What is class imbalance?

If you have had the chance of working around classification problems, then is probable you have faced a problem of imbalanced classes. This occurs in datasets with a disproportionate ratio of observations. In other words, in a binary classification problem, you'd have a lot of elements of a class and very few from another. But this could also happen in a multi-classification problem when we the vast majority of the observations are clustered in one category or we have one category that's highly under-represented in comparison with the rest.

The imbalance problem is not defined formally, so there's no 'official threshold' to say we're in effect dealing with class imbalance, but a ratio of 1 to 10 is usually imbalanced enough to benefit from using balancing techniques.

Why is this a problem?

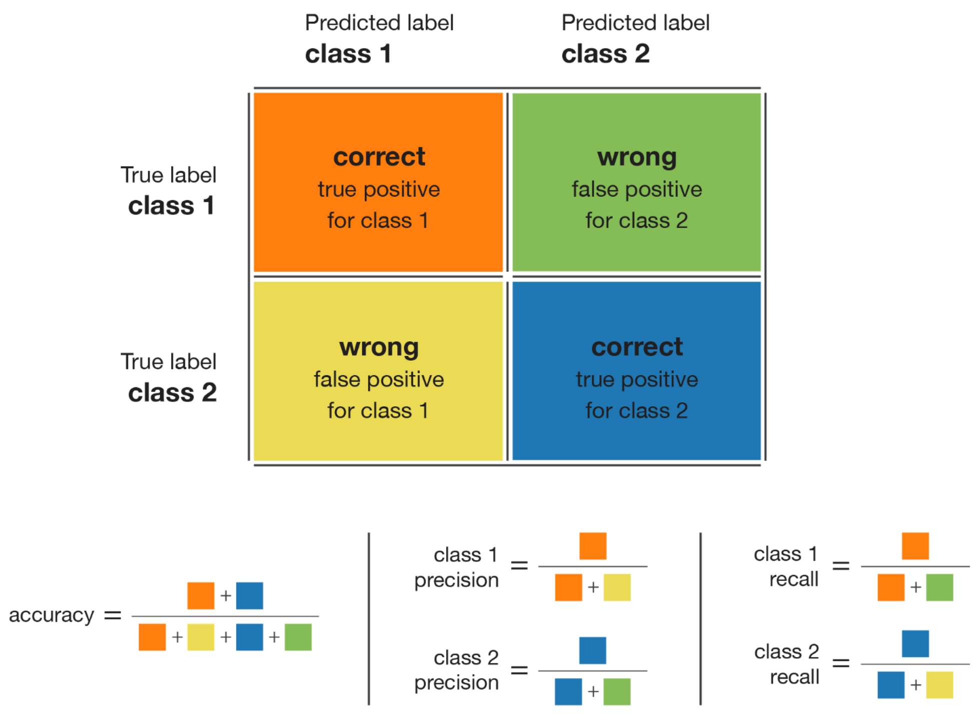
Most machine learning algorithms assume data equally distributed. So when we have a class imbalance, the machine learning classifier tends to be more biased towards the majority class, causing bad classification of the minority class.

The Accuracy Paradox

The Accuracy Paradox refers to the utility of using the Accuracy out of our Confusion Matrix as a metric for predictive modelling when classifying imbalanced classes in predictive analysis.

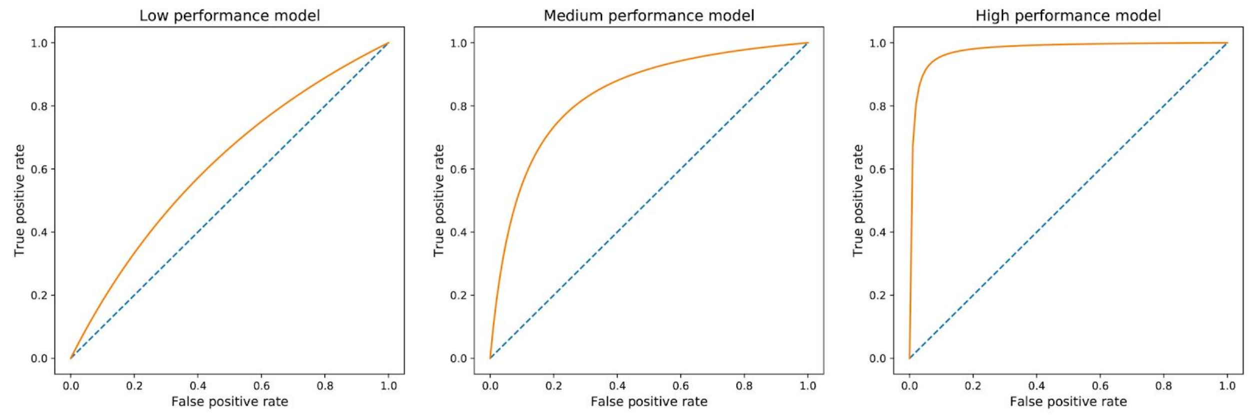
Let's take the following example to illustrate this better: suppose you're working on a binary classification problem and you model scores 90% accuracy at your first try. That sounds great, but when you dive a little deeper you discover that 90% of the data belongs to one class. This means you're predicting well only the majority class. For this reason, Precision and Recall are better measures in cases like this.

