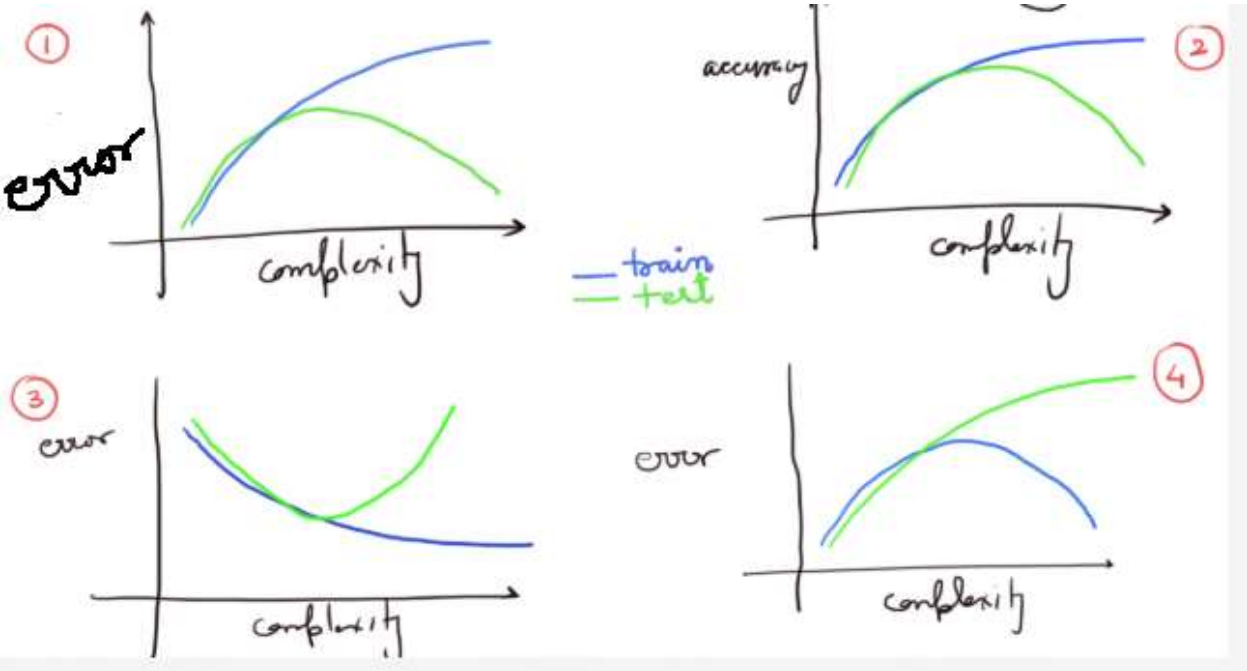
The Gini impurity index for a rectangle A is defined by

$$I(A) = 1 - \sum_{k=1}^{m} p_k^2,$$

where p_k is the proportion of observations in rectangle A that belong to class k. This measure takes values between 0 (if all the observations belong to the same class) and $(m-1)/m$ (when all m classes are equally represented). Figure 9.4



Which of the Above is Correct?

Over-fitting in Trees (Theory)

We have seen that to prevent over-fitting in Decision tree, we have to stop them from growing beyond a point. This brings us to the stopping criteria. The recursive binary splitting procedure described above needs to know when to stop splitting as it works its way down the tree with the training data.

The most common stopping procedure are discussed as follows. 2 concepts are used.

- **Pre-pruning** that stop growing the tree earlier, before it perfectly starts to overfit.
- **Post-pruning** that allows the tree to perfectly classify the training set, and then post prune the tree.

Pruning is the process of cutting the unwanted branches of a tree.

2 parameters are widely used-

Max_Depth/ Min_Samples_Leaf parameter:/ **Pre-pruning**

This is done to stop the tree from growing in the first place. The most common stopping procedure is to use a minimum count on the number of training instances assigned to each leaf node. If the count is less than some minimum then the split is not accepted and the node is taken as a final leaf/terminal node.

For example if you take Min_Samples_Leaf=5, then a Node with 5 or less than 5 data-points, will not be split further. This avoids the tree from growing further. If it is set to a too-small value, like 1, we may run the risk of over-fitting our model.

Similarly, we have the Max_Depth parameter which controls the depth of the tree. Setting max_depth to a large number may cause Overfitting.

Pruning the tree:/**Post-pruning**

This is done after allowing the tree to grow. The fastest and simplest pruning method is to work through each leaf node in the tree and evaluate the effect of removing it using a hold-out test set. Leaf nodes are removed only if it results in a drop in the overall cost function on the entire test set. You stop removing nodes when no further improvements can be made. Pruning a tree is beyond scope of this course, hence we will skip this.

