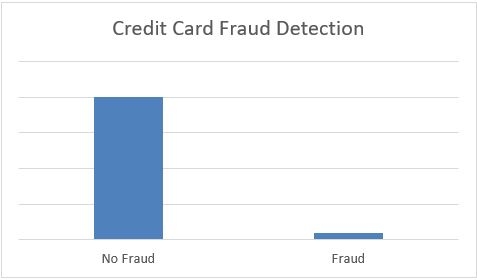
Imbalance data refers to the types of datasets where the target class is imbalanced or has an uneven distribution of observations.

For example, one class label has a very high number of observations and the other has a very low number of observations. We can better understand it with an example.

**Example:** Let’s assume that XYZ is a bank that issues a credit card to its customers. Now the bank is concerned that some fraudulent transactions are going on and when the bank checks their data they found that for each 2000 transaction there are only 30 Nos of fraud recorded. So, the number of fraud per 100 transactions is less than 2%, or we can say more than 98% transaction is “No Fraud” in nature. Here, the class “No Fraud” is called the **majority class,** and the much smaller in size “Fraud” class is called the **minority class**.



More such example of imbalanced data is –

* · Disease diagnosis
* · Customer churn prediction
* · Fraud detection
* · Natural disaster

Class imbalanced is generally normal in classification problems. But, in some cases, this imbalance is quite acute where the majority class’s presence is much higher than the minority class.

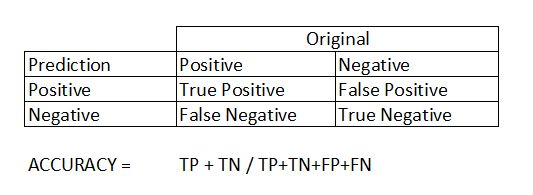
For example, For the same dataset of credit card transactions, there could be 99.9% of legitimate transactions and only 0.1% of fraud. This is a highly imbalanced dataset.

## 

## **Problems with imbalanced data classification**

If we explain it in a very simple manner, the main problem with imbalanced dataset prediction is how accurately the majority and minority classes are being predicted.

**Example:** Let’s assume we are going to predict disease from an existing dataset where for every 100 records only 5 patients are diagnosed with the disease. So, the majority class is 95% with no disease and the minority class is only 5% with the disease.   
Now, assume our model predicts that all 100 out of 100 patients have no disease. This is due to our classifier becoming biased towards the prediction.  
  
Sometimes when the records of a certain class are much more than the other class, our classifier may get biased towards the prediction.   
In this case, the confusion matrix for the classification problem shows how well our model classifies the target classes and we arrive at the accuracy of the model from the confusion matrix.



It is calculated based on the equation given below.

Accuracy =

In this case, Accuracy = = 0.95 = 95%

It means that the model fails to identify the minority class yet the accuracy score of the model will be 95%.

Thus our traditional approach of classification and model accuracy calculation is not useful in the case of the imbalanced dataset.

