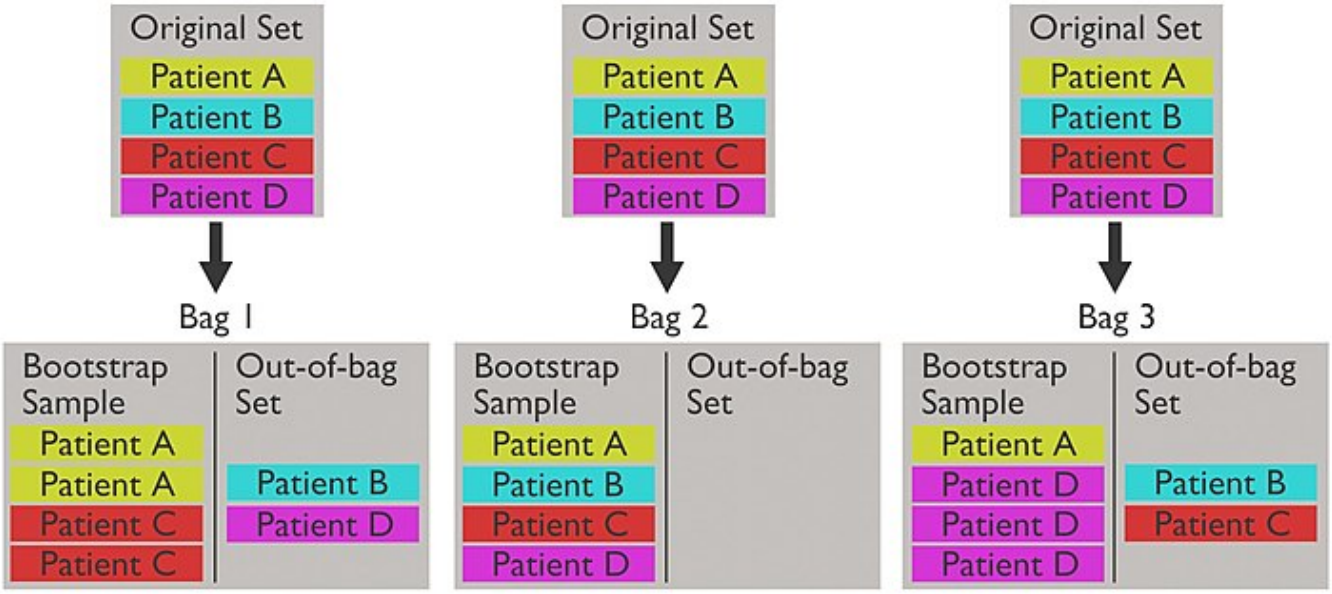
Review

Ensemble

$\epsilon_{ens} = \sum_k^n \binom{n}{k} \epsilon^k (1-\epsilon)^{n-k}$

- how is ϵ_{ens} affected by ϵ and n?
- what is the relationship betwen ϵ_{ens} and $\bar{\epsilon}$?

Out-of-bag error



- bootstrap sampling
- the $.632+$ rule:
  - a naive estimate of prediction error:
    - $e\bar{r}r = \frac{1}{N} \sum_{i=1}^N L(y_i, f(x_i))$
    - downward biased since it uses training error for estimation
  - cross validation:
    - $Err_{CV} = \frac{1}{N} \sum_{i=1}^N L(y_i, f_{-k(i)}(x_i))$
  - instead of cross-validation, we could introduce randomness using bootstrapping
    - $Err_{boot} = \frac{1}{B} \sum_{b=1}^B \frac{1}{N} \sum_{i=1}^N L(y_i, f_b(x_i))$
    - Where: $f_b(x_i)$ is the predicted value at x_i from the model fit to the b-th bootstrap dataset.
    - this is downward biased because training data is used to estimate test error...
  - Out-of-bag:

- $Err_{boot} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|C^{-i}|}\sum_{b\in C^{-i}} L(y_i, f_b(x_i))$
- Upward-biased: non-distinct observations in the bootstrap samples that result from sampling with replacement. (see lecture notes for the proof)

