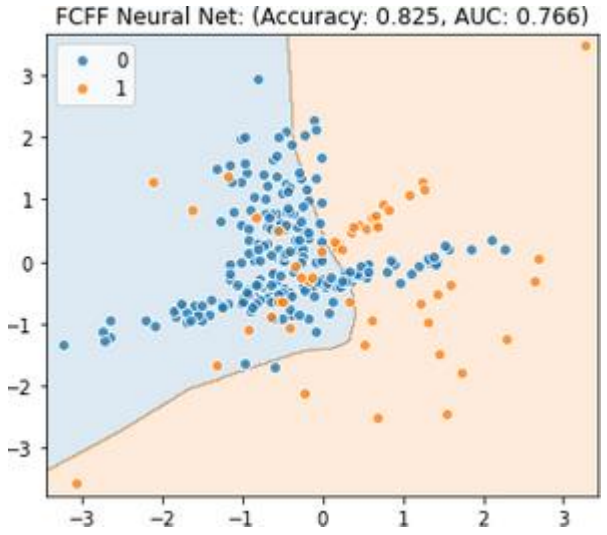


Undersampling with NearMiss-2

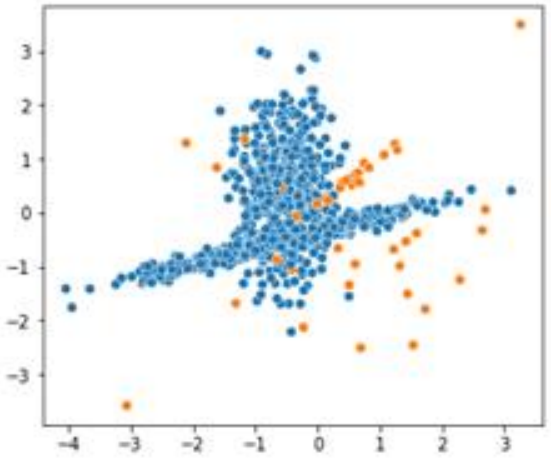


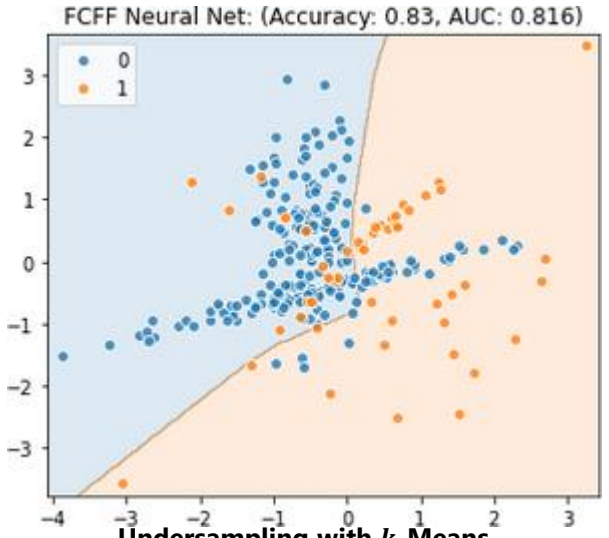
Undersampling with NearMiss-3

Comparison (classification with Neural Nets)

