PW SKILLS

Lesson Plan

Decision Tree

Topics Covered

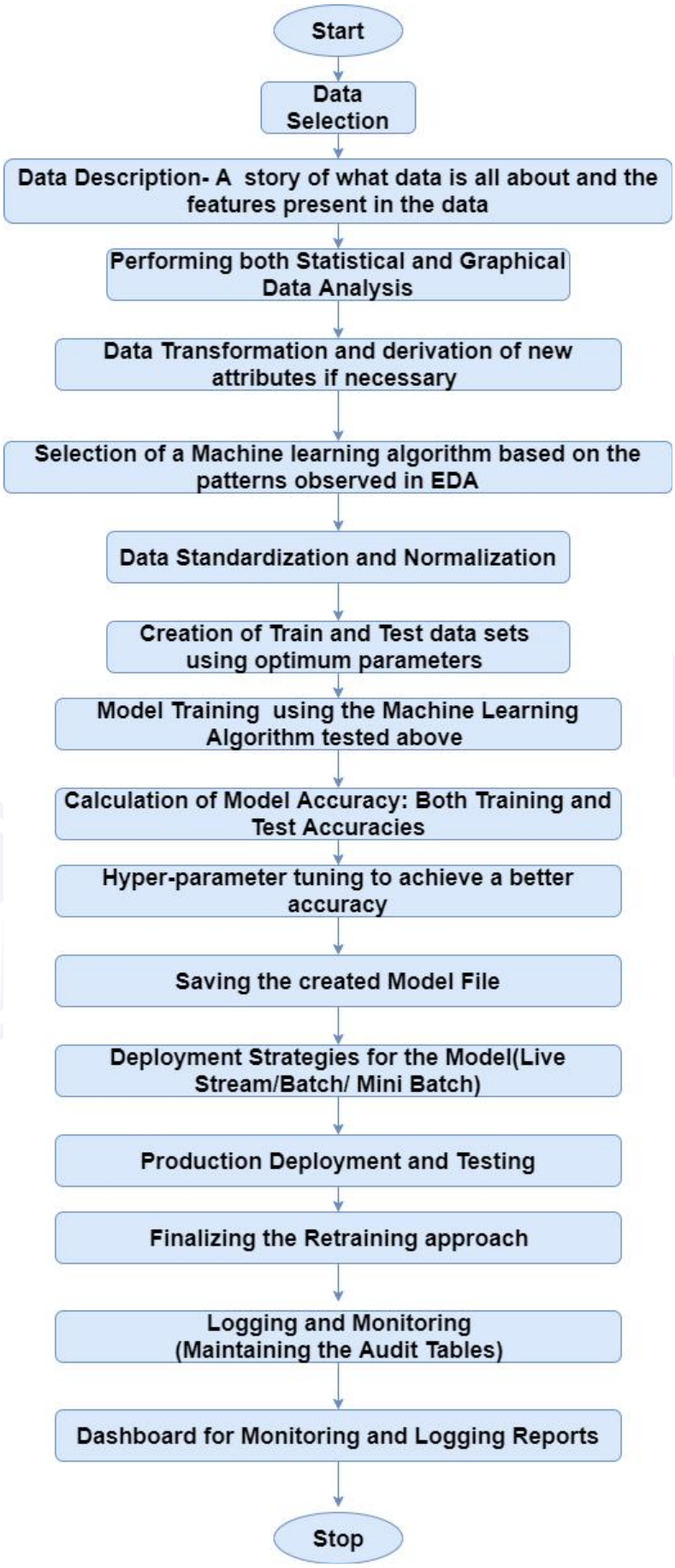
- Application Flow
- Introduction to Decision Tree
- Decision Tree for Regression
- Recursive binary splitting(Greedy approach)
- Tree Pruning
- Post-pruning
- Pre-pruning
- Classification Trees
- Entropy
- Information Gain
- Ginni Impurity
- Different Algorithms for Decision Tree
- Advantages of Decision Tree:
- Disadvantages of Decision Tree:
- Cross-Validation
- Bias Variance tradeoff for k-fold CV, LOOCV and Holdout Set
- CV

Application Flow

Decision Tree is one of the most fundamental algorithms for classification and regression in the Machine Learning world.

But before proceeding with the algorithm, let's first discuss the lifecycle of any machine learning model. This diagram explains the creation of a Machine Learning model from scratch and then taking the same model further with hyperparameter tuning to increase its accuracy, deciding the deployment strategies for that model and once deployed setting up the logging and monitoring frameworks to generate reports and dashboards based on the client requirements.

A typical lifecycle diagram for a machine learning model looks like:

