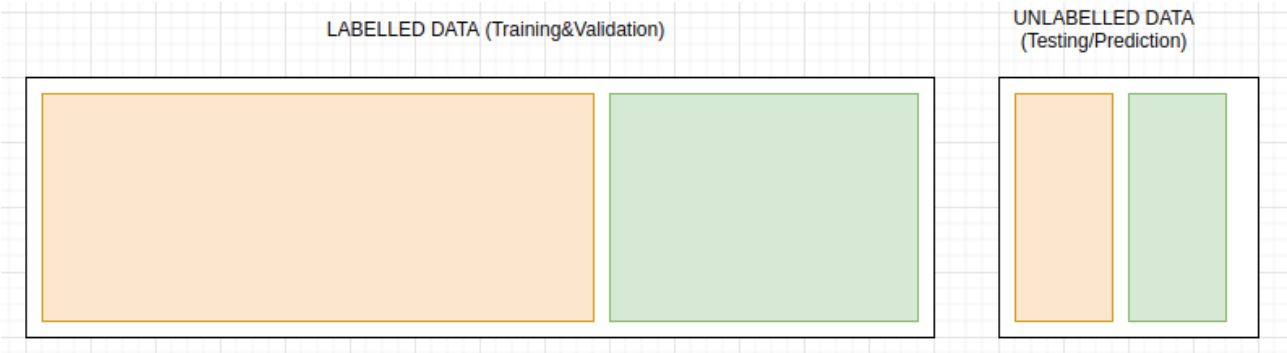
CLASSIFICATION WORKFLOW FOR IMBALANCED DATA

Here I describe the general classification workflow you may take to tackle the assignment. The major challenge with the assignment is the class imbalance, requiring you to perform extra steps in order to produce good prediction.

To have good prediction, the number one principle is to avoid information leaking from the validation set to the training set.

First, I assume that you have done proper data preparation (fixing numerous issues, remove irrelevant attributes, etc.). In the classification stage, you start with two separate sets:

- labelled set: used for training/validation
- unlabelled set: use for prediction (to submit)

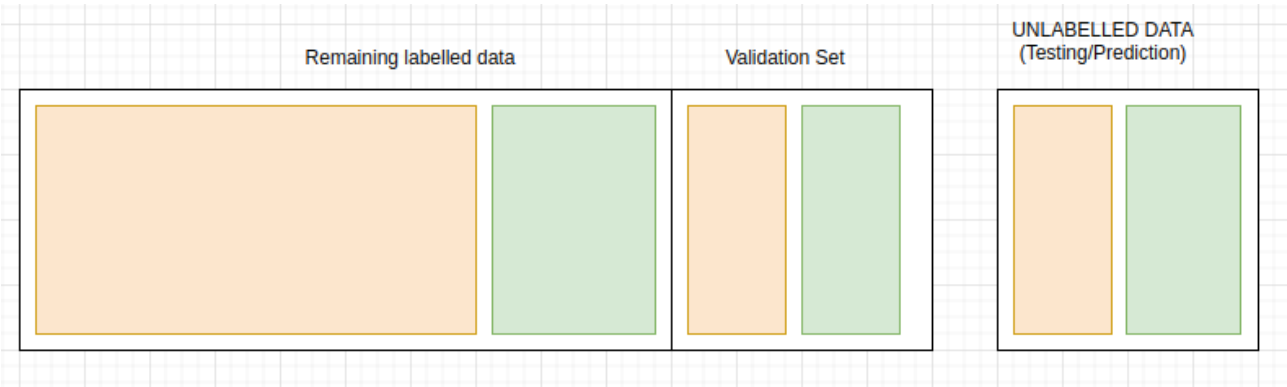


In the above diagram, I use two different colors to show that the test data is balanced whilst the labelled data is imbalanced.

Stage 1: Validation

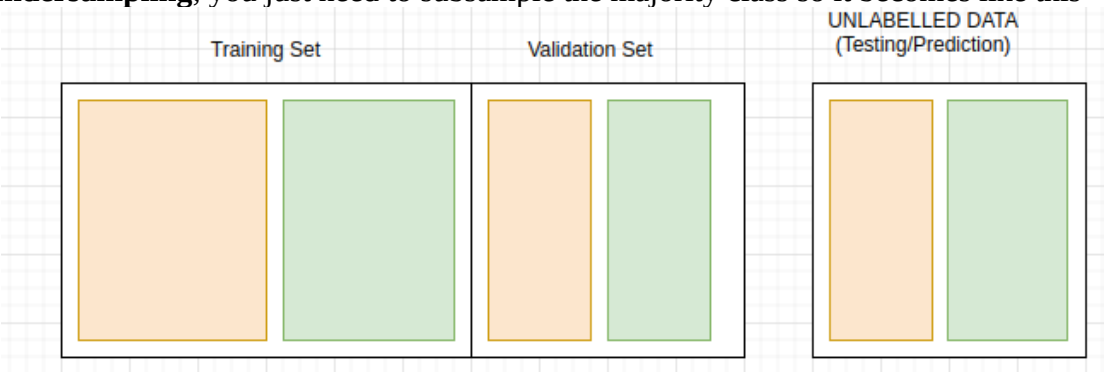
When buiding classification models you can use validation or cross-validation. I will describe the validation approach first. I will then describe what you could do to extend it to cross-validation.