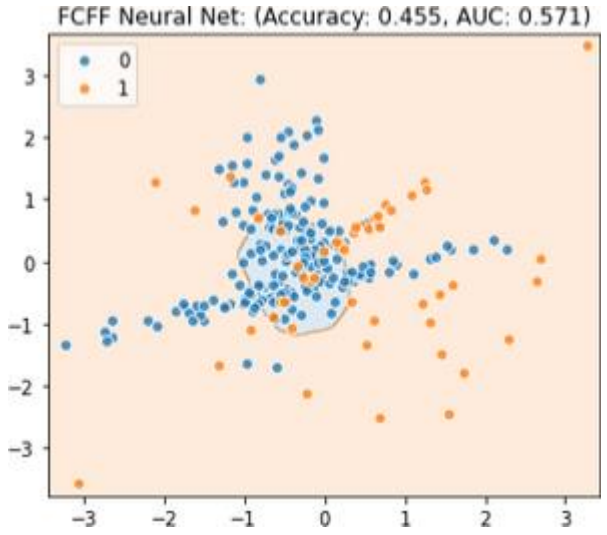


Random undersampling

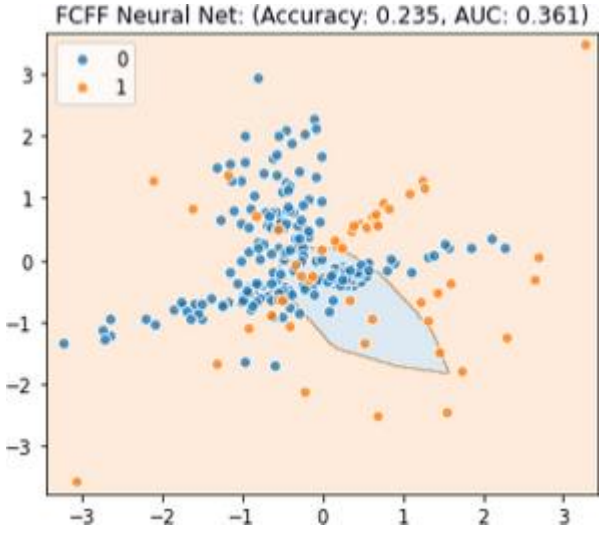




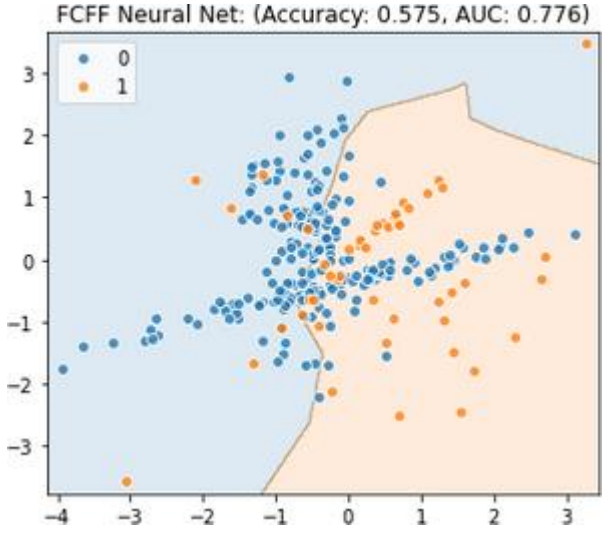
Undersampling with k-Means



Undersampling with NearMiss-1



Undersampling with NearMiss-2



Undersampling with NearMiss-3

Undersampling with data cleaning methods

- There a third family of undersampling techniques that employ well-established data cleaning methods.
  - All of them are based on the k-Nearest Neighbors algorithm.
  - Edited Nearest Neighbor (ENN).
  - Repeated Edited Nearest Neighbor.
  - Condensed Nearest Neighbor (CNN).
  - All kNN.
  - Tomek's Links.
- These methods are not covered in this tutorial, mainly because they do not support the determination of the imbalance ratio in the output dataset.

Hybrid sampling (1)

- In several cases, oversampling has infinitesimal impact on the performance of a classifier. It may also distort the probability distribution of the minority classes.
- On the other hand, the undersampling methods can be detrimental, as they may discard important samples from the majority class.
- **Hybrid sampling combines both oversampling and undersampling to mitigate class imbalance.**

Hybrid sampling (2)

- Three research questions:
- Q1: Which re-sampling method do we apply first? Two flavors:
  - First shrink the majority class with undersampling, then enlarge the minority classes by applying oversampling.
  - First enlarge the minority classes by applying oversampling, then shrink the majority class with undersampling.
- Q2: Which oversampling and which undersampling technique should we apply?
  - SMOTE, SMOTE variants, GANs, clustering, random undersampling, NearMiss?
- Q3: To what extent should we shrink the majority class and to what extent should we enlarge the minority classes?

