Challenges and solutions when handling imbalanced datasets

Week 7 – Advanced Topic 5

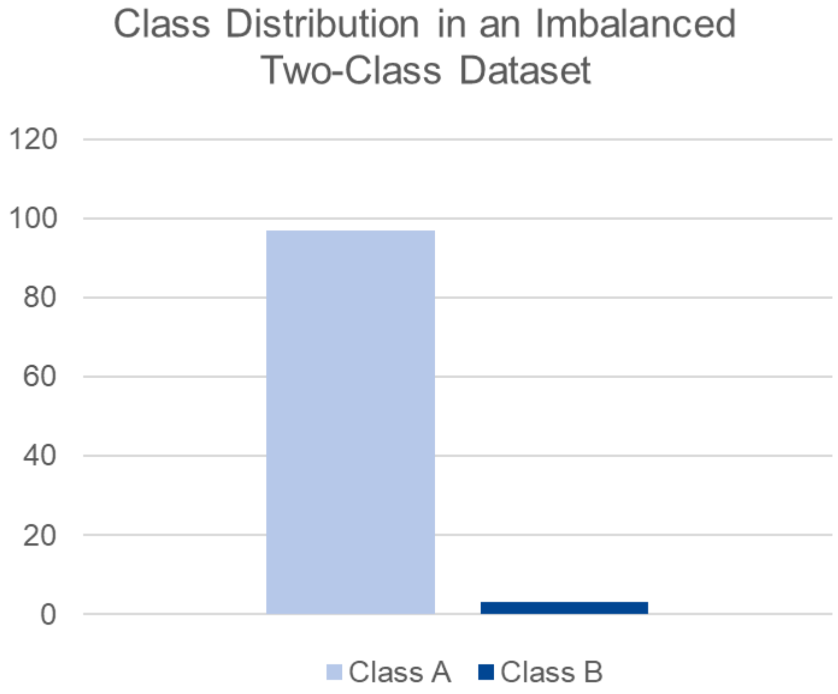
Norwegian University of Science and Technology

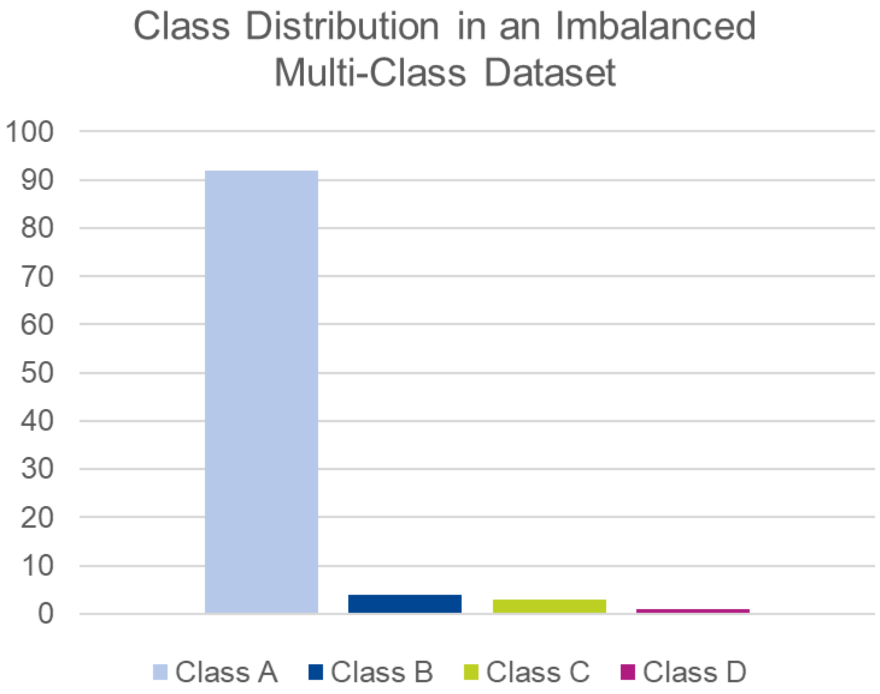
Outlines

- Introduction
- Challenges
- Solutions
- Summary

Imbalanced dataset

A dataset with unequal representation of classes.