Let's remember a little about all these metrics and where they come from:



https://cdn-images-1.medium.com/max/1600/1*Yslau43QN1pEU4jkGiq-pw.png

Another interesting metric to use in this cases is the ROC curve:

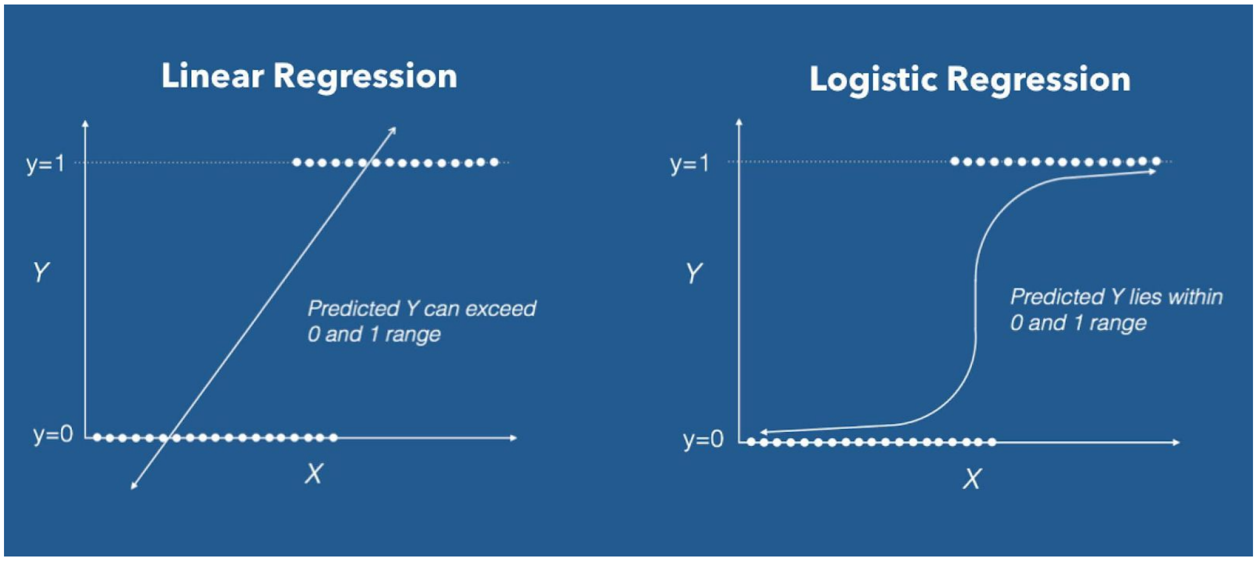


https://cdn-images-1.medium.com/max/2400/1*thHBCWlaKWIkouryKBh6Wg.jpeg

This metric shows the relationship between the Recall and the True Negative Rate or Specificity. The ideal point here would be when we have a Recall of 1 and a Specificity of also 1, because in that case, we would be predicting all the True values as True, and all the False values as False, but in practice that's usually hard and we'll try to play with the threshold of our classification algorithm to maximize the area under the ROC. Also in a multi-classification problem, we'll have as many curves as classes we may have.