Two hybrid sampling Mega-Tests (1)

- $IR = {}^{\text{minority\_samples}}/_{\text{majority\_samples}}$
- For each undersampling technique u in [Random Undersampling, NearMiss-1, NearMiss-2, NearMiss-3, Undersampling with Clustering]:
  - For each imbalance_ratio r in the range [initial_ratio, 1.0]:
    - initialize scenario $u(r)$
    - Undersamplers $U \leftarrow U \cup u(r)$
- For each oversampling technique o in [Random Oversampling, SMOTE, BorderlineSMOTE, SMOTE-SVM, k-Means SMOTE, ADASYN]:
  - For each imbalance_ratio r in the range [initial_ratio, 1.0]:
    - initialize scenario $o(r)$
    - Oversamplers $O \leftarrow O \cup o(r)$

Two hybrid sampling Mega-Tests (2)

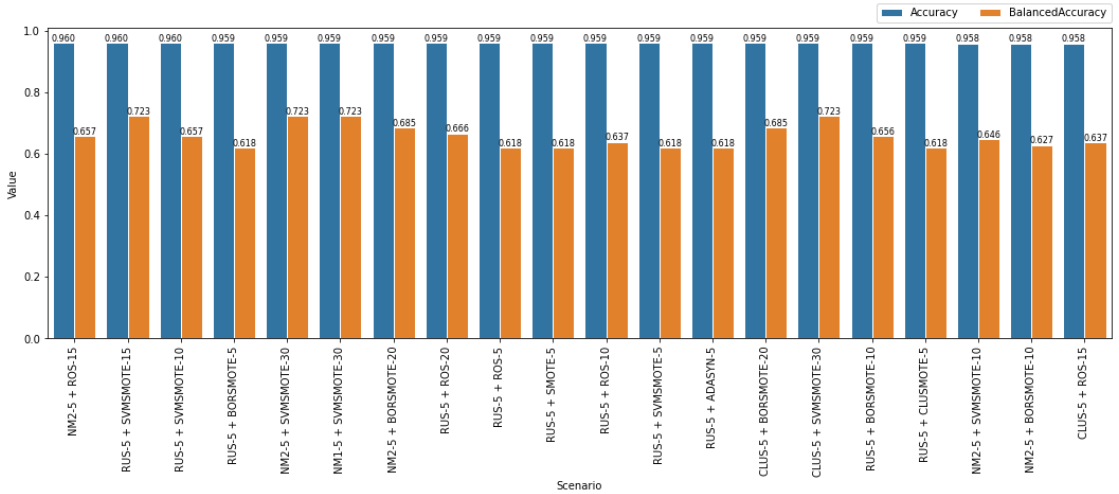
- Test 1: First undersampling, then oversampling.
- For each undersampler $u(r_1) \in U$:
  - Perform undersampling until $IR = r_1$
  - For each oversampler $o(r_2) \in O$:
    - If $(r_2 > r_1)$ Perform oversampling until $IR = r_2$
- For each classifier c:
  - Perform classification

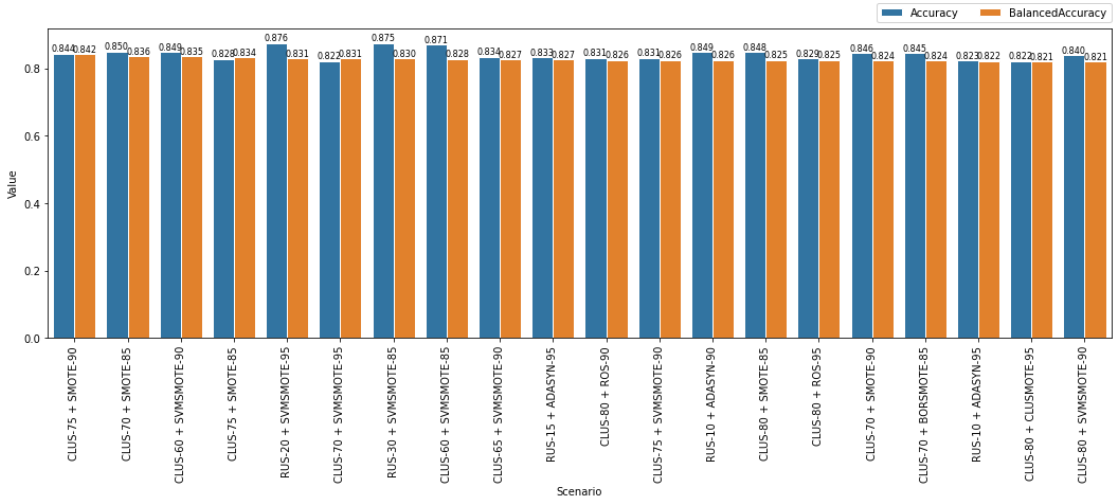
Two hybrid sampling Mega-Tests (3)