Classification

- For **classification** models the most commonly used metric is **accuracy** (*proportion of **correctly classified** classes by the model*)
- **Accuracy works** only if the dataset **is balanced** (*a **similar number** of samples in a dataset represents **each class***)

How to detect:

- use **Cohen's kappa coefficient** (*k*) or **Matthews correlation Coefficient** (*MCC*) instead of accuracy (***or at least, use not only accuracy***)

Example:

F1	F2	F3	F4	F5	Class
2	4	2	8	5	A
1	3	1	7	2	A
5	5	1	1	7	A
3	7	1	2	6	A
4	3	2	6	3	A
5	6	3	5	3	B
3	6	1	7	4	B
1	5	1	6	2	B
1	4	3	2	6	B
6	7	2	8	4	B

Always predict class A

Predicted	Correct?
A	TRUE
A	TRUE
A	TRUE
A	TRUE
A	TRUE
A	FALSE
A	FALSE
A	FALSE
A	FALSE
A	FALSE

Number of correct classifications

$$\text{Accuracy} = \frac{5}{10} \quad \mathbf{50\%}$$

Total number of classifications

F1	F2	F3	F4	F5	Class
2	4	2	8	5	A
1	3	1	7	2	A
5	5	1	1	7	A
3	7	1	2	6	A
4	3	2	6	3	A
5	6	3	5	3	A
3	6	1	7	4	A
1	5	1	6	2	A
1	4	3	2	6	A
6	7	2	8	4	B

Always predict class A

Predicted	Correct?
A	TRUE
A	TRUE
A	TRUE
A	TRUE
A	TRUE
A	TRUE
A	TRUE
A	TRUE
A	TRUE
A	FALSE

Number of correct classifications

$$\text{Accuracy} = \frac{9}{10} = \mathbf{90\%}$$

Total number of classifications

Aleksei

Source: [2108.02497] How to avoid machine learning pitfalls: a guide for academic researchers (arxiv.org)

Imbalance in classification – we know how to detect it, but how to mitigate it? *ChatGPT on the topic*

Case: Out of **10,000 transactions**, only **200 are fraudulent (class 1)**, and **9,800 are non-fraudulent (class 0)**. The dataset is **highly imbalanced**, with 2% fraudulent and 98% non-fraudulent transactions.

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score

# X: Features, y: Target (0 for non-fraud, 1 for fraud)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Initialize Logistic Regression without class balancing
model = LogisticRegression()

# Train the model
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Evaluate the model
print(classification_report(y_test, y_pred))
```

Aleksei

	precision	recall
0	0.98	1.00
1	0.50	0.02

Precision and recall - Wikipedia

ChatGPT on Precision and Recall

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

"Of all the instances that the model predicted as positive, how many were actually positive?"

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

"Of all the actual positive instances, how many did the model correctly identify?"

Imbalance in classification – we know how to detect it, but how to mitigate it? *ChatGPT on the topic*

Aleksei

```python
# Initialize Logistic Regression with class balancing
model_balanced = LogisticRegression(class_weight='balanced')