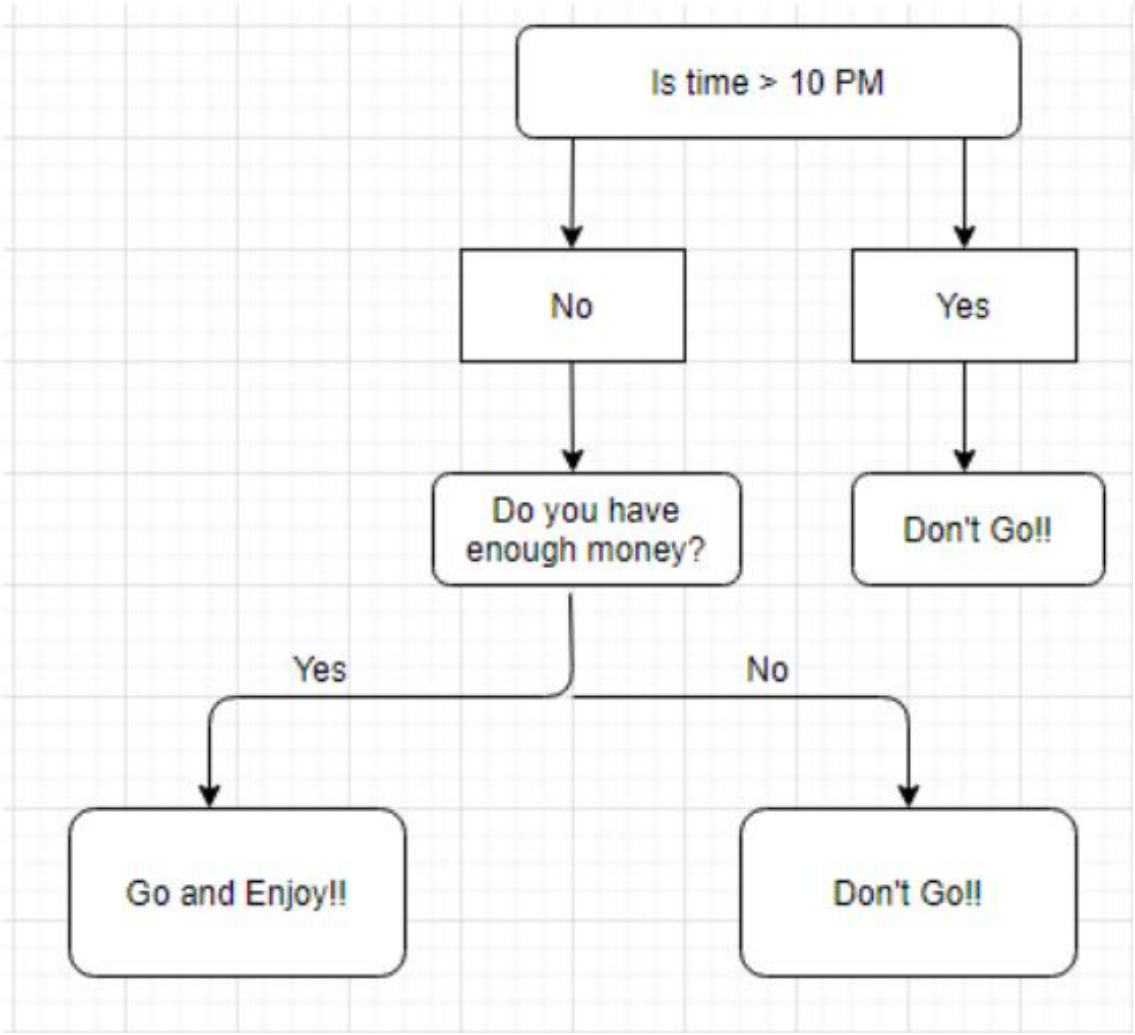

Start
Data Selection
Data Description- A story of what data is all about and the features present in the data
Performing both Statistical and Graphical Data Analysis
Data Transformation and derivation of new attributes if necessary
Selection of a Machine learning algorithm based on the patterns observed in EDA
Data Standardization and Normalization
Creation of Train and Test data sets using optimum parameters
Model Training using the Machine Learning Algorithm tested above
Calculation of Model Accuracy: Both Training and Test Accuracies
Hyper-parameter tuning to achieve a better accuracy
Saving the created Model File
Deployment Strategies for the Model(Live Stream/Batch/ Mini Batch)
Production Deployment and Testing
Finalizing the Retraining approach
Logging and Monitoring (Maintaining the Audit Tables)
Dashboard for Monitoring and Logging Reports
Stop

Decision Tree

Decision tree algorithm is one of the most versatile algorithms in machine learning which can perform both classification and regression analysis. It is very powerful and works great with complex datasets. Apart from that, it is very easy to understand and read. That makes it more popular to use. When coupled with ensemble techniques – which we will learn very soon- it performs even
better.

As the name suggests, this algorithm works by dividing the whole dataset into a tree-like structure based on some rules and conditions and then gives prediction based on those conditions.
Let's understand the approach to decision tree with a basic scenario.

Suppose it's Friday night and you are not able to decide if you should go out or stay at home. Let the decision tree decide it for you.



Although we may or may not use the decision tree for such decisions, this was a basic example to help you understand how a decision tree makes a decision. So how did it work?

- It selects a root node based on a given condition, e.g. our root node was chosen as time 10 pm.
- Then, the root node was split into child notes based on the given condition. The right child node in the above figure fulfilled the condition, so no more questions were asked.
- The left child node didn't fulfil the condition, so again it was split based on a new condition.
- This process continues till all the conditions are met or if you have
- predefined the depth of your tree, e.g. the depth of our tree is 3, and it reached there when all the conditions were exhausted.
- Let's see how the parent nodes and condition is chosen for the splitting to work.

Decision Tree for Regression

When performing regression with a decision tree, we try to divide the given values of X into distinct and non-overlapping regions, e.g. for a set of possible values X1, X2,..., Xp; we will try to divide them into J distinct and non- overlapping regions R1, R2, . . . , RJ.

For a given observation falling into the region Rj, the prediction is equal to the mean of the response(y) values for each training observations(x) in the region Rj.

The regions R1,R2, . . . , RJ are selected in a way to reduce the following sum of squares of residuals :

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2,$$

Where, yrj (second term) is the mean of all the response variables in the region 'j'.

Recursive binary splitting(Greedy approach)

As mentioned above, we try to divide the X values into j regions, but it is very expensive in terms of computational time to try to fit every set of X values into j regions. Thus, decision tree opts for a top-down greedy approach in which nodes are divided into two regions based on the given condition, i.e. not every node will be split but the ones which satisfy the condition are split into two branches. It is called greedy because it does the best split at a given step at that point of time rather than looking for splitting a step for a better tree in upcoming steps. It decides a threshold value(say s) to divide the observations into different regions(j) such that the RSS for Xj>= s and Xj <s is minimum.

$$\sum_{i:\, x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\, x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

Here for the above equation, j and s are found such that this equation has the minimum value.
The regions R1, R2 are selected based on that value of s and j such that the equation above has the minimum value.

Similarly, more regions are split out of the regions created above based on some condition with the same logic. This continues until a stopping criterion (predefined) is achieved.

Once all the regions are split, the prediction is made based on the mean of observations in that region.
The process mentioned above has a high chance of overfitting the training data as it will be very complex.

Tree Pruning

Tree pruning is the method of trimming down a full tree (obtained through the above process) to reduce the complexity and variance in the data. Just as we
regularised linear regression, we can also regularise the decision tree model by adding a new term.

$$\sum_{m=1}^{|T|} \sum_{i:\ x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

Where, T is the subtree which is a subset of the full tree T0
And α is the non-negative tuning parameter which penalises the MSE with an increase in tree length.
By using cross-validation, such values of α and T are selected for which our model gives the lowest test error rate.