- Test 2: First oversampling, then undersampling.
- For each oversampler $o(r_1) \in O$:
  - Perform oversampling until $IR = r_1$
  - For each undersampler $u(r_2) \in U$:
    - If $(r_2 > r_1)$ Perform undersampling until $IR = r_2$
- For each classifier c:
  - Perform classification

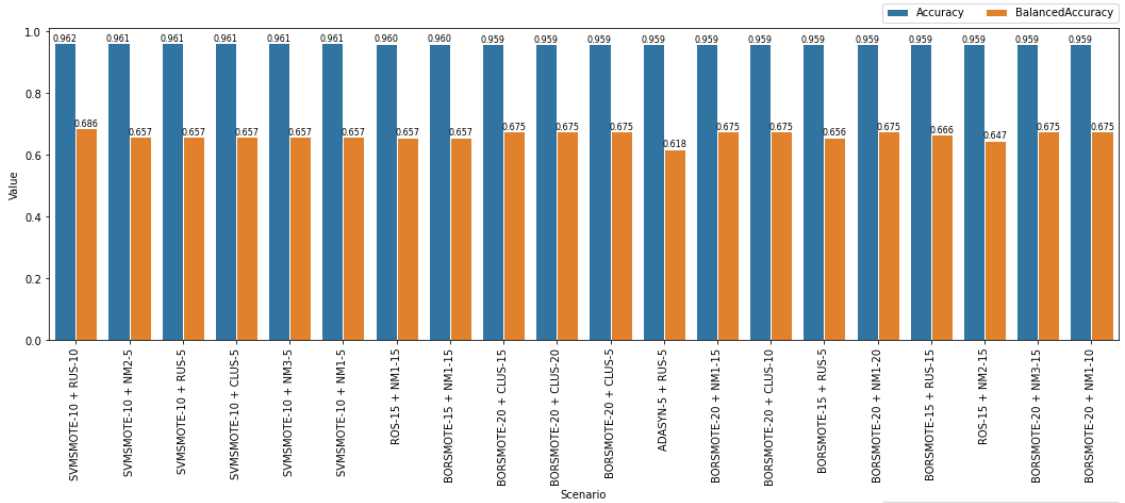
Two hybrid sampling Mega-Tests (4)

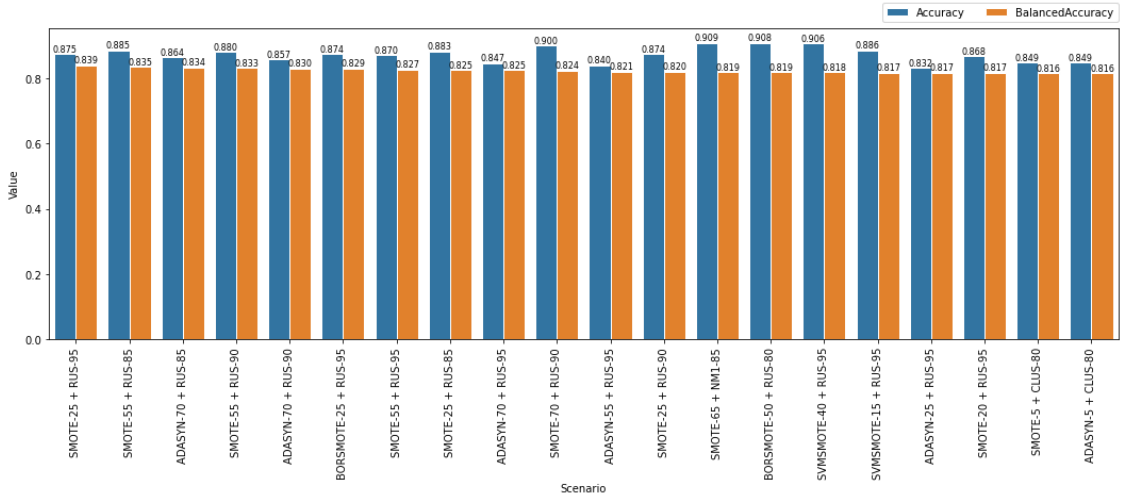
- First undersampling, then oversampling.
- Mean Accuracy values across five classifiers.