We will only focus on controlling over-fitting by using minbucket parameter.

Configure Jupyter to plot Decision Trees

Step 1) Download graphviz from here and install on your system -
- https://graphviz.gitlab.io/_pages/Download/Download_windows.html

(graphviz-2.38.msi.this has to be downloaded)

Step2) then go to my computer properties--> advanced system settings--> Environment variables

Click NEW -->
#variable name ---> PATH
#variable value -----> C:\Program Files (x86)\Graphviz2.38\bin

Step3) Goto Anaconda Prompt/Windows Prompt and install pydotplus using

pip install pydotplus

Step4) Restart your system

Need for Cross Validation

The validation set approach is conceptually simple and is easy to implement. But it has two potential drawbacks:

1) The validation estimate of error rate can be highly variable, depending on precisely which observations are included in the training set and which observations are included in the validation set.

2) In the validation approach, only a subset of the observations—those that are included in the training set rather than in the validation set—are used to fit the model. Since statistical methods tend to perform worse when trained on fewer observations.

In the next section, we will present cross-validation, a refinement of the validation set approach that addresses these two issues.

Read

ISLR Pages 176-178

Leave-one-out cross-validation (LOOCV) is closely related to the validation set approach that we talked about in the last Video.

Like the validation set approach, LOOCV involves splitting the set of observations into two parts. However, instead of creating two subsets of comparable size, a single observation (x1, y1) is used for the validation set, and the remaining observations {(x2, y2), . . . , (xn, yn)} make up the training set. The statistical learning method is fit on the n − 1 training observations, and a prediction yˆ1 is made for the excluded observation, using its value x1.

We can repeat the procedure by selecting (x2, y2) for the validation data, training the statistical learning procedure on the n − 1 observations {(x1, y1), (x3, y3), . . . , (xn, yn)}.

LOOCV has the potential to be expensive to implement, since the model has to be fit n times. This can be very time consuming if n is large, and if each individual model is slow to fit.

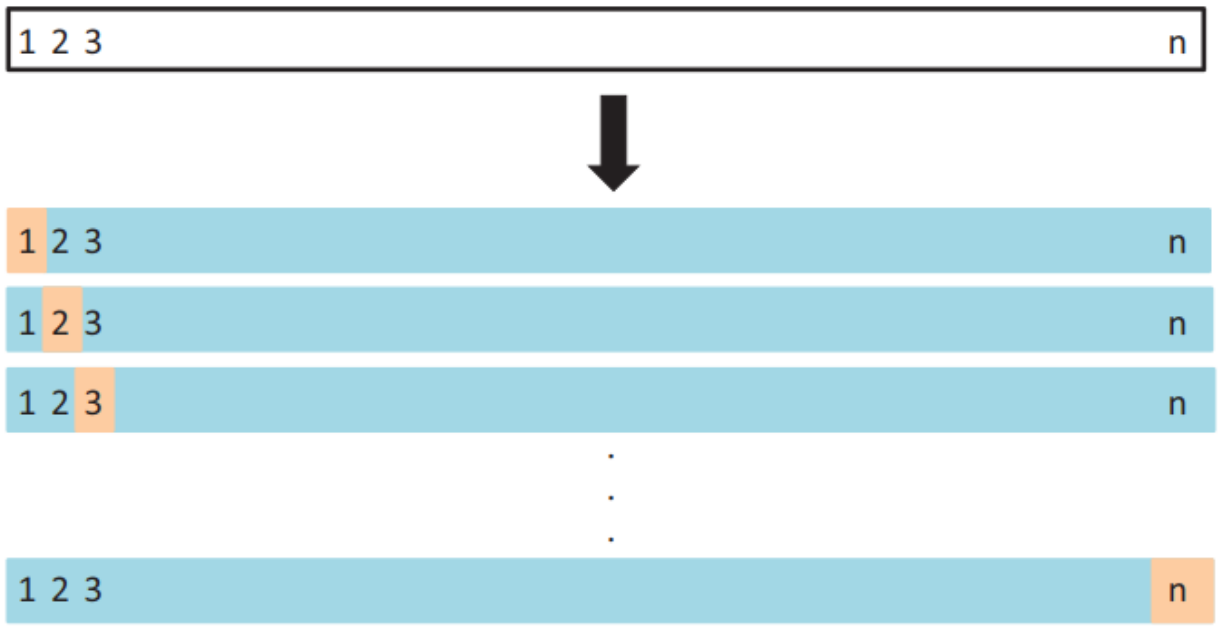


Figure-- A set of n data points is repeatedly split into a training set (shown in blue) containing all but one observation, and a validation set that contains only that observation (shown in beige)

An alternative to LOOCV is k-fold CV. This approach involves randomly dividing the set of observations into k groups, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining k − 1 folds. The mean squared error, MSE_1, is then computed on the observations in the held-out fold. This procedure is repeated k times; each time, a different group of observations is treated as a validation set. The Error is computed by taking a Average of all K Errors.

