Recap

01 May 2024 01:14

Imbalanced Data

output → category

fraud detection

rare disease → 1000

dumb model

No

995 No

5 Yes

99.5%

Problem → bias for majority → minority class

metrics

accuracy

Recall

Precision

Solution

995 → 50

1. Undersampling ✓✓ — Fri ← Monday
2. Oversampling ✓✓
   - Random over sampling
   - SMOTE
     - variants
- Hybrid
- Ensemble methods
- Cost sensitive learning (L)
- Class weighting
- Threshold tuning

[0.5 → 1, 0] → 0.6 → 1, 0

Thu → [Interview questions on Projects] → [Ronan] AVP - BoB

Random Oversampling

01 May 2024 20:16

SMOTE (why?)

Imbalanced data

↓

Biased for majority class

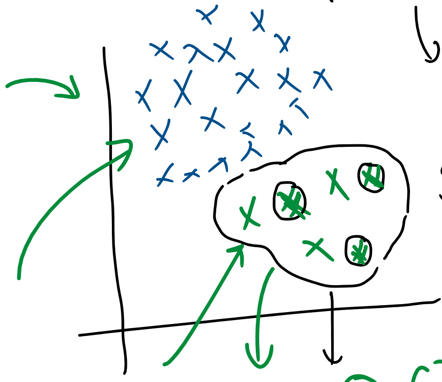
→ Random oversampling

overfitting → bad results

train test split

X train → resampling over → X_resampled → algorithm

X_test (imbalanced)



imb

balanced duplicate

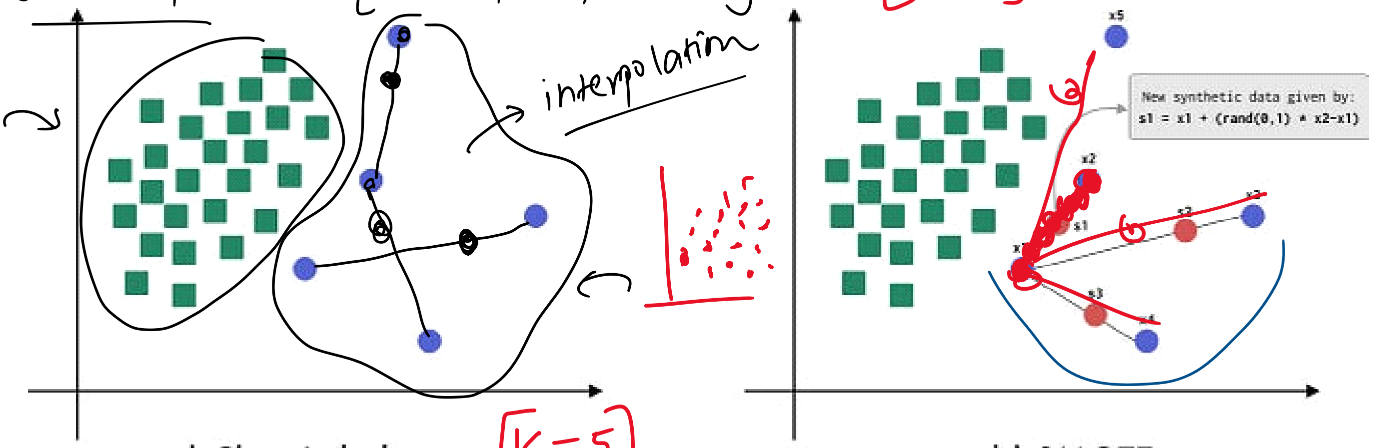
green increase

6 → 9 10

[SMOTE] Synthetic minority oversampling Technique

01 May 2024 01:14

oversampling → [minority → upsample]



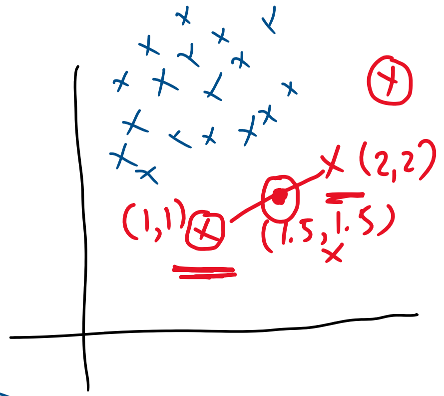
a) Class Imbalance [K=5] b) SMOTE

KNN → [K=5]

maj = min

- Train a KNN on minority class observations - find each observation's 5 closest neighbours