- First oversampling, then undersampling.
- Mean Accuracy values across five classifiers.





Boosting

- Boosting is a popular learning method for improving the predictive performance of a model.
- It converts multiple weak learners (e.g. decision tree stumps) into a single strong learning model (e.g. a deep decision tree).
  - AdaBoost, Gradient Boosting, Extreme Gradient Boosting.
- Initially, AdaBoost assigns equal weights to all training examples. This data is fed onto the first model which outputs its prediction for each example.
- Then, it increases the weight of the misclassified examples with a larger error. It also assigns a weight based on model performance. A model that produces good predictions will have a stringer impact on the final decision.
- In the sequel, AdaBoost passes the weighted data to the next classifier.
- These steps are repeated until the error falls under a certain threshold.

SMOTEBoost: Oversampling & Boosting

- N.V. Chawla, A. Lazarevic, L.O. Hall, K.W. Bowyer, "SMOTEBoost: Improving prediction of the minority class in Boosting", In *Proceedings of the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases*, 107–119, 2003.
- One year after the introduction of SMOTE, Chawla et al. enhanced their technique with a Boosting mechanism called SMOTEBoost.
- The method is a modification of the AdaBoost.M2 algorithm for converting weak classifiers into strong ones.

SMOTEBoost: Oversampling & Boosting

- The original AdaBoost algorithm assigns different weights to the misclassified examples compared to the correctly classified ones.
- However, all misclassified examples are assigned equal weights.
- AdaBoost still samples from a pool of data that predominantly consists of the majority class.
- There is a very strong learning bias towards the majority class cases in a skewed data set, and subsequent iterations of boosting can lead to a broader sampling from the majority class.
- AdaBoost treats both kinds of errors (FP and FN) in a similar fashion, and therefore sampling distributions in subsequent boosting iterations could have a larger composition of majority class cases.

SMOTEBoost: Oversampling & Boosting

- SMOTEBoost introduces SMOTE to each boosting round.
- In this way, each learner is fed with more samples from the minority class; **so it learns better decision regions for the minority class.**
- In addition, the introduction of SMOTE increases the diversity amongst the classifiers in the ensemble, as each iteration produces a different set of synthetic examples, and therefore different classifiers.
- In every round a weak learning algorithm is called and presented with a different distribution D_t by emphasizing particular training examples.
- The distribution is updated to give wrong classifications higher weights than correct classifications.

SMOTEBoost: Oversampling & Boosting

- Given: Set S $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ $x_i \in X$, with labels $y_i \in Y = \{1, \ldots, C\}$, where C_m, ($C_m < C$) corresponds to a minority class.
- Let $B = \{(i, y): i = 1,\ldots,m, y \neq y_i\}$
- Initialize the distribution D_1 over the examples, such that $D_1(i) = 1/m$.
- For $t = 1, 2, 3, 4, \ldots T$
  1. Modify distribution D_t by creating N synthetic examples from minority class C_m using the SMOTE algorithm
  2. Train a weak learner using distribution D_t
  3. Compute weak hypothesis h_t: $X \times Y \rightarrow [0, 1]$
  4. Compute the pseudo-loss of hypothesis h_t:

$$\varepsilon_t = \sum_{(i,y)\in B} D_t(i,y)(1 - h_t(x_i, y_i) + h_t(x_i, y))$$

  5. Set $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$ and $w_t = (1/2)\cdot(1-h_t(x_i,y)+h_t(x_i,y_i))$
  6. Update D_t : $D_{t+1}(i, y) = (D_t(i, y) / Z_t) \cdot \beta_t^{w_t}$

  where Z_t is a normalization constant chosen such that D_{t+1} is a distribution.

- Output the final hypothesis: $h_{fn} = arg \max_{y\in Y} \sum_{t=1}^{T} (log \frac{1}{\beta_t}) \cdot h_t(x, y)$

RUSBoost: Undersampling & Boosting

- C. Seiffert, T.M. Khoshgoftaar, J. Van Hulse, A. Napolitano, "RUSBoost: A hybrid approach to alleviating class imbalance", *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans,* 40 (1): 185–197, 2009.
- RUSBoost combines random undersampling with AdaBoost to mitigate class imbalance.
- Experiments have shown competitive performance compared to SMOTE and SMOTEBoost.
- It is also a simpler and faster method.