This is how the decision tree regression model works. Let's now see the working algorithm of doing classification using a decision tree.

Greedy Algorithm
As per Hands-on machine learning book "greedy algorithm greedily searches for an optimum split at the top level, then repeats the process at each level. It does not check whether or not the split will lead to the lowest possible impurity several levels down. A greedy algorithm often produces a reasonably good
solution, but it is not guaranteed to be the optimal solution."

Post-pruning

Post-pruning, also known as backward pruning, is the process where the decision tree is generated first and then the non-significant branches are removed. Cross-validation set of data is used to check the effect of pruning and tests whether expanding a node will make an improvement or not. If any improvement is there then we continue by expanding that node else if there is reduction in accuracy then the node not be expanded and should be converted in a leaf node.

Pre-pruning

Pre-pruning, also known as forward pruning, stops the non-significant branches from generating. It uses a condition to decide when should it terminate splitting of some of the branches prematurely as the tree is generated.

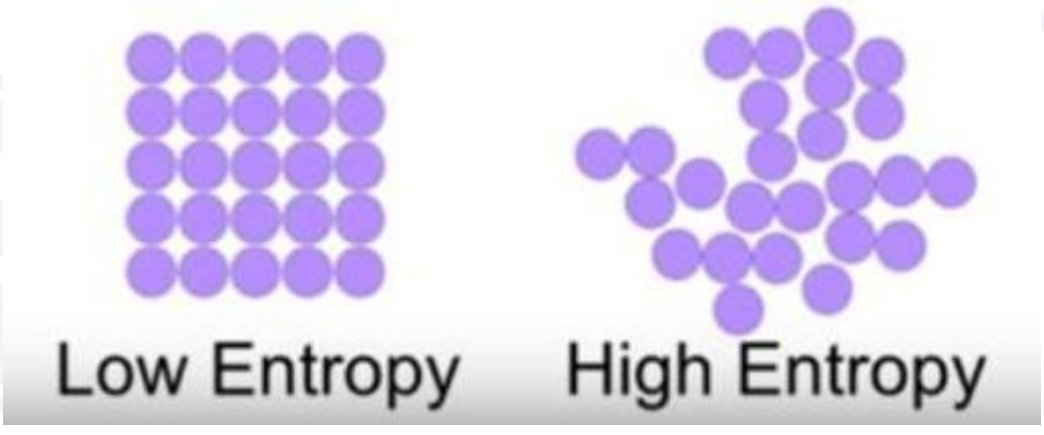
Classification Trees

Regression trees are used for quantitative data. In the case of qualitative data or categorical data, we use classification trees. In regression trees, we split the nodes based on RSS criteria, but in classification, it is done using classification error rate, Gini impurity and entropy.

Let's understand these terms in detail.

Entropy

Entropy is the measure of randomness in the data. In other words, it gives the impurity present in the dataset.



When we split our nodes into two regions and put different observations in both the regions, the main goal is to reduce the entropy i.e. reduce the randomness in the region and divide our data cleanly than it was in the previous node. If splitting the node doesn't lead into entropy reduction, we try to split based on a different condition, or we stop.

A region is clean (low entropy) when it contains data with the same labels and random if there is a mixture of labels present (high entropy).

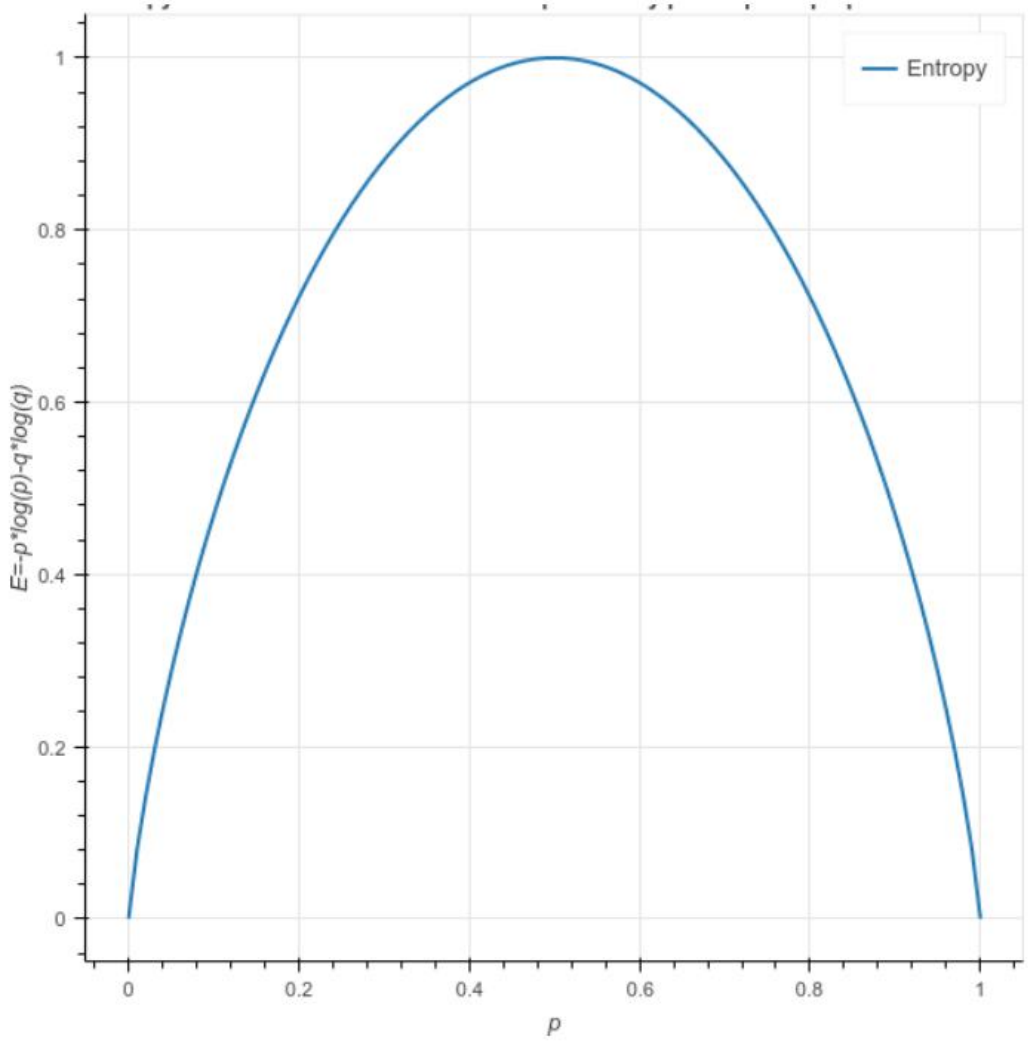
Let's suppose there are 'm' observations and we need to classify them into categories 1 and 2.
Let's say that category 1 has 'n' observations and category 2 has 'm-n' observations.