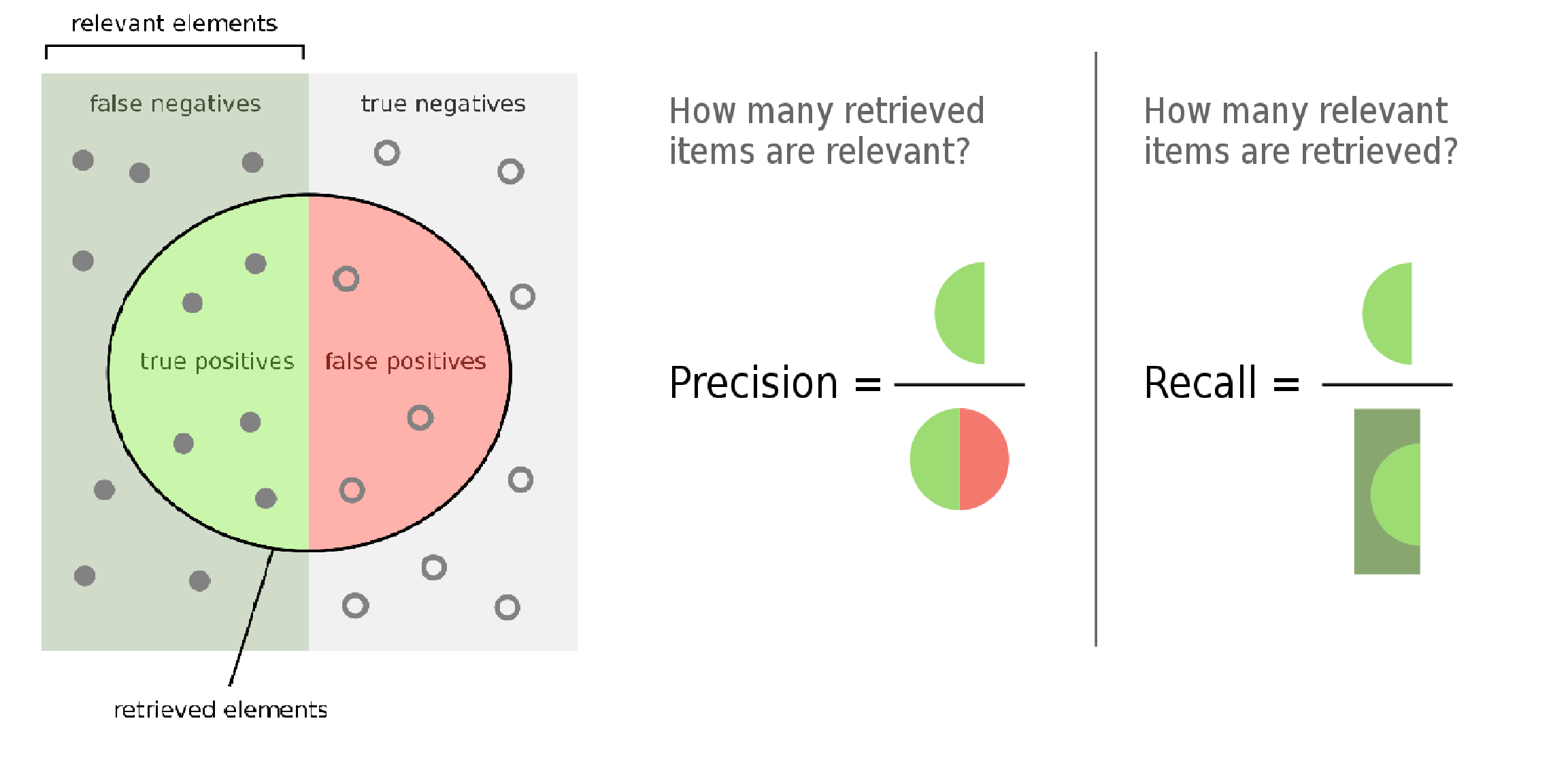
We can balance the imbalanced dataset in multiple ways, Those are,

### **1. Use the right evaluation metrics**

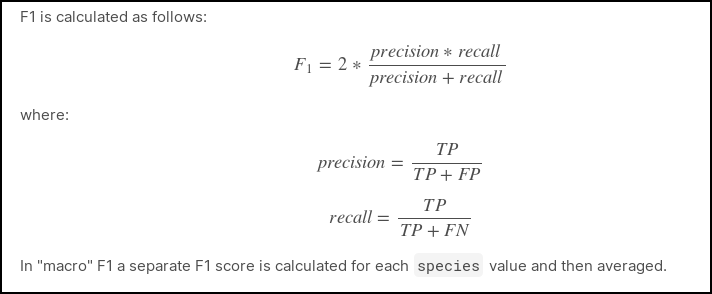
Applying inappropriate evaluation metrics for models generated using imbalanced data can be dangerous. Imagine our training data is the one illustrated in the graph above. If accuracy is used to measure the goodness of a model, a model which classifies all testing samples into “0” will have an excellent accuracy (99.8%), but obviously, this model won’t provide any valuable information for us.

In this case, other alternative evaluation metrics can be applied such as:

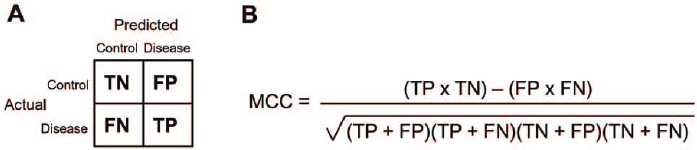
* **Precision/Specificity:** The number of relevant documents retrieved by a search divided by the total number of documents retrieved by that search.
* **Recall/Sensitivity:** The number of relevant documents retrieved by a search divided by the total number of existing relevant documents



* **F1 score:** This is a measure of a model's accuracy on a dataset. The F-score is a way of combining the precision and recall of the model, and it is defined as the harmonic mean of the model's precision and recall.