- $Err_{.632} = 0.368\overline{err} + 0.632Err_{boot}$
  - this might be downward-biased again if the prediction function is highly overfitted. A w is introduced to balance the two:
  - $Err_{.632} = (1-w)\overline{err} + wErr_{boot}$
  - no overfitting: R =0, w is minimized (=0.632)

Random Forest

Algorithm 1: Pseudo code for the random forest algorithm

To generate c classifiers:
for i = 1 to c **do**
  Randomly sample the training data D with replacement to produce D_i
  Create a root node, N_i containing D_i
  Call BuildTree(N_i)
end for

BuildTree(N):
if N contains instances of only one class **then**
  return
else
  Randomly select x% of the possible splitting features in N
  Select the feature F with the highest information gain to split on
  Create f child nodes of N, N_1,..., N_f, where F has f possible values ($F_1, \ldots, F_f$)
  for i = 1 to f **do**
    Set the contents of N_i to D_i, where D_i is all instances in N that match
    F_i
    Call BuildTree(N_i)
  end for
end if

(Src: https://d3i71xaburhd42.cloudfront.net/d63537ef1be0c9f71fac53805705ae78de959cf2/2-Figure1-1.png)

AdaBoost

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$.
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \to \{-1, +1\}$.
- Aim: select h_t with low weighted error:

$$\varepsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i].$$

- Choose $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right)$.
- Update, for $i = 1, \ldots, m$:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \text{sign} \left(\sum_{t=1}^{T} \alpha_t h_t(x) \right).$$

Fig. 1 The boosting algorithm AdaBoost.