0-1 0.5



To create the new synthetic data:

- Select examples from the minority class at random
- Select a neighbour of each example at random (for the interpolation)
- Extract a random number between 0 and 1
- Calculate the new examples as = original sample - factor * (original sample - neighbour)
- The final dataset consists of the original dataset + the newly created examples

maj min

900 100

800

sample - 0.5 [sample - neighbor]

(1,1) - 0.5 [(1,1) - (2,2)]

(1,1) - 0.5 [(-1,-1)]

(1,1) - (-0.5,-0.5)

(1.5,1.5)

Disadvantages

variants

1. Does Not Handle Categorical Data Well

numerical

SMOTE assumes that the data is numeric and continuous. Therefore, it struggles with categorical data because it relies on linear interpolations between samples. Applying SMOTE directly to categorical features can lead to the generation of nonsensical instances, as interpolation might produce values that do not correspond to any valid category.

2. Computational Complexity

high - distances

The method is computationally expensive, especially with large datasets or datasets with a high number of dimensions (features). This is because SMOTE calculates distances between all pairs in the minority class to find the k-nearest neighbors, which can be computationally intensive.

3. Dependency on the Choice of Neighbors

[K=5] [K=3] [K=25]

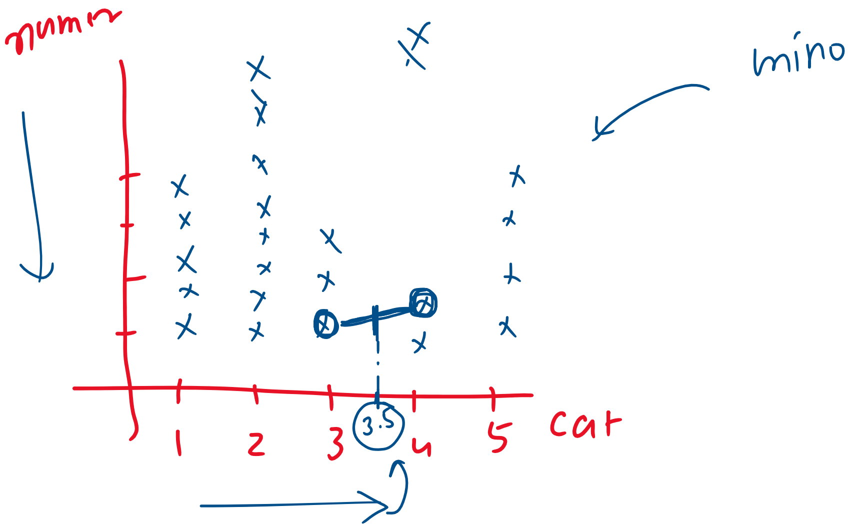
The performance of SMOTE heavily depends on the number of nearest neighbors (k-neighbors) chosen for generating the synthetic samples. Choosing too few neighbors can limit the diversity of the synthetic samples, while choosing too many might include distant, less relevant samples for interpolation, leading to less accurate synthetic data.