p= n/m and q = m-n/m 1-p
then, entropy for the given set is:

```
E = -p*log2(p) - q*log2(q)
```

When all the observations belong to category 1, then p 1 and all observations belong to category 2, then p 0, int both cases E 0, as there is no randomness in the categories.

If half of the observations are in category 1 and another half in category 2, then p 1/2 and q 1/2, and the entropy is maximum, E 1.

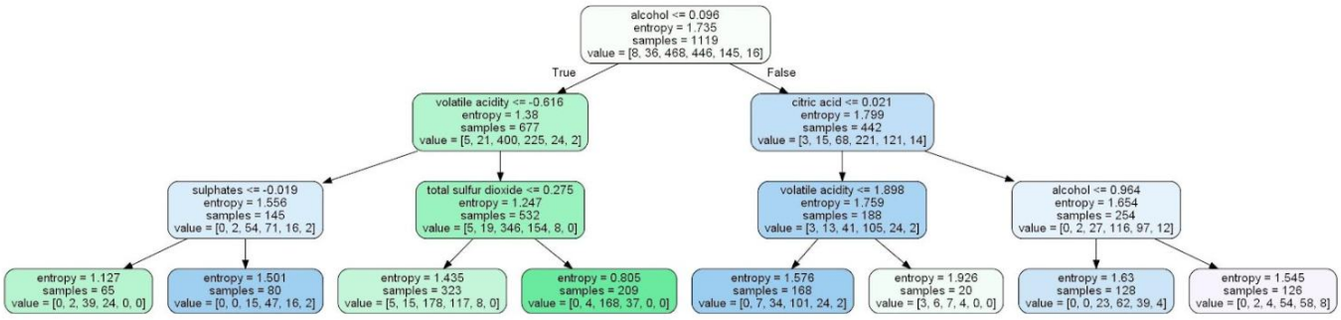


Information Gain

Information gain calculates the decrease in entropy after splitting a node. It is the difference between entropies before and after the split. The more the information gain, the more entropy is removed.

$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

Where, T is the parent node before split and X is the split node from T.
A tree which is splitted on basis of entropy and information gain value looks like:

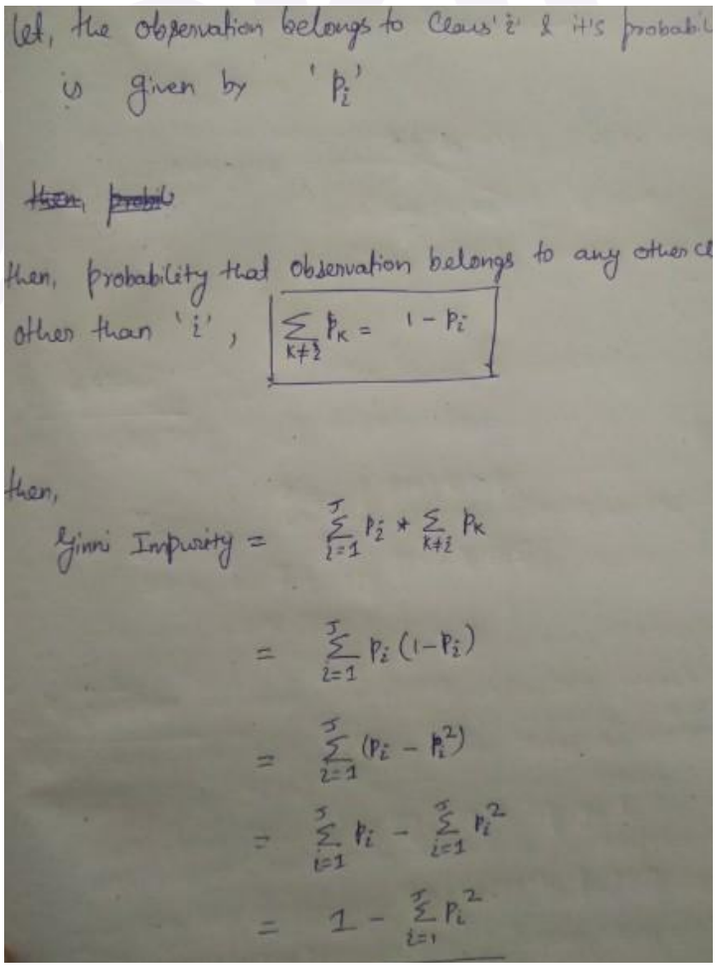


Ginni Impurity

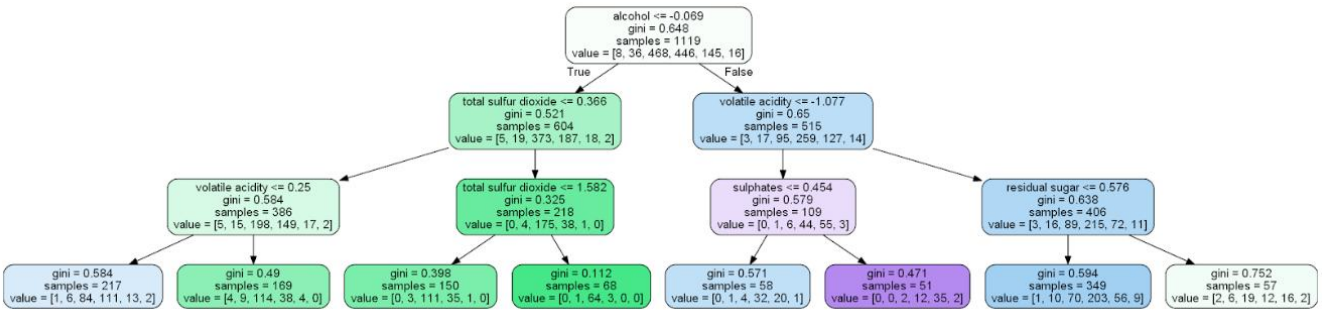
According to wikipedia, 'Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labelled if it was randomly labelled according to the distribution of labels in the subset.'