# Train the model
model_balanced.fit(X_train, y_train)

# Predictions
y_pred_balanced = model_balanced.predict(X_test)

# Evaluate the model
print(classification_report(y_test, y_pred_balanced))
```

How the weight is used:

•The **logistic loss** function is **modified by multiplying the loss associated with each sample by its class weight**. If a class has a higher weight, the errors associated with that class will have a greater impact on the optimization process, pushing the model to reduce misclassifications for that class.

Note that these weights will be multiplied with sample_weight (passed through the fit method) if sample_weight is specified.

	precision	recall
0	0.98	1.00
1	0.50	0.02

Training **without** class balancing

	precision	recall
0	0.99	0.97
1	0.30	0.73

Training **with** class balancing

[LogisticRegression — scikit-learn 1.5.2 documentation]

Solution: balance classes

if they are imbalanced, **use robust metrics** to reveal the imbalance

Solutions

- Use of robust metrics
- Apply cost-sensitive functions
- Resampling of classes
- Algorithmic solutions

Classification – robust metrics

Aleksei

- **Cohen's Kappa (*k*)**: This metric considers the chance of agreement between the model and the data occurring by random guessing. It ranges from **-1 (complete disagreement)** to **+1 (perfect agreement),** where **0 means no agreement beyond chance**.

Cohen's kappa – Wikipedia

- **Matthews Correlation Coefficient (MCC)**: MCC is a **balanced** measure that considers **all four values from the confusion matrix (true positives, true negatives, false positives, and false negatives).** It produces a value between -1 and -1, where **1 is a perfect prediction**, **0 is a random prediction**, and **-1 indicates total disagreement**.

Phi coefficient – Wikipedia

- Or simply use **Precision and Recall** together as metrics if you prefer simpler metrics

Cost-sensitive learning

- Defining a ***cost matrix*** that specifies different penalties for each type of misclassification
- The cost matrix is designed to specify ***higher*** penalties for errors in the ***minority*** class.

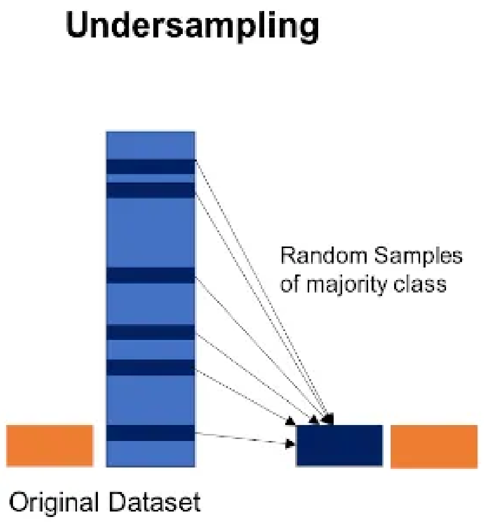
```
                   | Actual Negative | Actual Positive
Predicted Negative | C(0,0), TN      | C(0,1), FN
Predicted Positive | C(1,0), FP      | C(1,1), TP
```

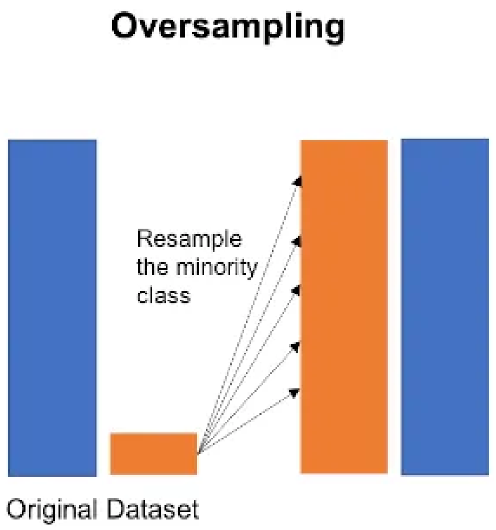
```
                   | Actual Negative | Actual Positive