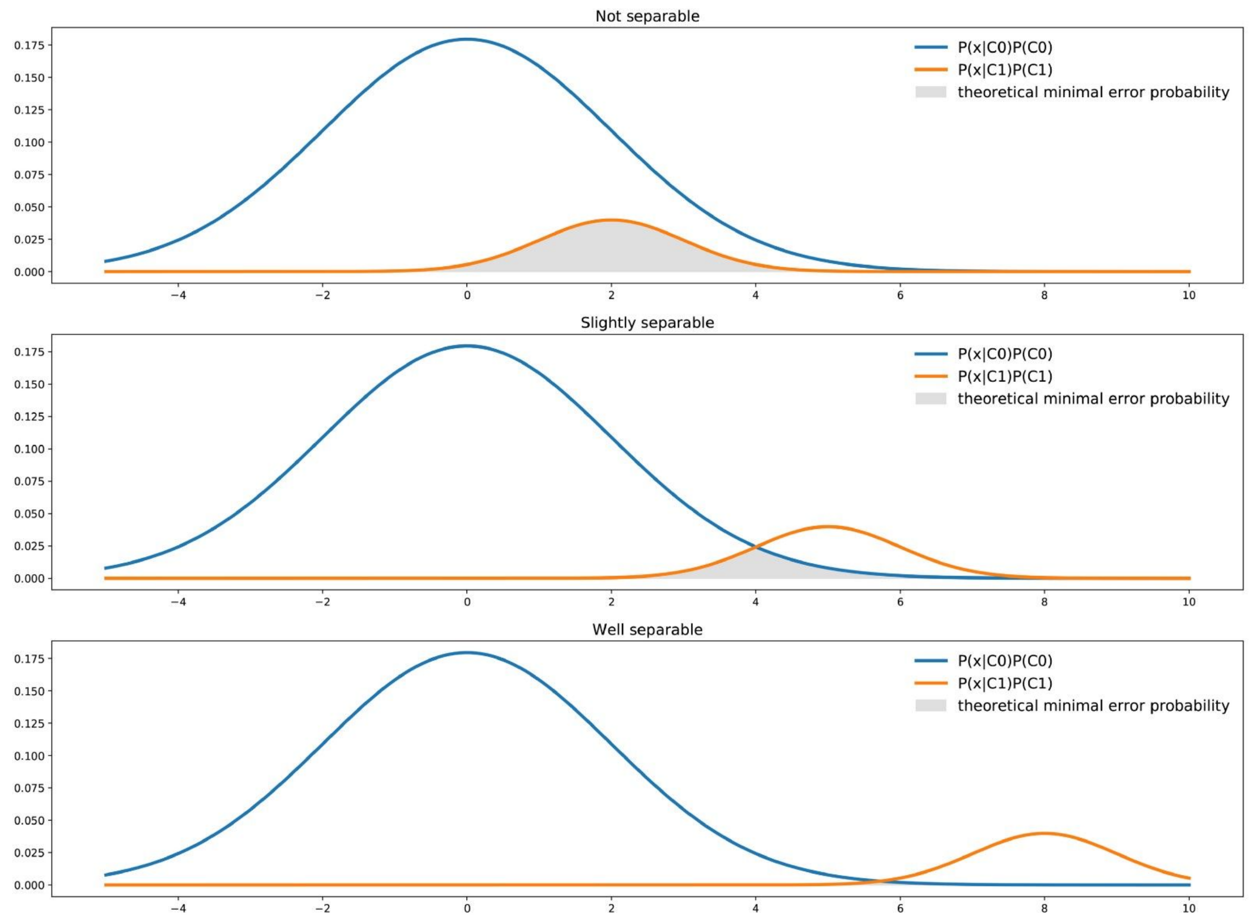
About the separability

Let's remember first that while in Linear Regression we're trying to fit a line using least squares, in a classification algorithm such a Logistic Regression, we don't have the same concept of a 'residual', so it can't use leas squares and it can't calculate R2. Instead, Logistic Regression uses something called 'maximud likelihood':


Linear Regression
y=1
Y
Predicted Y can exceed 0 and 1 range
y=0
X
Logistic Regression
y=1
Y
Predicted Y lies within 0 and 1 range
y=0
X

https://www.machinelearningplus.com/wp-content/uploads/2017/09/linear_vs_logistic_regression.jpg