It is calculated by multiplying the probability that a given observation is classified into the correct class and sum of all the probabilities when that particular observation is classified into the wrong class.

Let's suppose there are k number of classes and an observation belongs to the class 'i', then Ginni impurity is given as:



Ginni impurity value lies between 0 and 1, 0 being no impurity and 1 denoting random distribution.
The node for which the Ginni impurity is least is selected as the root node to split.
A tree which is splitted on basis of ginni impurity value looks like:



Maths behind Decision Tree Classifier

Before we see the python implementation of decision tree. let's first understand the math behind the decision tree classfication. We will see how all the above mentioned terms are used for splitting.

We will use a simple dataset which contains information about students of different classes and gender and see whether they stay in school's hostel or not.

This is how our data set looks like :

Class	Gender	Stay in hostel
9	M	Yes
10	F	No
8	F	Yes
8	F	No
9	M	Yes
10	M	No
11	F	Yes
11	M	Yes
8	F	Yes
9	M	No
11	M	No
11	M	Yes
10	F	No
10	M	Yes

Let's try and understand how the root node is selected by calcualting gini impurity. We will use the above mentioned data.

We have two features which we can use for nodes: "Class" and "Gender". We will calculate gini impurity for each of the features and then select that feature which has least gini impurity.

Let's review the formula for calculating ginni impurity:

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

Let's start with class, we will try to gini impurity for all different values in "class".

Class	Stay in hostel	(m) Total values	P(yes)	P(NO)
8	Yes=2, NO=1	3	2/3	1/3
9	Yes=2, NO=1	3	2/3	1/3
10	Yes=1, NO=3	4	1/4	3/4
11	Yes=3, NO=1	4	3/4	1/4
		M=14		

Let's Calculate Gini impurity for "Class" feature

$$G_I(\text{class}=8) = 1 - [P(\text{yes})]^2 - [P(\text{NO})]^2$$

$$= 1 - (2/3)^2 - (1/3)^2$$

$$= 4/9$$

$$G_I(\text{class}=9) = 1 - (2/3)^2 - (1/3)^2$$

$$= 4/9$$

$$G_I(\text{class}=10) = 1 - (1/4)^2 - (3/4)^2$$

$$= 6/16$$

$$G(\text{class}=11) = 1 - (3/4)^2 - (1/4)^2$$

$$= 6/16$$

Weighted Sum of Gini Impurities for Class features

$$G(\text{class}) = \frac{\text{no. of Instance (class=8)}}{\text{total Instances}} * G(\text{class}=8) + \frac{n(\text{class}=9)}{m} \times G(\text{class}=9)$$

(no. of Instance (class=8): let's call it 'n'; total Instances: let's call it 'm')

$$+ \frac{n(\text{class}=10)}{m} * G(\text{class}=10)$$

$$+ \frac{n(\text{class}=11)}{m} * G(\text{class}=11)$$

$$= \frac{8}{14} \times \frac{4}{9} + \frac{3}{14} \times \frac{4}{9} + \frac{4}{14} \times \frac{6}{16} + \frac{4}{14} \times \frac{6}{16}$$

$$= 0.19 + 0.214 = 0.404$$

Let's calculate Gini Impurity for "Gender" Feature

Gender	Stay on hostel	n	P(Yes)	P(No)
M	Yes = 5, NO = 3	8	5/8	3/8
F	Yes = 3, NO = 3	6	3/6 = 1/2	1/2
		m = 14		

$G_1(\text{Gender} = M) = \frac{8}{14} \times 1 - (5/8)^2 - (3/8)^2$

$= 1 - 0.39 - 0.14$

$G_1(\text{Gender} = F) = 1 - (1/2)^2 - (1/2)^2$

$= 0.5$

$G_1(\text{Gender}) = \frac{8}{14} \times 0.47 + \frac{6}{14} \times 0.5$

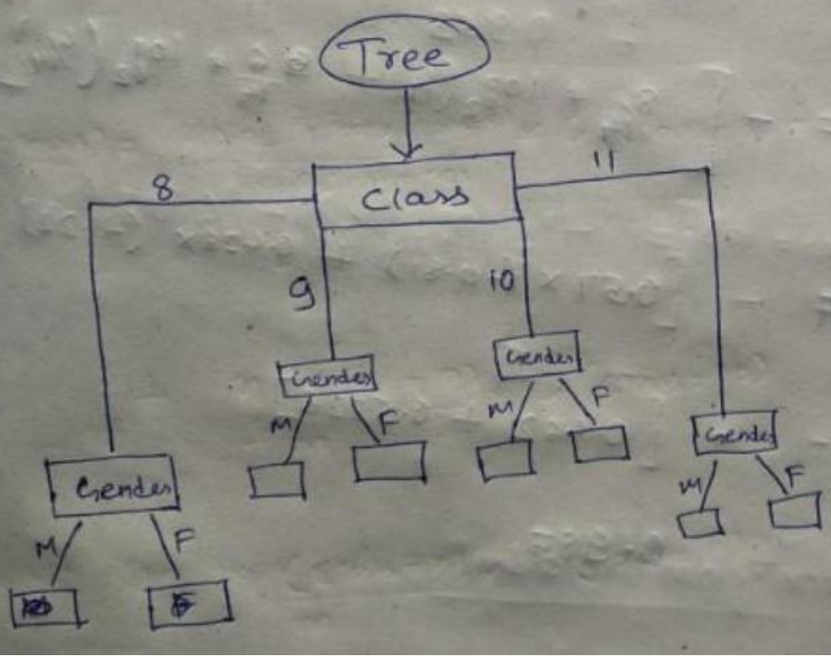
$= 0.268 + 0.214$

$= 0.4822$

We can see that,

$G_1(\text{class}) < G_1(\text{Gender})$

Thus, our root node is class.