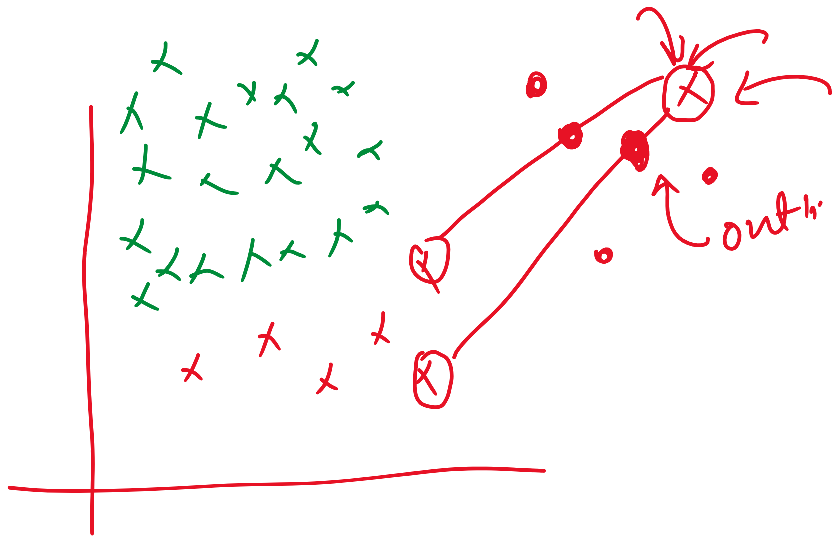
4. Sensitive to Outliers

SMOTE can be sensitive to outliers because it uses these as bases to create more samples. If outliers are present in the minority class, SMOTE will generate more outliers, which can skew the model and lead to poor generalization on unseen data.

5. Balance Achieved May Not Reflect True Nature

Balancing the classes via synthetic data does not mean that the resulting dataset reflects the true nature of the underlying problem. This artificial balance can sometimes lead to models that perform well on balanced training data but less effectively on real-world, imbalanced datasets.



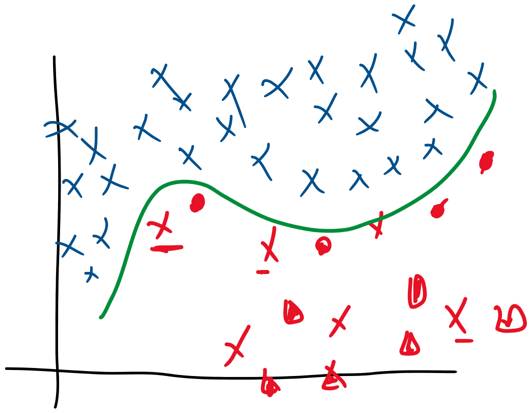


Variants

Borderline SMOTE

ADASYN

SVMSMOTE



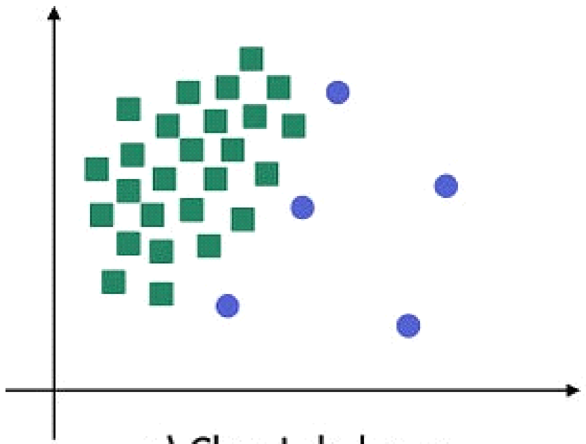
A → more pts near the

B → far from decision boundary

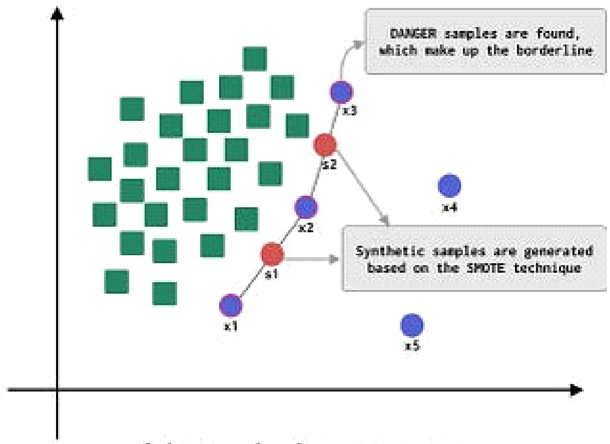
classification → complex

Borderline SMOTE

01 May 2024 08:45



a) Class Imbalance



b) Borderline-SMOTE

Core Principle - To improve the decision boundary

- Train a KNN on entire dataset
- Find the M closest neighbours to each observation from the minority group
- If most, but not all, neighbours belong to a different class, add the observation to a DANGER group

- Train another KNN only on minority group, find each DANGER group observation's closest K neighbours
- Interpolate as in SMOTE from templates in DANGER group to minority neighbours