In the previous, we can notice that the predictor variable X has some overlap between y=0 and y=1. This is a simple example of two classes that are slightly separable. We'll have cases where classes will overlap more and others where instead it will not overlap. For example:



https://cdn-images-1.medium.com/max/1600/1*4YVk2qmNam23rcX1xuyx7Q.jpeg

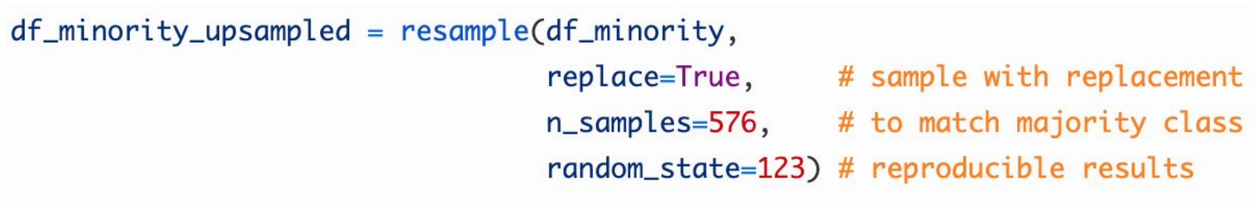
Linking this with the Theoretical Minimal Error Probability, the best possible classifier will choose for each point x the most likely of the two classes and for a given point x, the best theoretical error probability is given by the less likely of these two classes. As in the example above, for a classifier with one feature and two classes, the theoretical minimal error probability is given by the area under the minimum of the two curves. In cases where our predictor variable is well separable and we don't have any overlap in between classes, the two classes are separated enough to compensate the imbalance. But can we do when we don't have a good separability in between classes and we do have imbalanced classes?

Techniques to fight imbalanced data

If we cannot collect more data or our classes are naturally imbalanced, here are a few techniques we can use to improve our classification performance.

1. Up-sample minority class

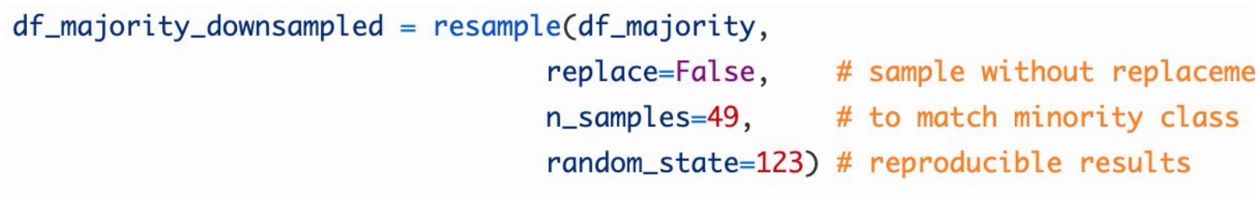
Up-sampling is a simple process about randomly duplicating observations from a minority class. You can import the resample module from sklearn.utils, separate observatons from the minority class into a new DataFrame and run the following code:



```
df_minority_upsampled = resample(df_minority,
                                 replace=True,     # sample with replacement
                                 n_samples=576,    # to match majority class
                                 random_state=123) # reproducible results
```

2. Down-sample majority class

Similar to the previous technique, but in this case removing randomly observations.