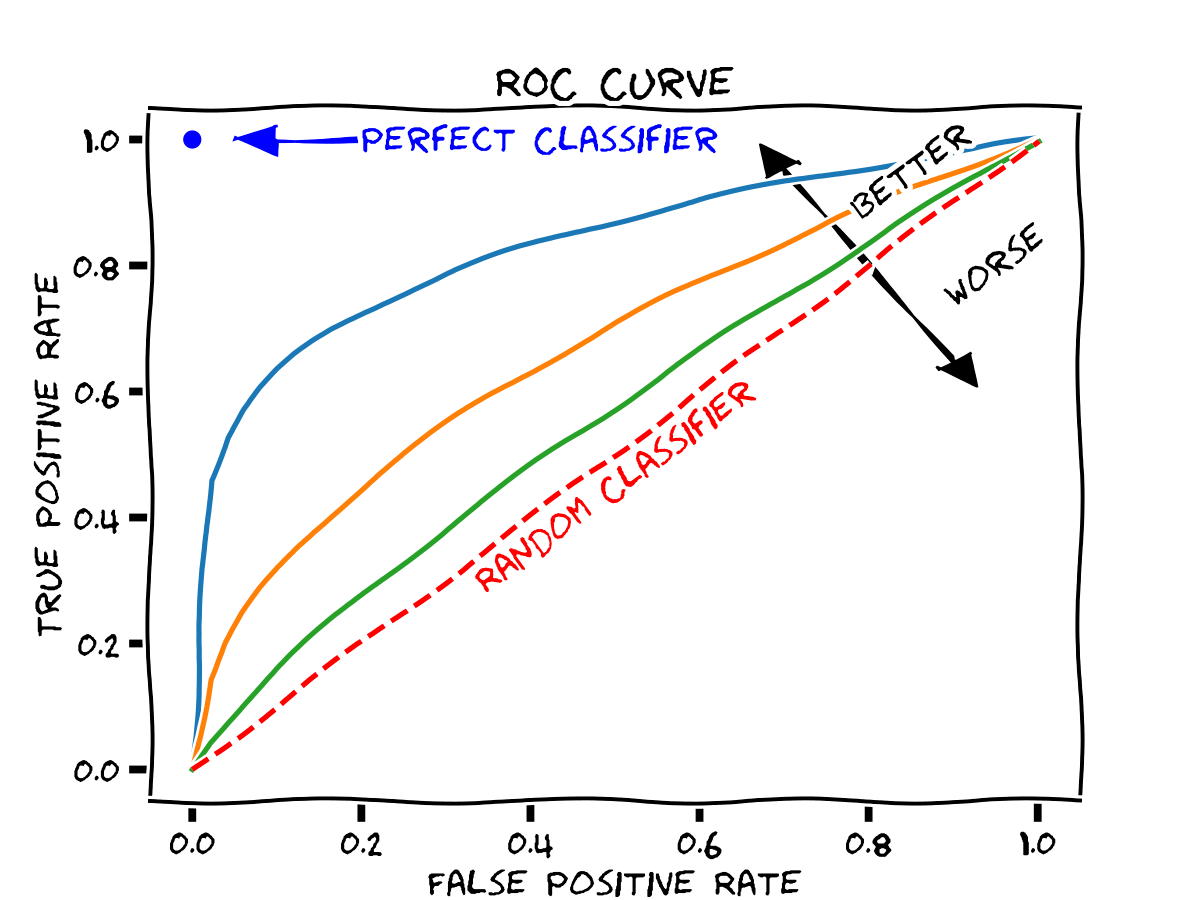


* **Matthews correlation coefficient (MCC):** It is the measure of the quality of binary classifications. It is a statistical measure used to evaluate the strength of association between two dichotomous variables.