Predicted Negative | 0               | 88
Predicted Positive | 5               | 0
```

	C_1	C_2	C_3
$\hat{C}_1$	0	0.5	1
$\hat{C}_2$	2	0	1
$\hat{C}_3$	3	0.5	0

Resampling

- Aims to rebalance the class distribution by
  - Undersampling: **Remove** observations from the majority class
  - Oversampling: D**uplicate** observations from the minority class

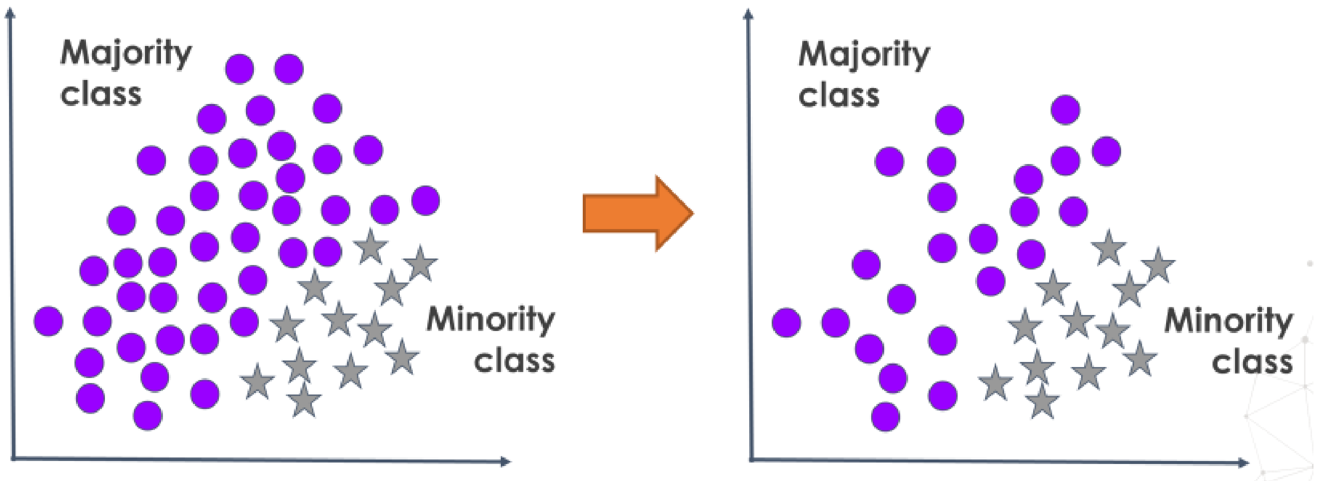




Source: https://towardsdatascience.com/how-to-deal-with-imbalanced-data-34ab7db9b100

Undersampling Techniques (1/4)

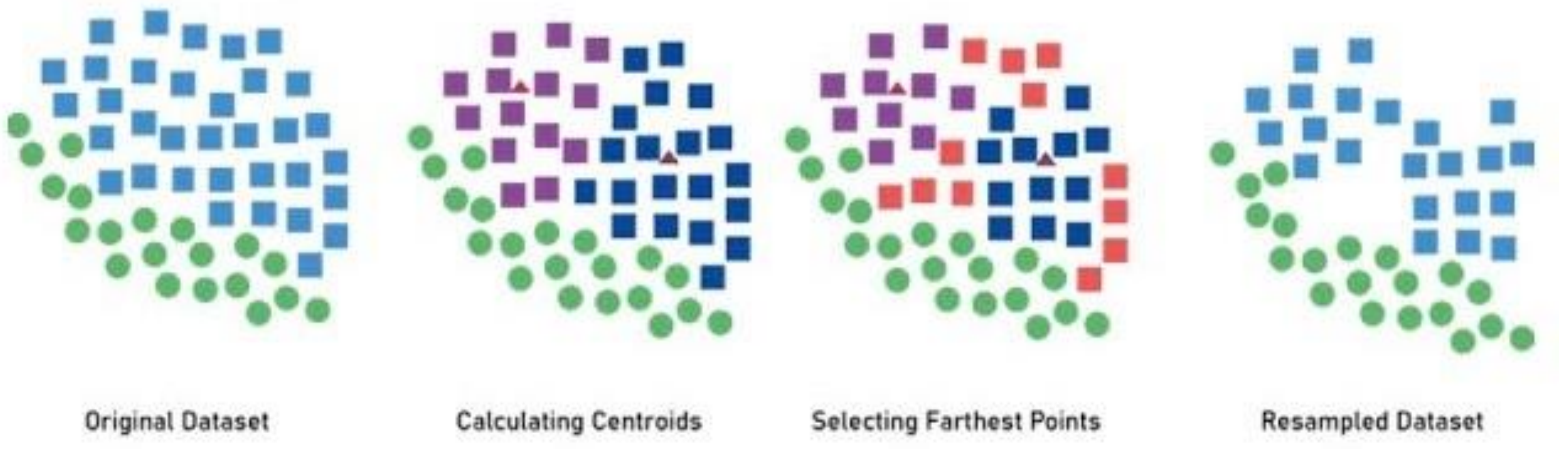
- **Random undersampling**: Randomly select and discard samples from the majority class until the desired class balance is achieved



Source: https://www.blog.trainindata.com/machine-learning-with-imbalanced-data/

Undersampling Techniques (2/4)

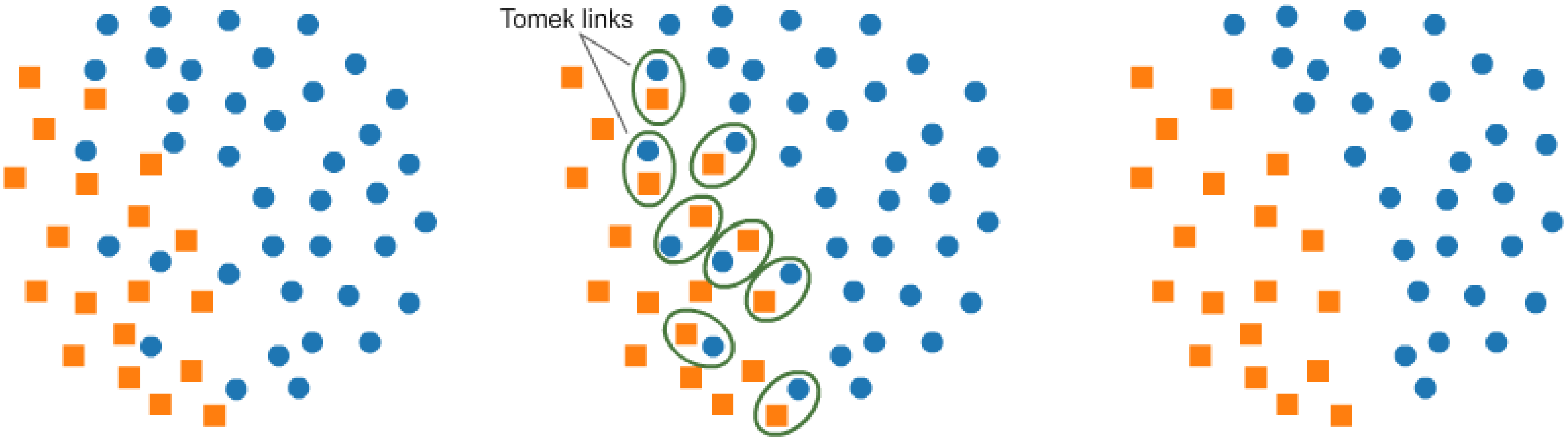
- **Cluster-based undersampling**: Uses clustering algorithms to group similar samples from the majority class and then select representative samples from each cluster



Source: https://www.youtube.com/watch?v=wnk5DJBpMb8

Undersampling Techniques (3/4)

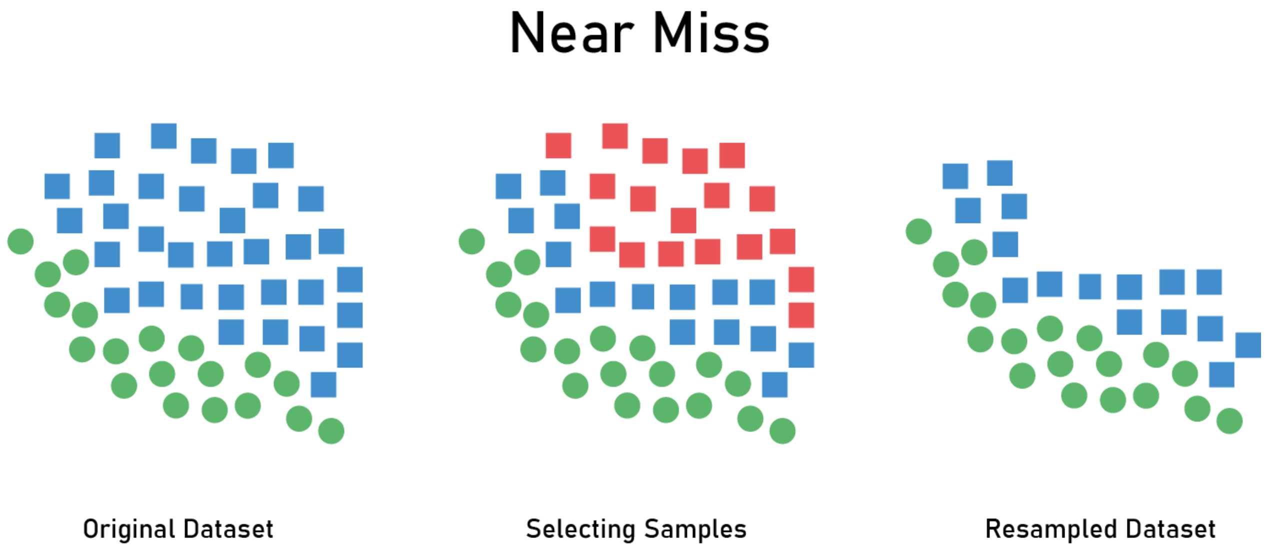
- **Tomek links undersampling**: Removes noisy and borderline samples from the majority class