$\boxed{K=5}$

majority / minority

minority

M neighbors

K=5

minority

normal smote

entire KNN

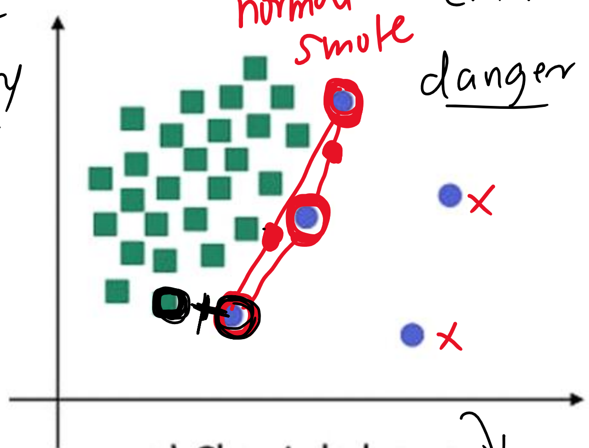
danger points

danger data points

3 type
↓
minority

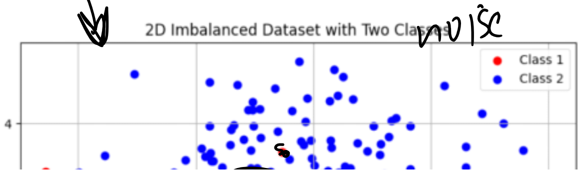
① → all the neighbors are majority wale

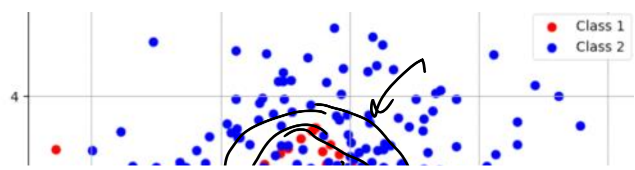
② → all neighbor minority datapoint

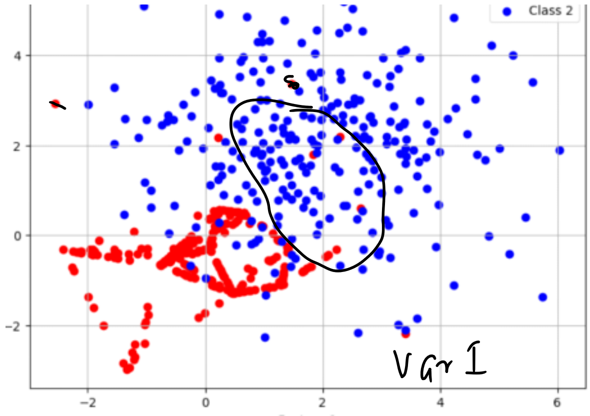


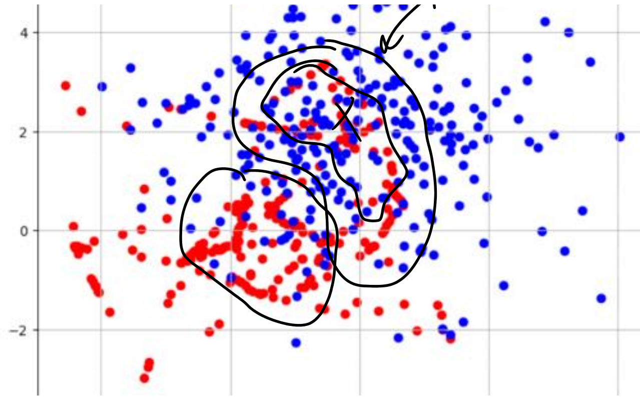
a) Class Imbalance

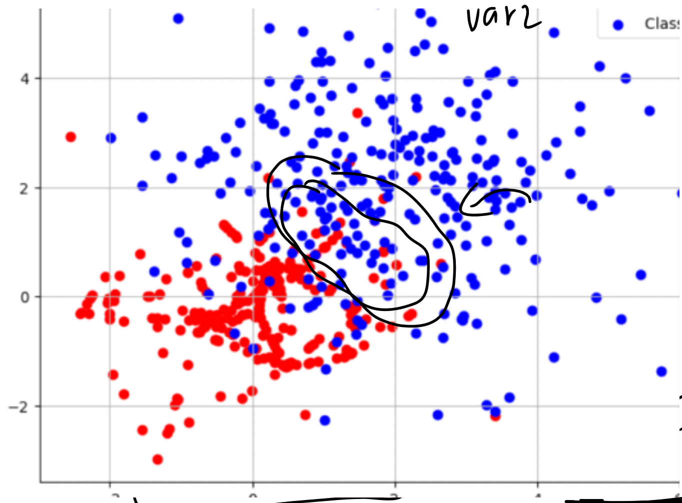
noise











factor [0 – 0.5]