RUSBoost: Undersampling & Boosting

Algorithm RUSBoost

Given: Set S of examples $(x_1, y_1), ..., (x_m, y_m)$ with minority class $y^r \in Y$, $|Y| = 2$

Weak learner, $WeakLearn$

Number of iterations, T

Desired percentage of total instances to be represented by the minority class, N

1 Initialize $D_1(i) = \frac{1}{m}$ for all i.

2 Do for $t = 1, 2, ..., T$

a Create temporary training dataset S'_t with distribution D'_t using random undersampling

b Call $WeakLearn$, providing it with examples S'_t and their weights D'_t.

c Get back a hypothesis $h_t : X \times Y \rightarrow [0, 1]$.

d Calculate the pseudo-loss (for S and D_t):

$$\epsilon_t = \sum_{(i,y):y_i \neq y} D_t(i)(1 - h_t(x_i, y_i) + h_t(x_i, y)).$$

e Calculate the weight update parameter:

$$\alpha_t = \frac{\epsilon_t}{1 - \epsilon_t}.$$

f Update D_t:

$$D_{t+1}(i) = D_t(i)\alpha_t^{\frac{1}{2}(1+h_t(x_i,y_i)-h_t(x_i,y:y\neq y_i))}.$$

g Normalize D_{t+1}: Let $Z_t = \sum_i D_{t+1}(i)$.

$$D_{t+1}(i) = \frac{D_{t+1}(i)}{Z_t}.$$

3 Output the final hypothesis:

$$H(x) = \underset{y \in Y}{\operatorname{argmax}} \sum_{t=1}^{T} h_t(x, y) \log \frac{1}{\alpha_t}.$$