Source: https://mlwhiz.com/blog/2020/01/28/imbal/

Undersampling Techniques (4/4)

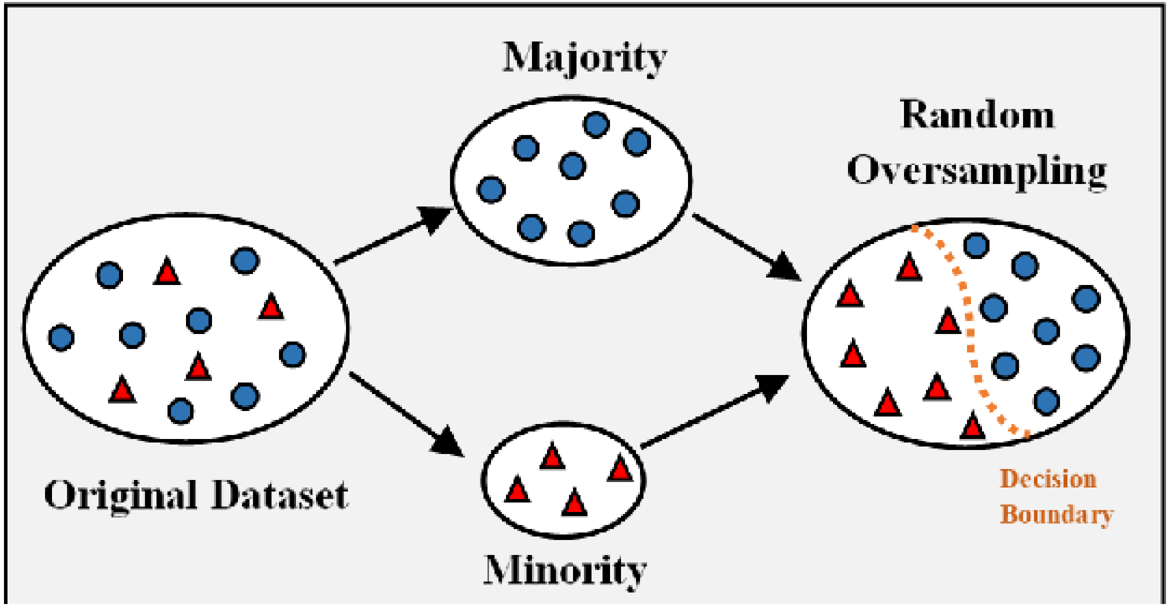
- **NearMiss undersampling**: Uses the distance between the samples to select the ones to keep or discard from the majority class



Source: https://www.comet.com/site/blog/resampling-to-properly-handle-imbalanced-datasets-in-machine-learning/

Oversampling Techniques (1/2)

- **Random:** Selects the samples randomly and generate new samples in minority class

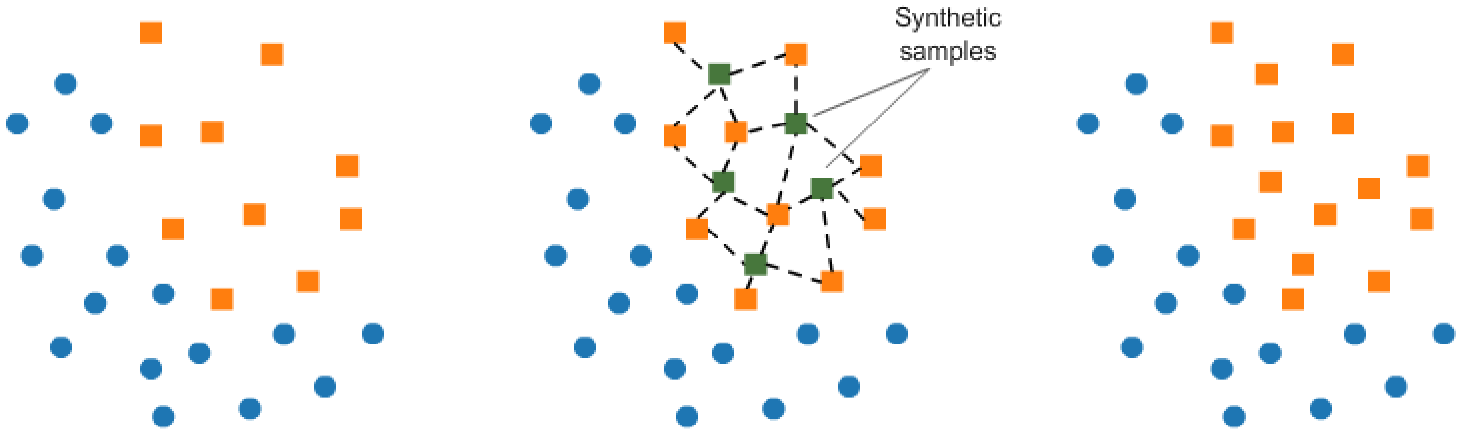


 Source: https://www.semanticscholar.org/paper/Comparison-of-Sampling-Methods-for-Imbalanced-Data-Paing-Pintavirooj/d2954ac125563670f1f630c70029aace61e8[illegible]1e

Oversampling Techniques (2/2)

- **Synthetic Minority Oversampling Technique (SMOTE)**

  Finds 'k' nearest neighbour data points of minority class and creates synthetic data points on the lines joining the primary point



Source: https://medium.com/@asheshdas.ds/oversampling-to-remove-class-imbalance-using-smote-94d5648e7d35

Drawbacks of resampling

- **Overfitting** towards minority class
- Chances of adding **noise** and **overlap**
- Model fine-tuned to balanced dataset might fail in dynamic environment with **changing data distributions**