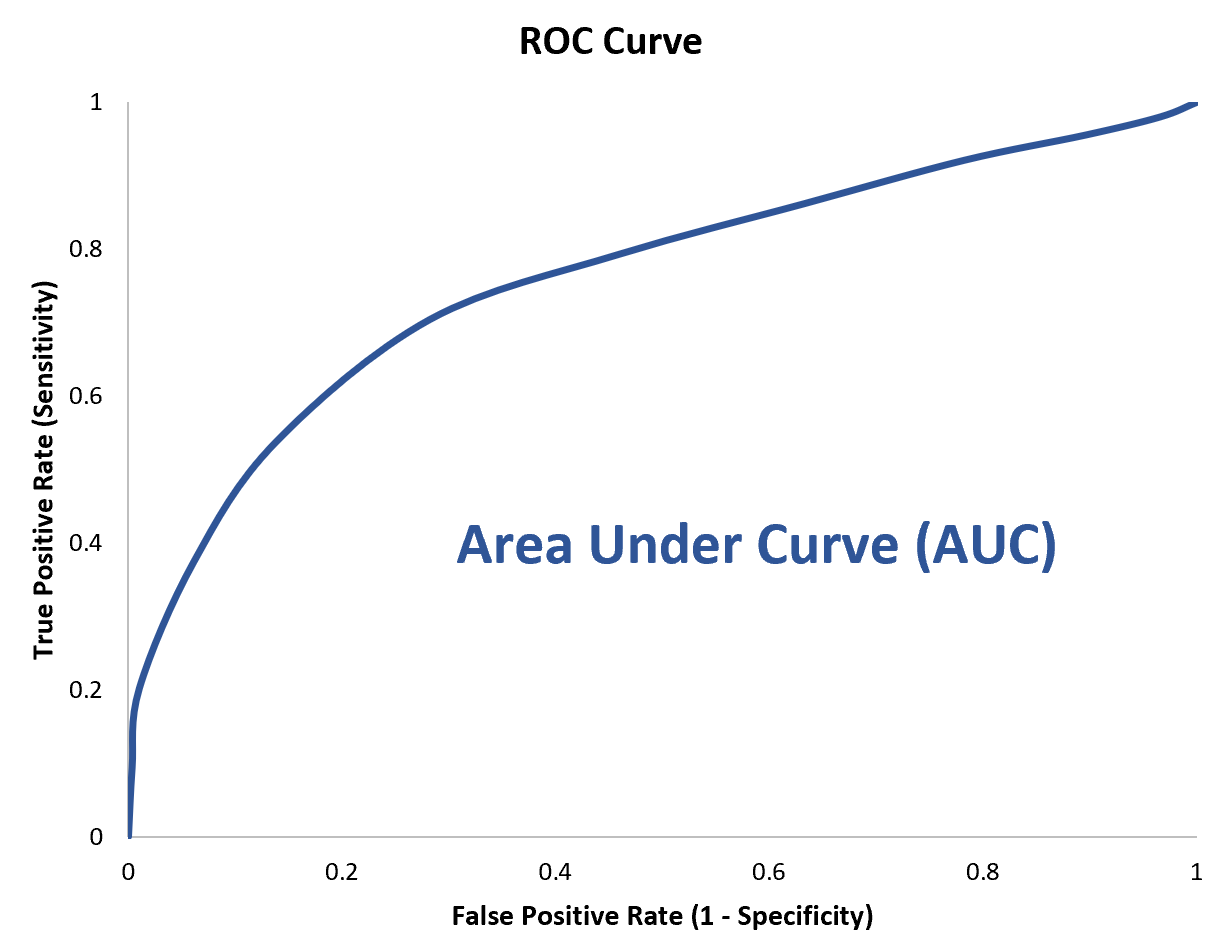
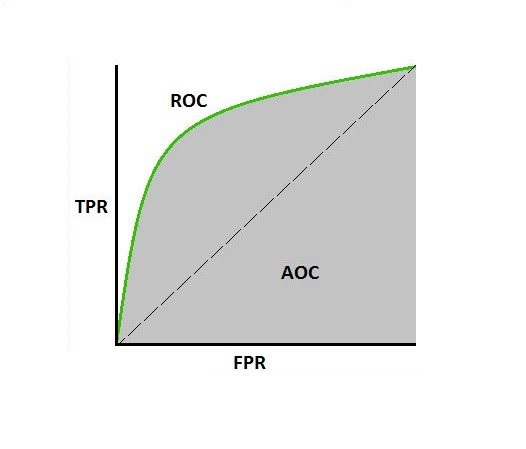


* **Receiver operating characteristic (ROC):** A Receiver Operator Characteristic (ROC) curve is a graphical plot used to show the diagnostic ability of binary classifiers.

The ROC curve shows the trade-off between sensitivity (or TPR) and specificity (1 – FPR). Classifiers that give curves closer to the top-left corner indicate a better performance. As a baseline, a random classifier is expected to give points lying along the diagonal (FPR = TPR). The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test.