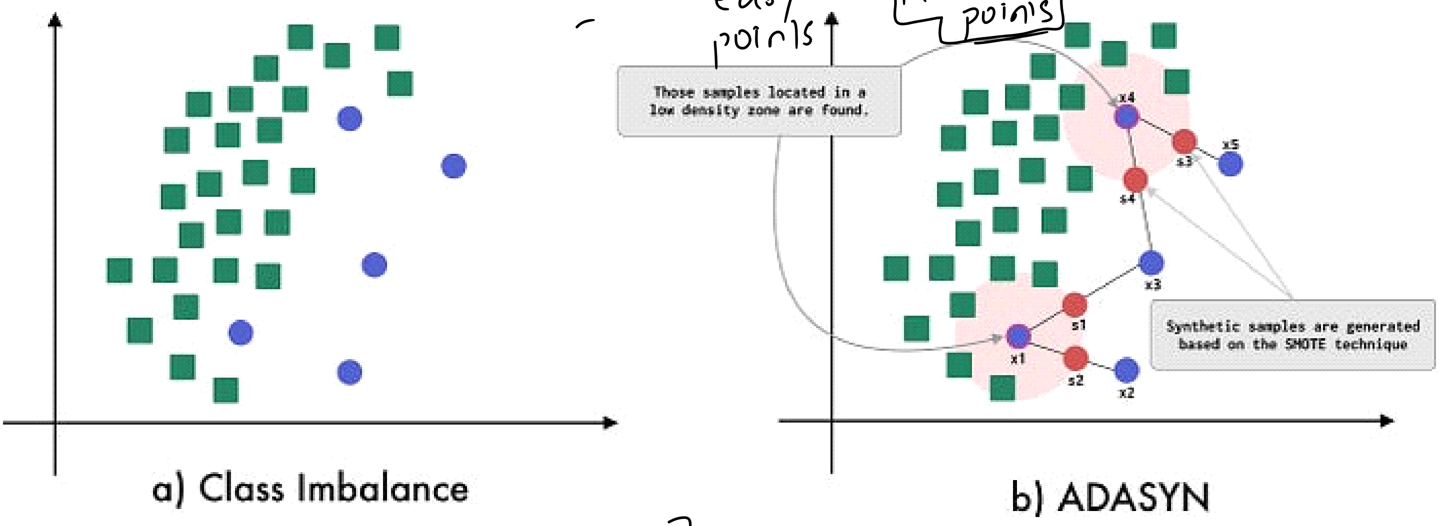
[variant 2 → unclear]

[ADASYN] Adaptive Synthetic → Classification

01 May 2024 08:59

easy points

Hard data points

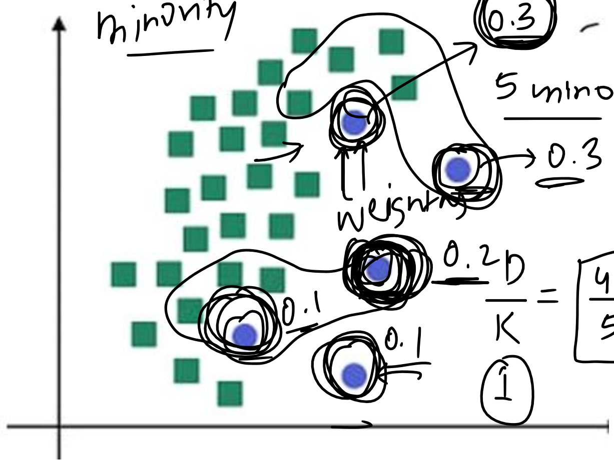


[smote] • [borderline smote]