It is not hard to see that LOOCV is in fact a special case of k-fold CV in which k is set to equal n.

In practice, one typically performs k-fold CV using k = 5 or k = 10. What is the advantage of using k = 5 or k = 10 rather than k = n? The most obvious advantage is computational efficiency. LOOCV requires fitting the statistical learning method n times. This has the potential to be computationally expensive (except for linear models fit by least squares, in which case formula (5.2) can be used). In contrast, performing 10-fold CV requires fitting the learning procedure only ten times, which may be much more feasible.

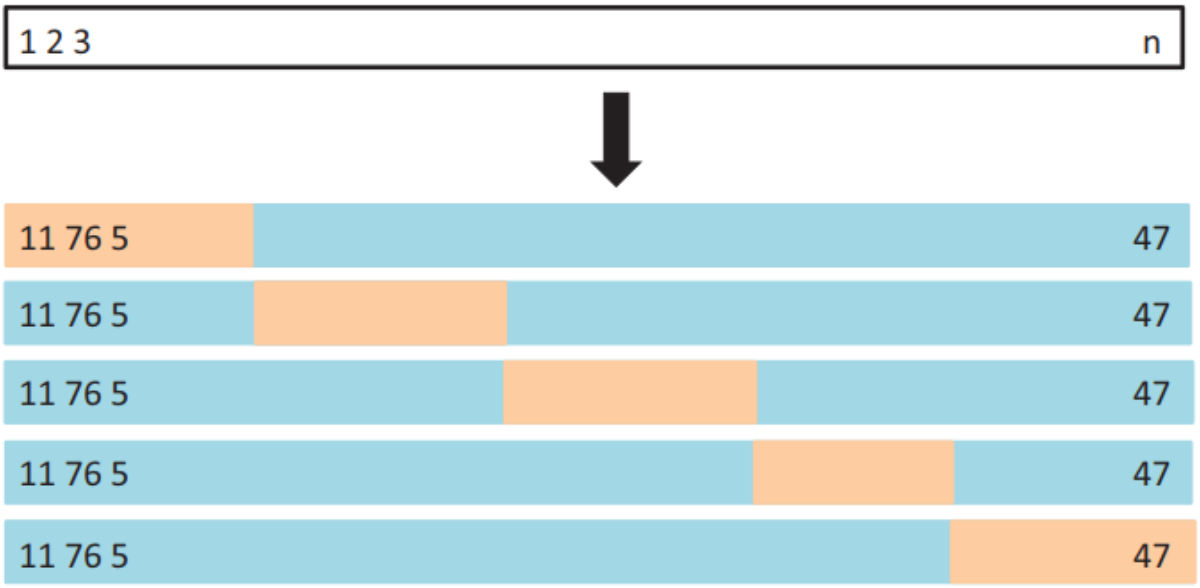


Figure -- A schematic display of 5-fold CV. A set of n observations is randomly split into five non-overlapping groups. Each of these fifths acts as a validation set (shown in beige), and the remainder as a training set (shown in blue). The test error is estimated by averaging the five resulting MSE estimates.

Tree Advantages & Disadvantages

Easy to Understand: Decision tree output is very easy to understand even for people from non-analytical background. It does not require any statistical knowledge to read and interpret them.

Useful in Data exploration: Decision tree is one of the fastest way to identify most significant variables and relation between two or more variables. With the help of decision trees, we can create new variables / features that has better power to predict target variable.

Less data cleaning required: It requires less data cleaning compared to some other modeling techniques. It is not influenced by outliers and missing values to a fair degree.

Data type is not a constraint: It can handle both numerical and categorical variables.

Non Parametric Method: Decision tree is considered to be a non-parametric method. This means that decision trees have no assumptions about the space distribution and the classifier structure.

Disadvantages:-

Over fitting: Over fitting is one of the most practical difficulty for decision tree models. This problem gets solved by using CV.

Gini Index | Entropy | Information Gain

While GINI Index is the most popular Splitting criteria for Decision Trees in case of classification (it is SSE for Regression Trees), there are 2 more parameters used-

Entropy:-

The second impurity measure is Entropy. The entropy of a Bucket A is defined as

$$\text{Entropy } (A) = - \sum p_k \log_2(p_k)$$

This measure ranges between 0 (for a pure bucket) and $\log_2(m)$ when m classes are equally represented. (1 when m=2)

p_k is the proportion of observations in bucket A that belong to class k.

Information gain:-

Information Gain is similar to Entropy and can be calculated using the formula-

$$\text{Information Gain} = 1 - \text{Entropy}$$