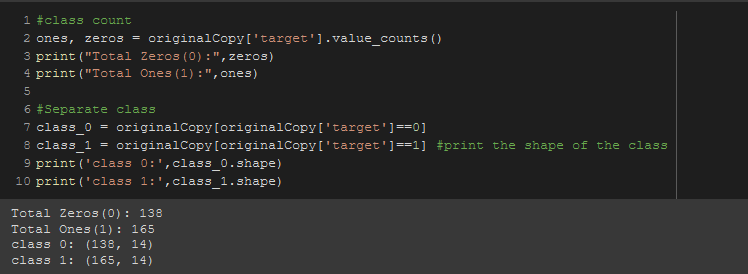


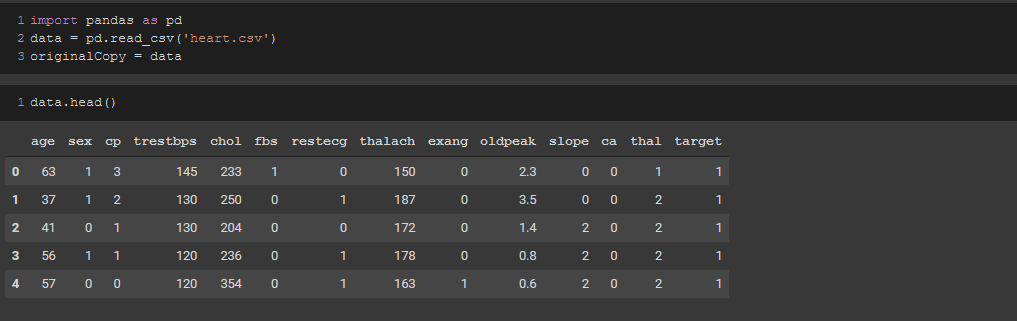
* **Area Under the Curve (AUC):** This is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve. This upholds the relation between true-positive rate and false-positive rate.   
  The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.



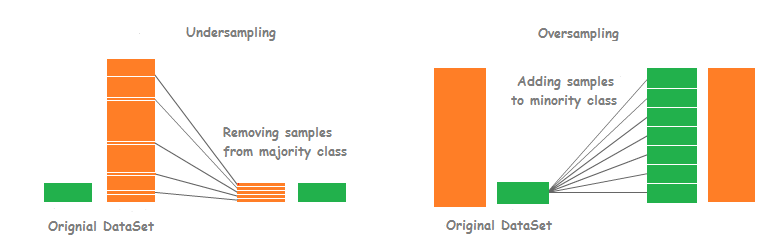
**Example**

Let the class count of a target variable of a dataset be,



**Dataset:  
**

### **2. Resample the training set**

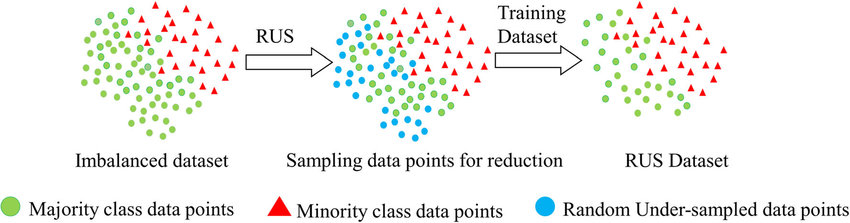


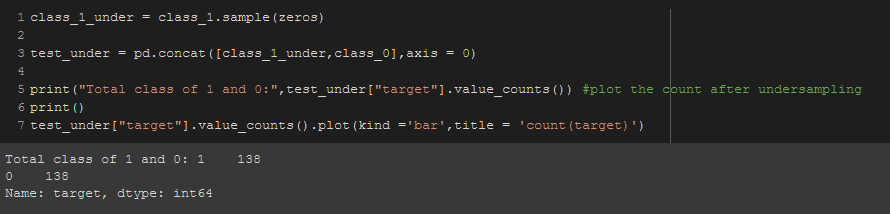
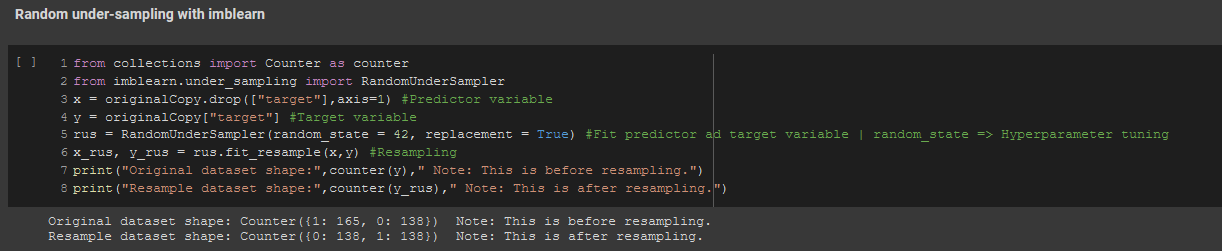
**Fig:** Undersampling & Oversampling

Apart from using different evaluation criteria, one can also work on getting different datasets. Two approaches to making a balanced dataset out of an imbalanced one are under-sampling and over-sampling.

* **Under-sampling:** Undersampling resamples the majority class points in the data to make them equal to the minority class points.   
  The major disadvantage of undersampling is that we do not use a significant chunk of the data, which contains some information.  
  We can avoid that by using a technique called oversampling instead of undersampling.

**Random Under sampling:** Random undersampling involves randomly selecting examples from the majority class and deleting them from the training dataset. In the random under-sampling, the majority class instances are discarded at random until a more balanced distribution is reached.