1. Determine the balancing ratio = X(minority)/X(majority)
2. Determine the number of new examples to create: G = (Xmaj - Xmin) * factor (the factor is 1 to attain a balancing ratio of 1)
3. Train a KNN on entire dataset
4. Find the K closest neighbours to each example from the minority class
5. Determine a weighting factor for each observation of Xmin: ri = D / K, where K is the number of neighbours and D is the number of neighbours that do not belong to Xmin
6. Normalise ri: rnorm = ri / sum(r)
7. Determine how many observations should be created from each observation of Xmin: Gi = ri * G (G was determined in step 2, ri on step 6)
8. Select a neighbour of each example at random (for the interpolation, can be Xmin or Xmaj)
9. Extract a random number between 0 and 1
10. Calculate the new examples as = original sample - factor * (original sample - neighbour)

11.

minority



0.3

5 mino

0.3

weight

0.2 $\frac{D}{K} = \frac{4}{5}$

0.1

0.1

1

G = 800

factor = 1

maj 900 min 100

balancing = $\frac{100}{900} = \frac{1}{9}$

(900 − 100)

800 × factor = 1

weight = $\frac{D}{K}$

$\frac{D}{K} = \frac{3}{5}$

weight ↑ 1

0 →

- The final dataset consists of the original dataset + all the newly created examples

800 × 0.3 = 240

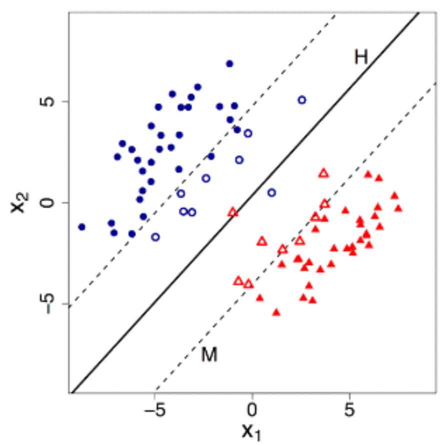
0.1 × 800 = 80

160

SVM SMOTE

01 May 2024 15:02

- Train a SVM on entire dataset
- Find the support vectors of the minority class, these will be the templates
- Train a KNN on entire dataset, find the 10 closest neighbours of the support vectors.
- Decide between inter and extrapolation: if most of the neighbours are from majority class, interpolate, otherwise, extrapolate
- Train another KNN, this time only on minority group, find the 5 closest neighbours to the support vectors
- Create the synthetic examples by inter or extrapolation between templates and their neighbours
- Note that the neighbours are not chosen at random, but instead from the closer to the furthest to create the synthetic data
- The final dataset consists of the original dataset + all the newly created examples



$$\text{new\_sample} = \text{support\_vector} + \lambda \times (\text{neighbor} - \text{support\_vector})$$

Which one to use when?

01 May 2024 15:07

1. Class Imbalance Severity:
    - If the imbalance is severe, more aggressive methods like SMOTE or its variants might be necessary. For milder imbalances, simpler techniques like random oversampling might suffice without introducing much noise.
2. Data Complexity and Overlap:
    - For datasets with complex decision boundaries or significant overlap between classes, techniques like Borderline SMOTE or ADASYN (Adaptive Synthetic Sampling) can be more effective because they focus on the regions where the classifier is most likely to make errors.
3. Model Sensitivity:
    - Consider how sensitive your model is to noise. Techniques like SMOTE might introduce synthetic outliers which can be problematic for models that are sensitive to outliers, such as k-nearest neighbors or linear models. In such cases, less aggressive techniques or a combination of oversampling with undersampling might be preferred.
4. Feature Type:
    - Check whether features are categorical or continuous. Some oversampling techniques work best with continuous features since they involve interpolation. Techniques like SMOTE and its variants typically assume continuous features. For categorical data, you might need to adapt these methods or choose techniques designed for categorical features.
5. Computational Resources:
    - Some oversampling methods are computationally intensive. If you're dealing with very large datasets or limited computational resources, you might prefer simpler or more efficient methods like random oversampling.
6. Domain-Specific Considerations:
    - Some domains might have specific requirements or constraints. For example, in medical diagnostics, false negatives might be more critical than false positives, affecting the choice of oversampling techniques.

7. Experimentation:

- Ultimately, the best way to determine the most effective oversampling technique for your dataset is through experimentation. Trying out different methods and tuning their parameters while monitoring performance on a validation set is often the most practical approach.