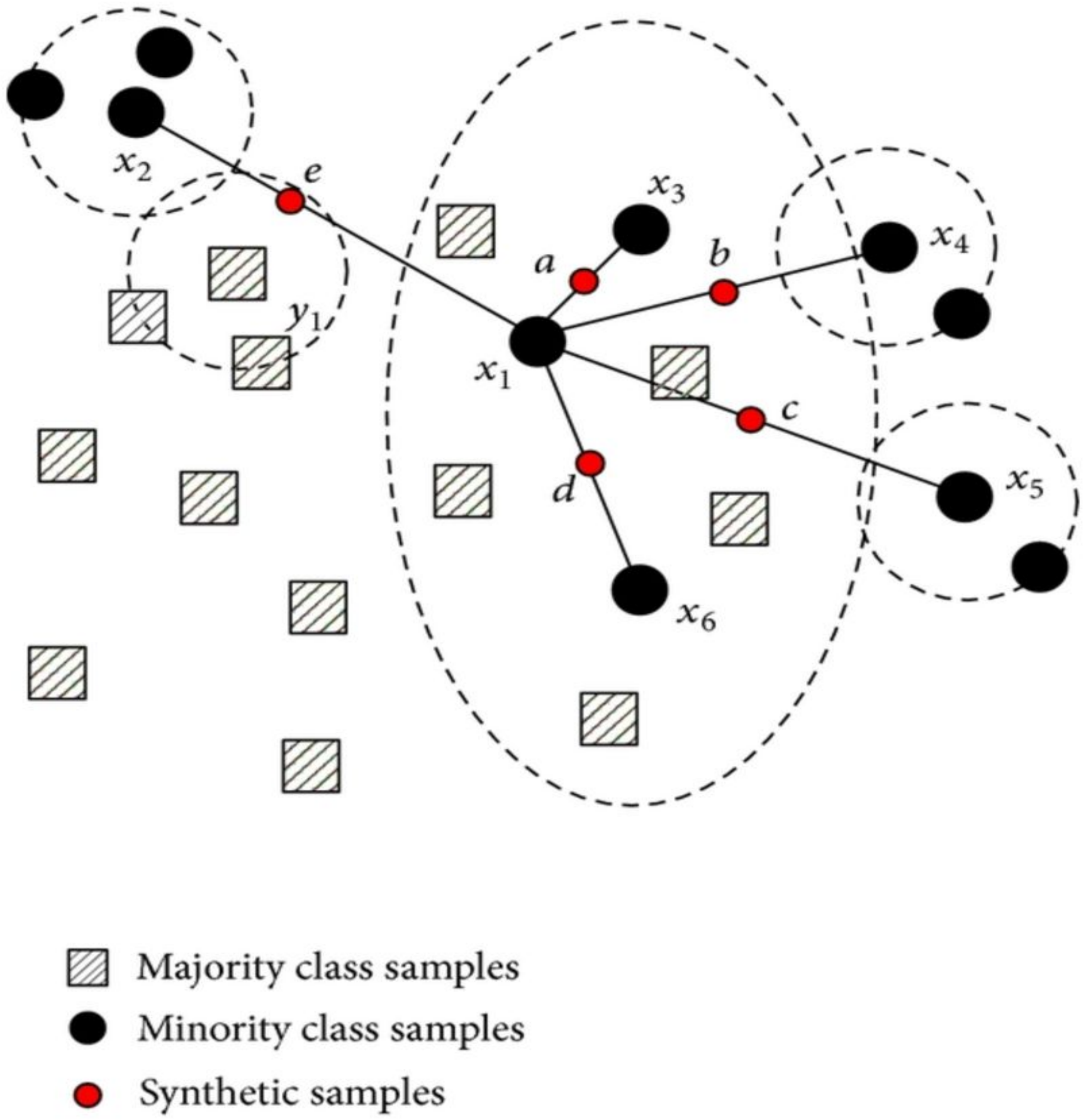


```
df_majority_downsampled = resample(df_majority,
                                 replace=False,    # sample without replaceme
                                 n_samples=49,     # to match minority class
                                 random_state=123) # reproducible results
```

The negative side of this is, as in the example, if we have few observations since we would be reducing our dataset and probably affecting our predicting power.

3. Generate Synthetic Samples

Synthetic samples are artificially generated from the original data sample. The most commonly used algorithms for generating synthetic data are SMOTE and ADASYN. The first one creates new samples based on the distances between the point and its nearest neighbours. SMOTE calculates the distances for the minority samples near the decision boundary and generates the new samples. Let’s look an example about how SMOTE works:



The key difference between ADASYN and SMOTE is that the former uses a density distribution, as a criterion to automatically decide the number of synthetic samples that must be generated for each minority sample by adaptively changing the weights of the different minority samples to compensate for the skewed distributions. The latter generates the same number of synthetic samples for each original minority sample.

4. Change the performance metric

As we talk earlier, Accuracy is not the right metric to use when we're working with imbalanced data. Instead, we could use for example Recall, Precision or ROC curves.

5. Try different algorithms

Some algorithms as Support Vector Machines and Tree-Based algorithms are better to work with imbalanced classes. The former allows us to use the argument class_weight='balanced' to penalize mistakes on the minority class by an amount proportional to how under-represented it is. Meanwhile, Decision Trees often perform well on imbalanced datasets because their hierarchical structure allows them to learn signals from both classes.

References:

https://en.wikipedia.org/wiki/Accuracy_paradox
https://towardsdatascience.com/handling-imbalanced-datasets-in-machine-learning-7a0e84220f28
https://www.machinelearningplus.com/wp-content/uploads/2017/09/linear_vs_logistic_regression.jpg
https://www.datascience.com/blog/imbalanced-data
https://elitedatascience.com/imbalanced-classes
https://www.datasciencecentral.com/profiles/blogs/handling-imbalanced-data-sets-in-supervised-learning-using-family
https://stackoverflow.com