****

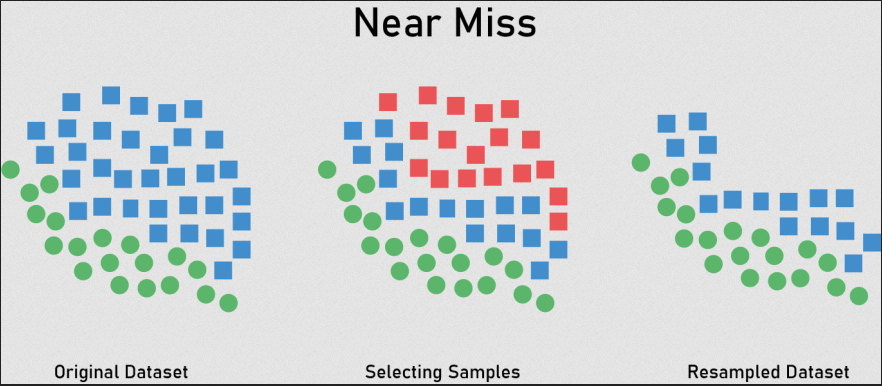
**  
**

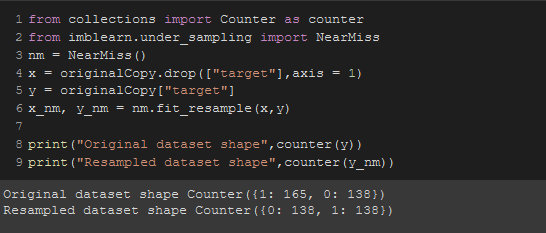
**Tomek Link:** Tomek link is an undersampling technique that are basically pairs of instances of opposite classes who are their own nearest neighbors. In other words, they are pairs of opposing instances that are very close together. Tomek's algorithm looks for such pairs and removes the majority instance of the pair.

### 

**Fig:** Tomek Links

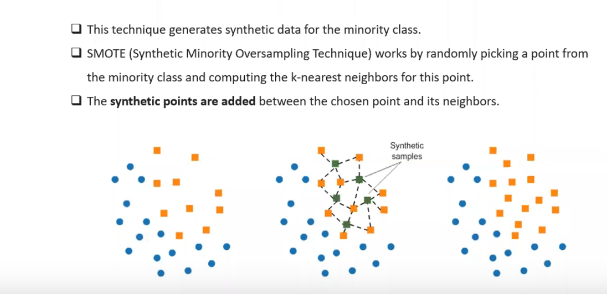
**Near miss:** Near-miss is **an algorithm that can help in balancing an imbalanced dataset**. ... When two points belonging to different classes are very close to each other in the distribution, this algorithm eliminates the datapoint of the larger class thereby trying to balance the distribution.



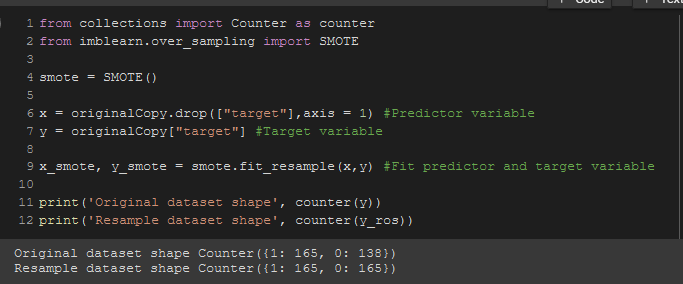


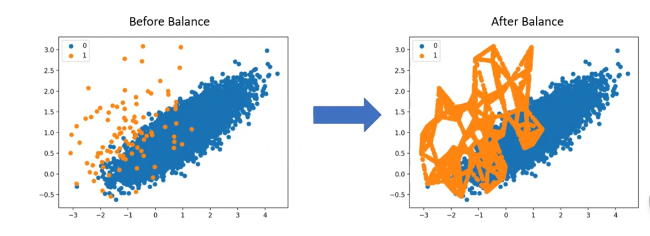
* **Over-sampling:** Oversampling refers to the resampling of the minority class points to equal the total number of majority points. Repetition of the minority class points is one such type of oversampling technique.  
  Apart from repetition, we can provide the class weights to both classes. Providing the large weights to the minority class will give the same result as a form of repetition.

**Smote:** SMOTE (Synthetic minority oversampling technique) is **an oversampling technique where the synthetic samples are generated for the minority class**. This algorithm helps to overcome the overfitting problem posed by random oversampling.