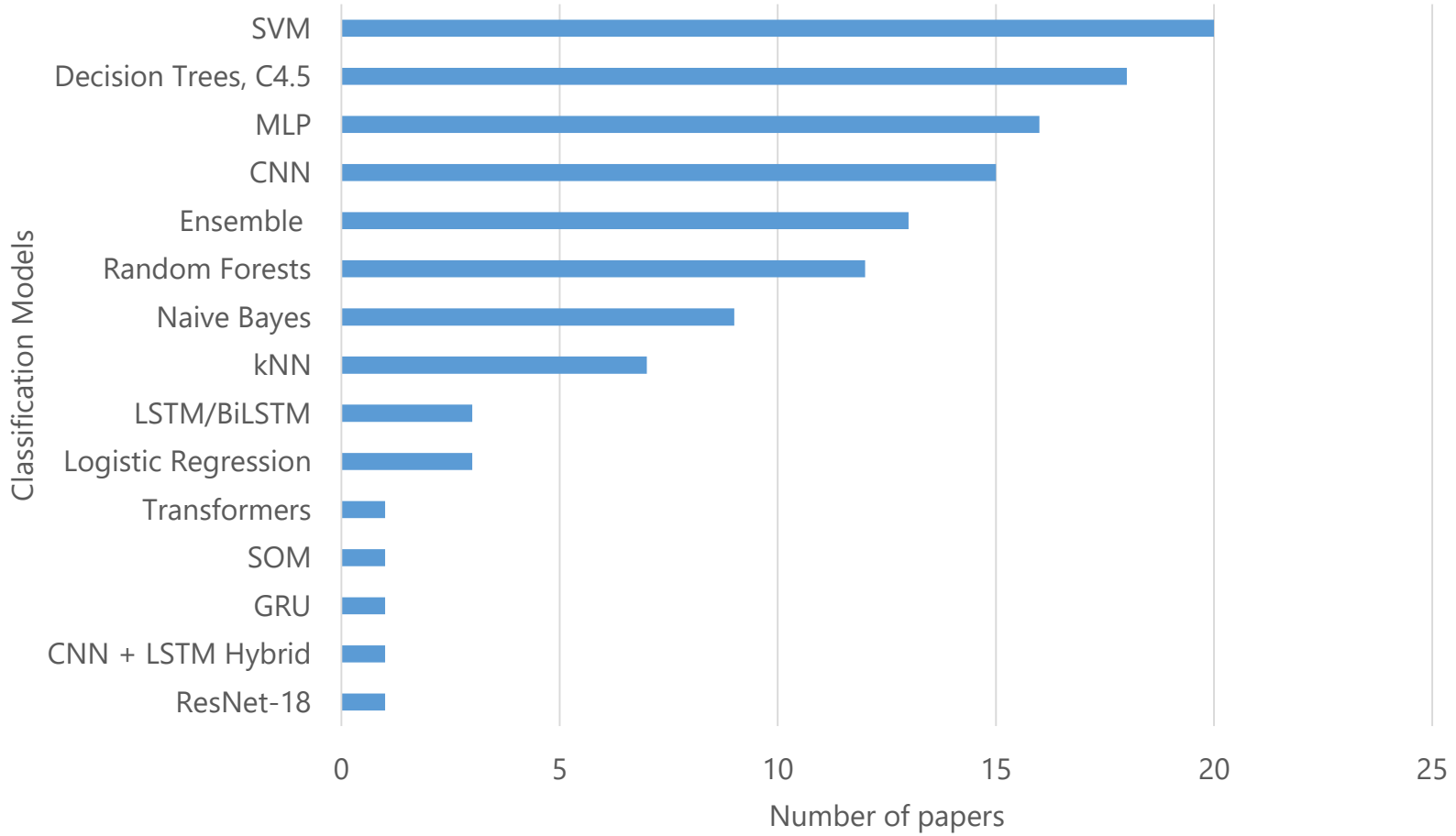
Bagging

- Bagging (Bootstrap Aggregation) is another popular ensemble learning method for improving the predictive performance of a model.
- This technique reduces the variance within a noisy dataset.
- In Bagging, a random sample of data from the training set is selected with replacement.
  - "With replacement": the sample can be selected multiple times.
- After several data samples have been selected, a group of weak models are trained independently.
- The final decision of the classifier is obtained by examining the majority of the decisions of all weak learners.

SMOTEBagging: Oversampling & Bagging

- S. Wang, X. Yao, "Diversity Analysis on Imbalanced Data Sets by using Ensemble Models", In *Proceedings of the 2009 IEEE Symposium on Computational Intelligence and Data Mining*, 324–331, 2009.
- SMOTEBagging is a combination of SMOTE and Bagging: It generates synthetic samples during subset construction.
- Then, the weak models are trained on balanced datasets, thus improving the performance of the ensemble model.

Relevant Literature: Which classifiers?



From the survey paper: L. Akritidis, P. Bozanis, "A Multi-Dimensional Survey on Learning from Imbalanced Data", Chapter in Machine Learning Paradigms - Advances in Theory and Applications of Learning from Imbalanced Data, to appear, 2023.

Conclusions (1)

- **Imbalanced data cause severe problems in classification tasks.**
- Predictors become strongly biased towards the majority class.
- They cannot effectively learn the minority classes.
- Their performance degrades rapidly.
- Very active research field.

Conclusions (2)

- **The Generative Adversarial Networks (GANs) are quickly gaining the attention of the research community in oversampling tasks.**
- Multiple such models have been presented in the literature to augment the minority classes with useful samples.
- Many more researchers are still trying to devise novel GAN-based methods for this purpose.

Conclusions (3)

- **Complex solutions do not necessarily yield better results compared to their simpler (and older) counterparts.**
- A large number of modern recent works are still employing baseline solutions, like SMOTE and its variants, or random undersampling and its variants, to restore balance in imbalanced datasets.

Conclusions (4)

- **Despite their proved effectiveness in multiple research fields, the deep learning models have not been used extensively in the problem of data imbalance yet.**
  - See the chart just 3 slides earlier.
- Isolated exceptions include the usage/modification of Convolutional Neural Nets in image classification tasks, and several deep learning models in intrusion detection systems.
- Much work is still required to effectively integrate the powerful features of deep learning to methods that confront data imbalance.

Thank you for watching

Thank you!

I would be happy to answer your questions.

e-mail: lakritidis@ihu.gr

Github repository: https://github.com/lakritidis/imbalanced