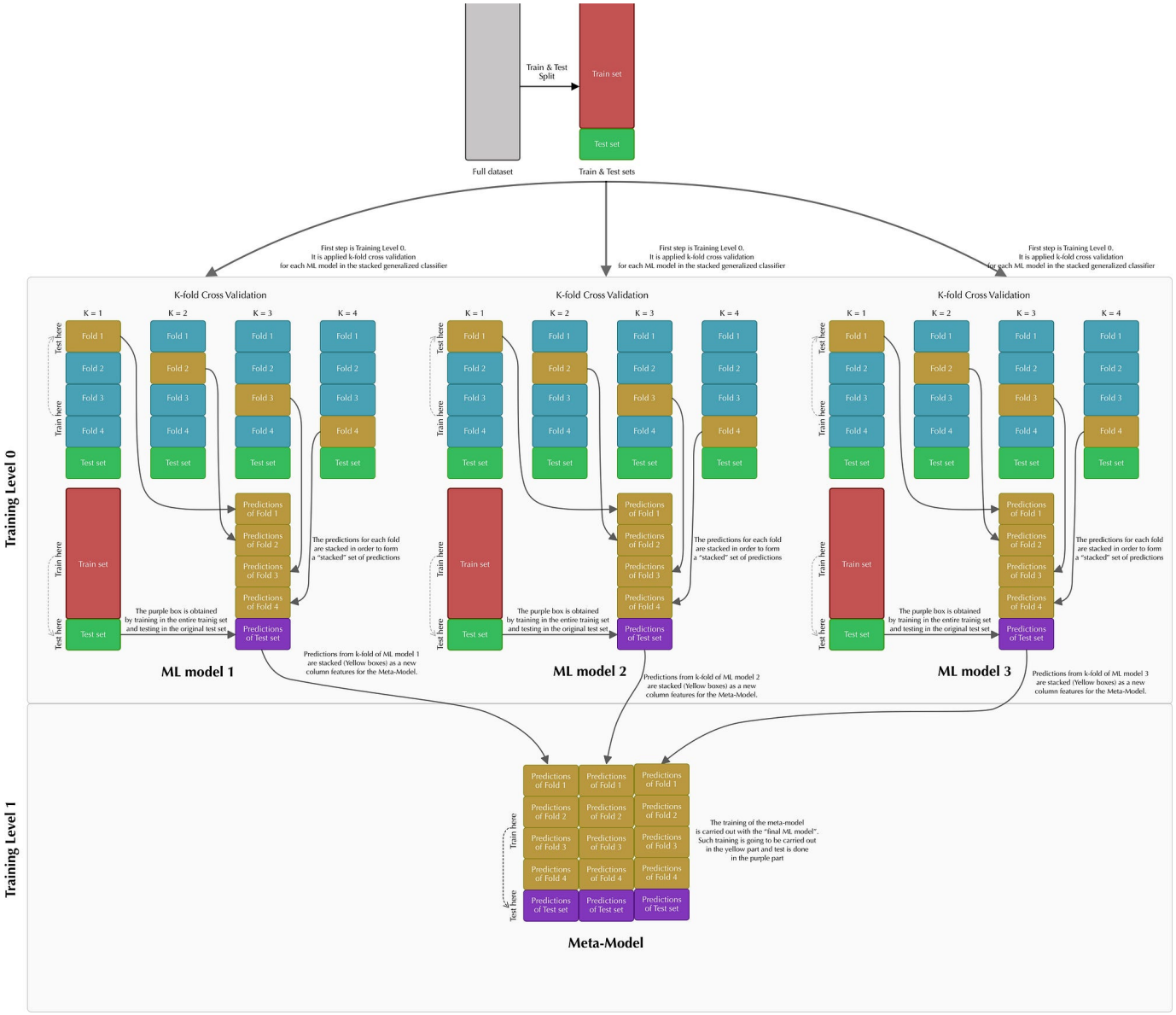
Questions

1. Ensemble Learning / Bootstrap aggregating (Bagging) / Boosting

Bagging (Bootstrap Aggregation) and **ensemble learning** are two popular techniques in machine learning for improving the accuracy and stability of predictive models.
Bagging involves creating multiple subsets of the training data by randomly sampling with replacement. Each subset is then used to train a separate instance of the model. The final predictions are made by aggregating the predictions of all the models. Bagging helps to reduce overfitting and improve the stability of the model.