**Fig:** SMOTE

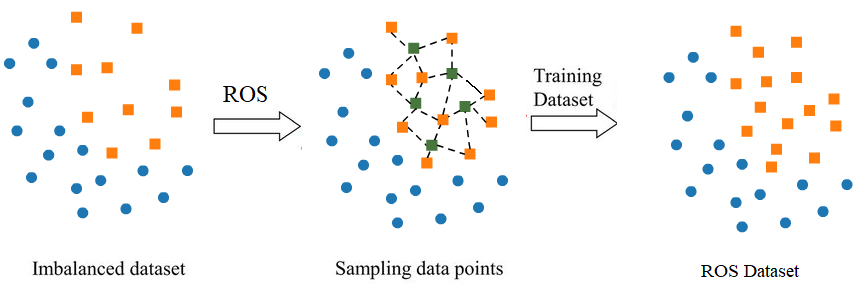


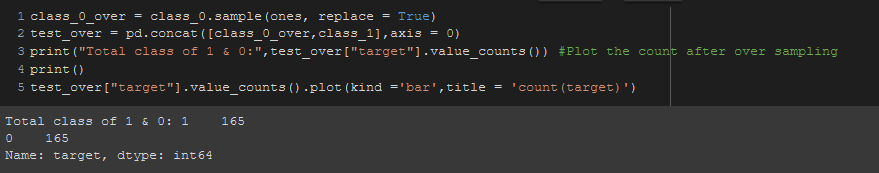
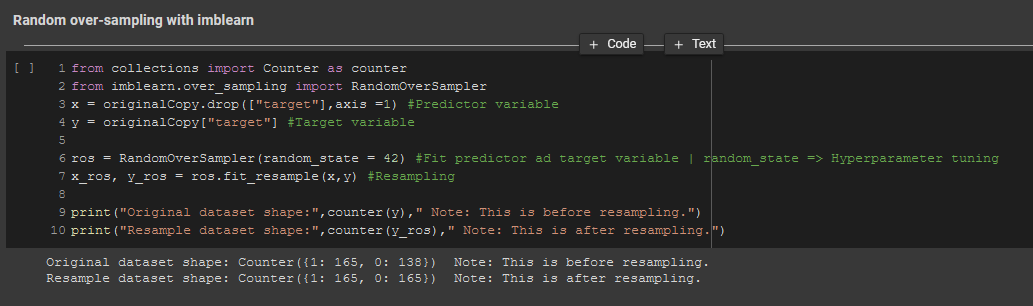


**Fig:** Oversampling

**Random over sampling**

Random oversampling involves randomly selecting examples from the minority class, with replacement, and adding them to the training dataset. Random undersampling involves randomly selecting examples from the majority class and deleting them from the training dataset.