- Sequential data, such as time series, looses **temporal** information when resampled

Solutions – Trying different algorithms

Classification Trees

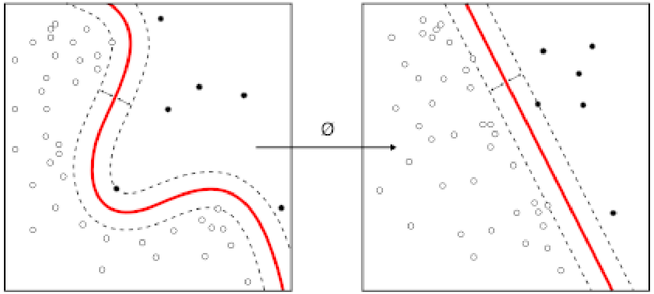
Nodes are split based on *Impurity* or *Entropy*, which automatically consider class distributions at each split.

Classification trees can *focus on the minority class* in regions of the feature space where it is dominant, even if the overall dataset is imbalanced.

Solutions – Trying different algorithms

Anomaly Detection Approach

- Treat the minority class as anomalies or outliers.
- Any observation that does not fit the majority class is considered an anomaly or outlier.
- Apply anomaly detection techniques such as **One-Class SVM**, **Isolation Forest**, or **Autoencoders**.



Anomaly Detection image from Wikipedia (CC BY-SA 4.0)

Summary

- Imbalance data can cause **overfitting** of the majority class
- Use **resampling** methods for having a balanced dataset
- Use **cost sensitive** learning \ class weighting for making the model focus on the minority class
- Handling class imbalance is best suited for task like **classification** but not for anomaly detection