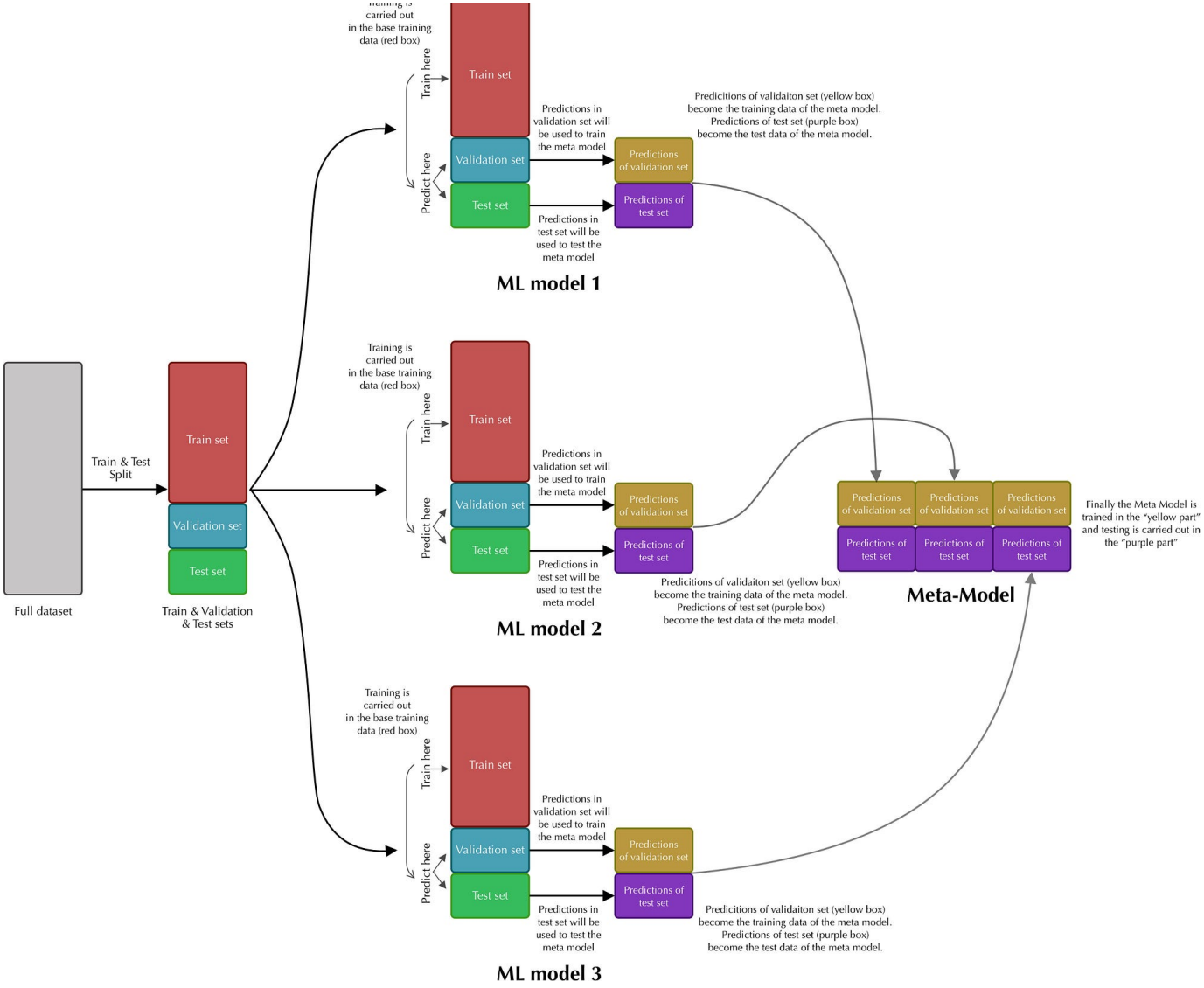
Ensemble learning is a broader concept that involves combining multiple models to improve predictive per-performance. In addition to bagging, other popular techniques for ensemble learning include boosting, stacking, and blending. Ensemble learning is particularly useful when individual models have complementary strengths and weaknesses. By combining their predictions, the resulting model can often achieve better accuracy and robustness than any individual model.

- **Stacking**: build multiple different learners (level 0), and then add a new model (level 1) which learns from the intermediate predictions (from level 0) the same target.This final model is said to be stacked on top of the others.



(src: https://miro.medium.com/v2/resize:fit:2000/1*CoauXirckomVXxw2Id2w_Q.jpeg)

- **Blending**: derived from stacking, but instead of using k-fold cross-validation it uses the 'one-holdout set' technique to generate training data.



(Src: https://miro.medium.com/v2/resize:fit:2000/format:webp/1*T0L64nrOJSr8-LRJlWfLtQ.jpeg)

1. Why is ensemble and bagging model more robust to over-fitting and outliers?
2. What is Out-Of-Bag(OOB) Error? Does that needs a hold-out test-set? Why it is considered as a pessimistic measurement of the errors

2. Boosting

Boosting is a type of ensemble learning, which combines multiple weak learners into a single strong learner. The key difference between boosting and other ensemble methods, such as bagging, is that boosting trains each weak learner sequentially, and adjusts the weights of the training data based on the errors of the previous weak learners. This allows boosting to focus more on the observations that are difficult to classify, and can result in a more accurate and robust model.

1. How is boosting different from Ensemble learning, and why is it considered better?
    - Higher accuracy: especially when dealing with complex and noisy data, by focusing more on 'difficult' instanced.

- More efficient use of resources: requires fewer base learners, by assigning adaptive weights to training data.
- Robustness: again by focusing more on the difficult-to-classify observations.
- Flexibility: classification / regression / ranking; feature selection / model interpretation...

2. Can we interpret the ensemble learning model? Why or why not?

Assignment 7

Adaboost: the toy example from lecture notes...