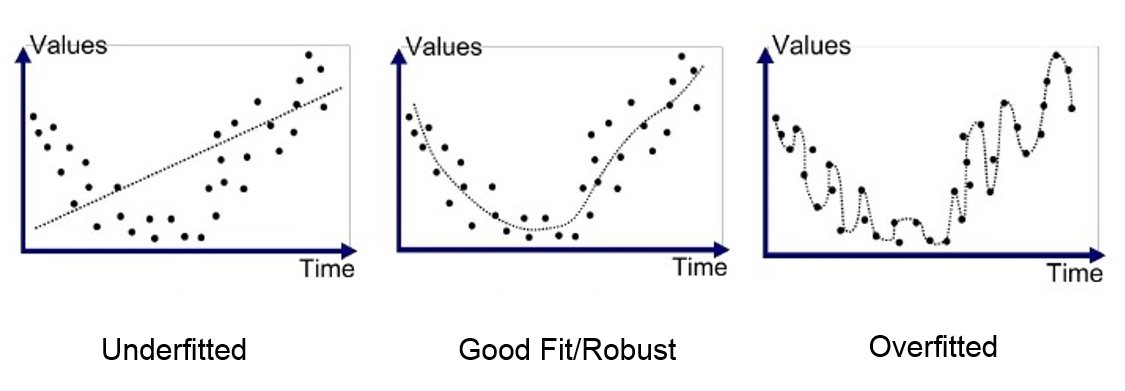


**3. Use K-fold Cross-Validation in the right way**

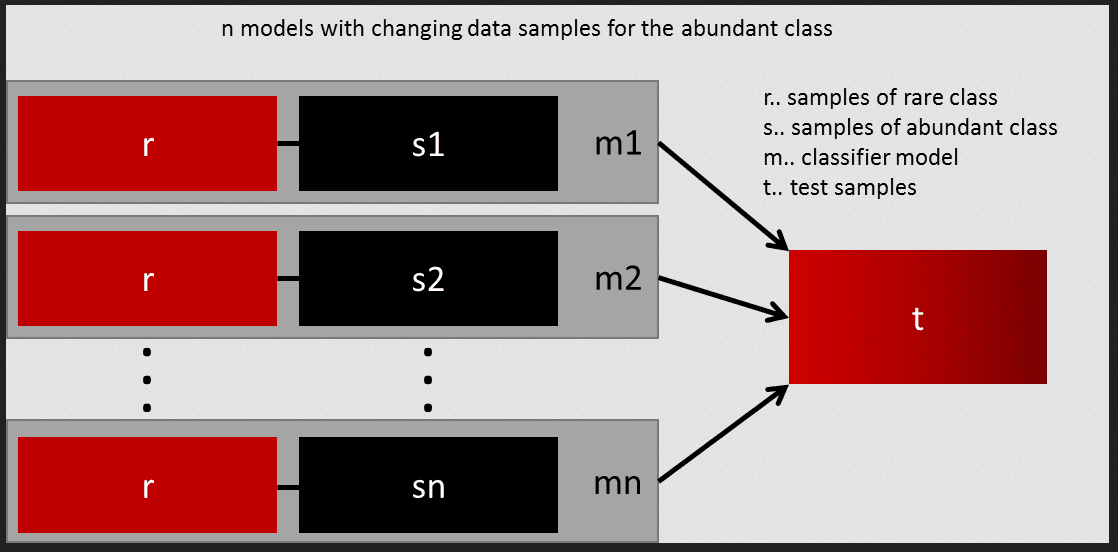
It is noteworthy that cross-validation should be applied properly while using the over-sampling method to address imbalance problems.

Keep in mind that over-sampling takes observed rare samples and applies bootstrapping to generate new random data based on a distribution function.   
If cross-validation is applied after over-sampling, basically what we are doing is overfitting our model to a specific artificial bootstrapping result.   
**That is why cross-validation should always be done before over-sampling the data, just as how feature selection should be implemented.** Only by resampling the data repeatedly, randomness can be introduced into the dataset to make sure that there won’t be an overfitting problem.



### **4. Ensemble different resampled datasets**

The easiest way to successfully generalize a model is by using more data. The problem is that out-of-the-box classifiers like logistic regression or random forest tend to generalize by discarding the rare class.   
One easy best practice is building n models that use all the samples of the rare class and n-differing samples of the abundant class.   
Given that you want to ensemble 10 models, you would keep e.g. the 1000 cases of the rare class and randomly sample 10000 cases of the abundant class. Then you just split the 10.000 cases into 10 chunks and train 10 different models.

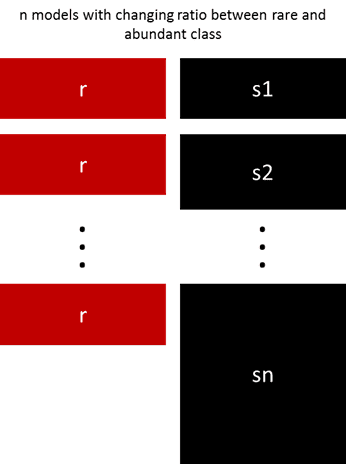


This approach is simple and perfectly horizontally scalable if you have a lot of data since you can just train and run your models on different cluster nodes. Ensemble models also tend to generalize better, which makes this approach easy to handle.

### 

### **5. Resample with different ratios**

The previous approach can be fine-tuned by playing with the ratio between the rare and the abundant class. The best ratio heavily depends on the data and the models that are used. But instead of training all models with the same ratio in the ensemble, it is worth trying to ensemble different ratios. So if 10 models are trained, it might make sense to have a model that has a ratio of 1:1 (rare: abundant) and another one with 1:3, or even 2:1. Depending on the model used this can influence the weight that one class gets.



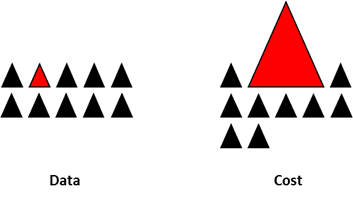
### **6. Cluster the abundant class**

Instead of relying on random samples to cover the variety of the training samples, he suggests clustering the abundant class in **“r”** groups, with **“r”** being the number of cases in **“r”**. **For each group, only the medoid (center of cluster) is kept. The model is then trained with the rare class and the medoids only.**

### **7. Design your own models**

All the previous methods focus on the data and keep the models as a fixed component. But in fact, there is no need to resample the data if the model is suited for imbalanced data.   
The famous XGBoost is already a good starting point if the classes are not skewed too much, because it internally takes care that the bags it trains on are not imbalanced. But then again, the data is resampled, it is just happening secretly.

By designing a cost function that is penalizing the wrong classification of the rare class more than the wrong classifications of the abundant class, it is possible to design many models that naturally generalize in favor of the rare class.   
For example, tweaking an SVM to penalize wrong classifications of the rare class by the same ratio that this class is underrepresented.

****