With the validation approach, to avoid information leaking, you should create a validation set before applying anything to the training data (such as oversampling or attribute selection) .

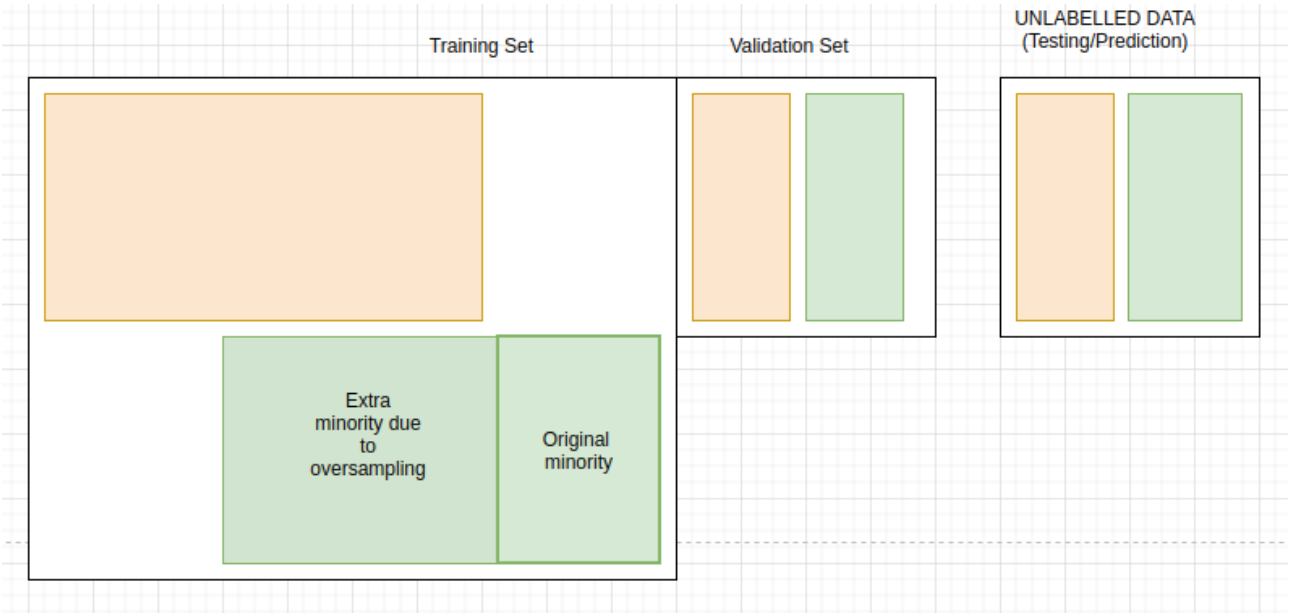


The remaining labelled data cannot be used for training straightaway because it is imbalanced. To create a training set out of the remaining labelled data, you can consider either oversampling (like SMOTE) or undersampling.

With undersampling, you just need to subsample the majority class so it becomes like this



With oversampling, you make the minority class bigger. Note that the validation set is unchanged, you should not do anything to the validation set. You may consider other things like attribute selection, but they need to be performed on the training set only, you should not involve the validation set.



Now, you can train your model on the training set and validate it on the validation set. Varying the **hyper-parameters** and fine tune your model until you've found the "best" model that produces the "best" performance on the validation set.