This is how our Decision tree node is selected by calculating gini impurity for each node individually.
If the number of feautures increases, then we just need to repeat the same steps after the selection of the root node.

We will try and find the root nodes for the same dataset by calculating entropy and information gain.
DataSet:

Class	Gender	Stay in hostel
9	M	Yes
10	F	No
8	F	Yes
8	F	No
9	M	Yes
10	M	No
11	F	Yes
11	M	Yes
8	F	Yes
9	M	No
11	M	No
11	M	Yes
10	F	No
10	M	Yes

We have two features and we will try to choose the root node by calculating the information gain by splitting each feature.

Let' review the formula for entropy and information gain:

```
E = -p*log2(p) - q*log2(q)
```

$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

Let's start with feature "class" :

Entropy of Whole Dataset

Out of 14 instances, $n(\text{yes}) = 8$, $n(\text{NO}) = 6$

$$E(S) = -p(\text{yes}) \log_2 p(\text{yes}) - p(\text{NO}) \log_2 p(\text{no})$$

$$= -\left[\frac{8}{14} * \log_2 \left(\frac{8}{14}\right) + \frac{6}{14} * \log_2 \left(\frac{6}{14}\right) \right]$$

$$= -\left[0.57 \times (-0.81) + 0.428 \times (-1.22)\right]$$

$$= -\left[0.462 + 0.5228\right]$$

$$= 0.985$$

Class	Stay in hostel	(n) Total values	p(yes)	p(NO)
8	Yes = 2, NO = 1	3	2/3	1/3
9	Yes = 2, NO = 1	3	2/3	1/3
10	Yes = 1, NO = 3	4	1/4	3/4
11	Yes = 3, NO = 1	4	3/4	1/4
		M = 14		

Selection of Root Node

~~E(class = 08)~~

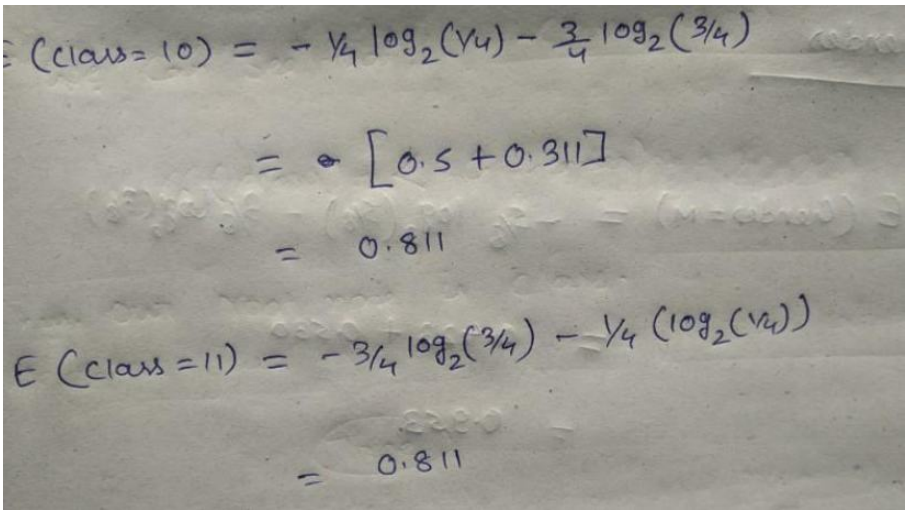
$$E(\text{class} = 8) = -2/3 \log_2(2/3) - 1/3 \log_2(1/3)$$

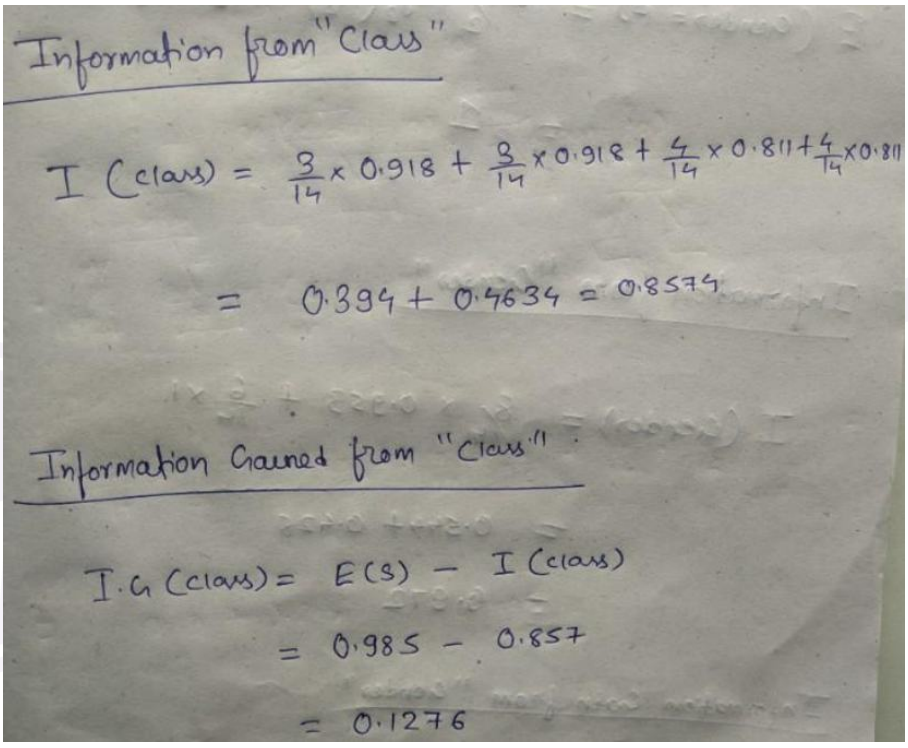
$$= -\left[0.666(-0.586) + 0.333(-1.586)\right]$$

$$= 0.39 + 0.52 = 0.918$$

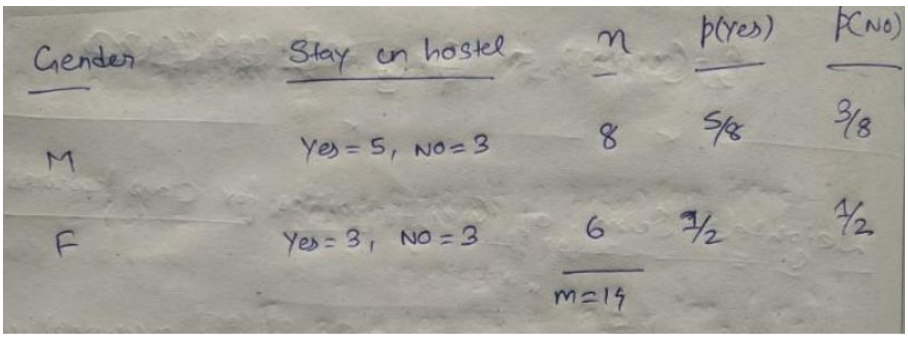
$$E(\text{class} = 9) = -2/3 \log_2(2/3) - 1/3 \log_2(1/3)$$

$$= 0.918$$





Let' see the information gain from feature "gender" :



Gender

$$E(\text{Gender}=M) = -5/8 \log_2(5/8) - 3/8 \log_2(3/8)$$

$$= 0.423 + 0.530$$

$$= 0.953$$

$$E(\text{Gender}=F) = -1/2 \log_2(1/2) - 1/2 \log_2(1/2)$$

$$= 1$$

Information from "Gender"

$$I(\text{Gender}) = 8/14 \times 0.953 + 6/14 \times 1$$

$$= 0.544 + 0.428$$

$$= 0.972$$

Information Gain from "Gender"

$$I.G(\text{Gender}) = E(S) - I(\text{Gender})$$

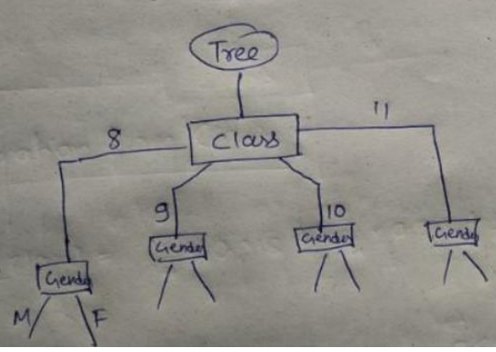
$$= 0.985 - 0.972$$

$$= 0.012$$

We can clearly see,

$$I.G(\text{Gender}) < I.G(\text{class})$$

Since, "Class" provides more information gain, thus our root node will be "class".



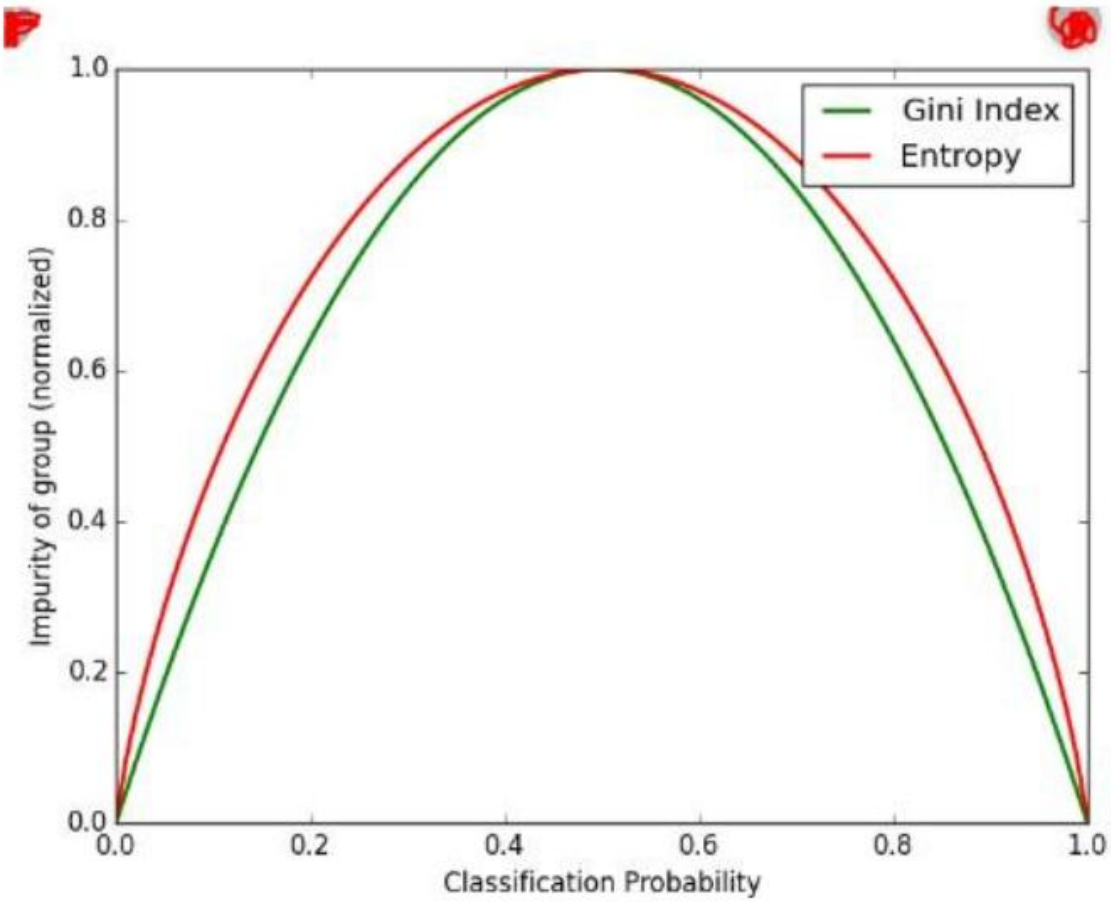
Different Algorithms for Decision Tree

- **ID3 (Iterative Dichotomiser) :** It is one of the algorithms used to construct decision tree for classification. It uses Information gain as the criteria for finding the root nodes and splitting them. It only accepts categorical
- attributes.
- **C4.5 :** It is an extension of ID3 algorithm, and better than ID3 as it deals both continuous and discreet values.It is also used for classfication purposes.
- **Classfication and Regression Algorithm(CART) :** It is the most popular algorithm used for constructing decison trees. It uses ginni impurity as the default calculation for selecting root nodes, however one can use "entropy" for criteria as well. This algorithm works on both regression as well as

classfication problems. We will use this algorithm in our pyhton implementation.