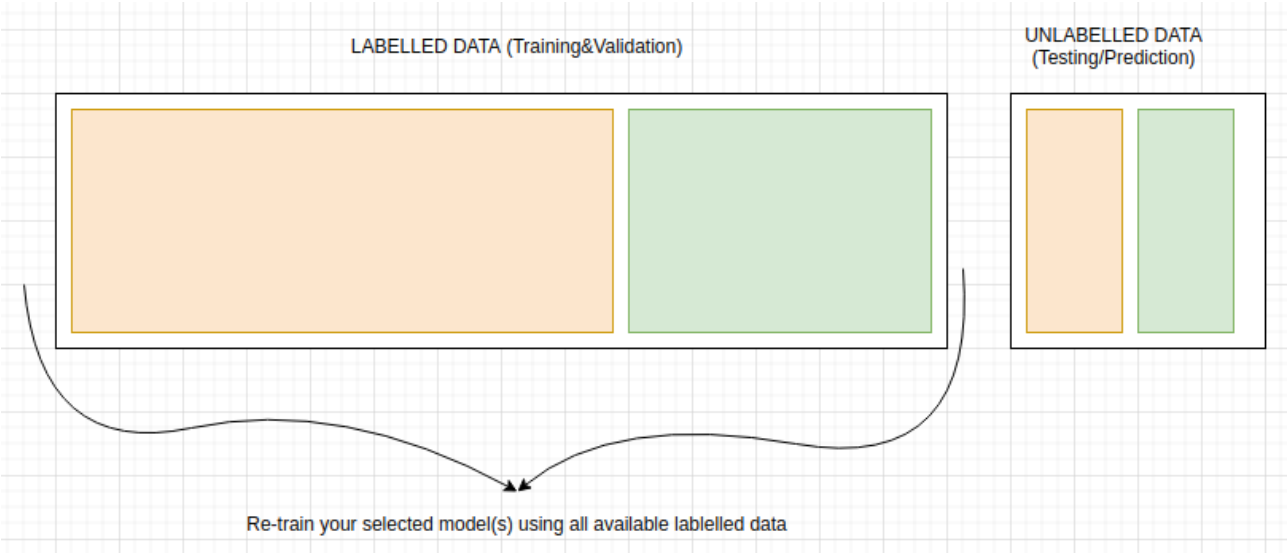
Note:

- *the hyper-parameters could be something like a True/False option, or a numeric variable that you can vary its value. You should not accept the default values and considered done. You should demonstrate an effort to fine tune the model.*
- *"best" performance should be based on accuracy, F-measure, and confusion matrix. Aim for good accuracy whilst maintaining good class balance in the validation performance. Aim for models that produce similar/consistent performance between training and validation.*
- *If you need to extend this to cross-validation, you just need to create different non-overlapping validation sets and repeat the above steps, then take the average. If you use oversampling/attribute selection you have to repeat it for each validation set.*

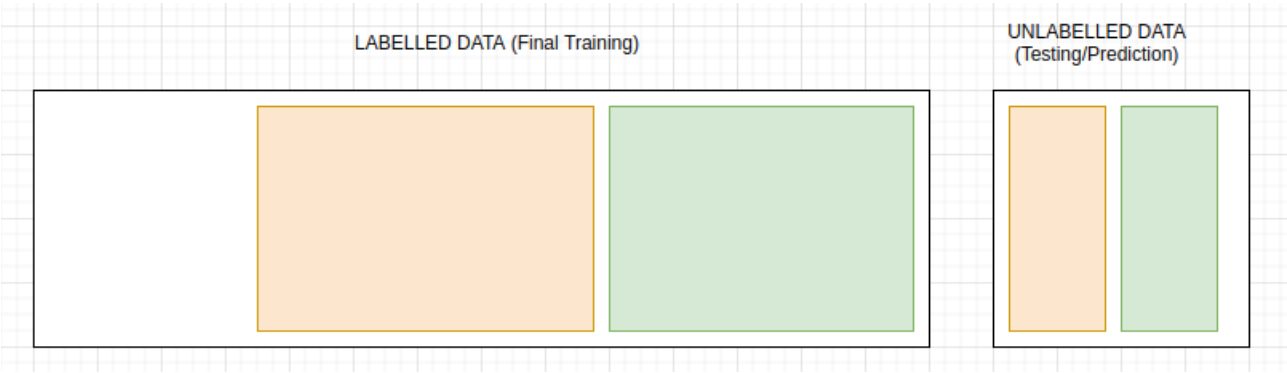
Stage 2: Producing the prediction

Now, suppose that you are happy with the model(s) you've found from validation/cross-validation. You still need to produce the prediction.

If you apply the model trained on the partial labelled data, you won't exploit the available information within the validation set described above. To get extra prediction performance you should repeat the whole process on **all the labelled data**. In other words, the whole labelled data now acts like the partial labelled data in the validation step.



Again, if you use the **undersampling** approach, you just need to sample the majority class from all the labelled data



If you use the **oversampling** approach, you need to oversample the minority class from all the labelled data – *yes you have to do it again!* If you use attribute selection, you will need to do it again too!