Entropy and Ginni impurity can be used reversibly. It doesn't affects the result much. Although, ginni is easier to compute than entropy, since entropy has a log term calculation. That's why CART algorithm uses ginni as the default algorithm.

If we plot ginni vs entropy graph, we can see there is not much difference between them:



Advantages of Decision Tree:

- It can be used for both Regression and Classification problems.
- Decision Trees are very easy to grasp as the rules of splitting is clearly mentioned.
- Complex decision tree models are very simple when visualized. It can be understood just by visualising.
- Scaling and normalization are not needed.

Disadvantages of Decision Tree:

- A small change in data can cause instability in the model because of the greedy approach.
- Probability of overfitting is very high for Decision Trees.
- It takes more time to train a decision tree model than other classification algorithms.

Cross-Validation

Suppose you train a model on a given dataset using any specific algorithm. You tried to find the accuracy of the trained model using the same training data and found the accuracy to be 95% or maybe even 100%. What does this mean? Is your model ready for prediction? The answer is no.

Why? Because your model has trained itself on the given data, i.e. it knows the data and it has generalized over it very well. But when you try and predict over a new set of data, it's most likely to give you very bad accuracy, because it has never seen the data before and thus it fails to generalizes well over it. This is the problem of overfitting.

To tackle such problem, Cross-validation comes into the picture. Cross- validation is a resampling technique with a basic idea of dividing the training dataset into two parts i.e. train and test. On one part(train) you try to train the model and on the second part(test) i.e. the data which is unseen for the model, you make the prediction and check how well your model works on it. If the model works with good accuracy on your test data, it means that the model has not overfitted the training data and can be trusted with the prediction, whereas if it performs with bad accuracy then our model is not to be trusted and we need to tweak our algorithm.

Let's see the different approaches of Cross-Validation:

Hold Out Method:
It is the most basic of the CV techniques. It simply divides the dataset into two sets of training and test. The training dataset is used to train the model and then test data is fitted in the trained model to make predictions. We check the accuracy and assess our model on that basis. This method is used as it is computationally less costly. But the evaluation based on the Hold-out set can have a high variance because it depends heavily on which data points end up in the training set and which in test data. The evaluation will be different every time this division changes.

k-fold Cross-Validation

img_src:Wikipedia

To tackle the high variance of Hold-out method, the k-fold method is used. The idea is simple, divide the whole dataset into 'k' sets preferably of equal sizes.

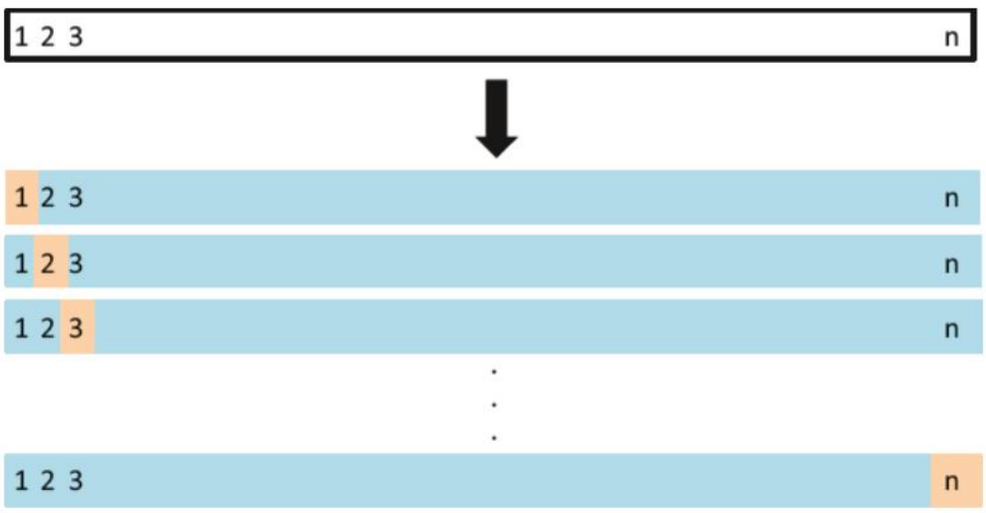
Then the first set is selected as the test set and the rest 'k-1' sets are used to train the data. Error is calculated for this particular dataset.

Then the steps are repeated, i.e. the second set is selected as the test data, and the remaining 'k-1' sets are used as the training data. Again, the error is calculated. Similarly, the process continues for 'k' times. In the end, the CV error is given as the mean of the total errors calculated individually, mathematically given as:

$$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k}\mathrm{MSE}_i.$$

The variance in error decreases with the increase in 'k'. The disadvantage of k- fold cv is that it is computationally expensive as the algorithm runs from scratch for 'k' times.

- Leave One Out Cross Validation (LOOCV)



LOOCV is a special case of k-fold CV, where k becomes equal to n (number of observations). So instead of creating two subsets, it selects a single observation as a test data and rest of data as the training data. The error is calculated for this test observations. Now, the second observation is selected as test data, and the rest of the data is used as the training set. Again, the error is calculated for this particular test observation. This process continues 'n' times and in the end, CV error is calculated as:

$$\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\mathrm{MSE}_i.$$

Bias Variance tradeoff for k-fold CV, LOOCV and Holdout Set CV

There is a very good explanation given in the ISLR Book as given below:

A k-fold CV with k < n has a computational advantage to LOOCV. But putting computational issues aside, a less obvious but potentially more important advantage of k-fold CV is that it often gives more accurate estimates of the test error rate than does LOOCV.

The validation set approach can lead to overestimates of the test error rate since in this approach the training set used to fit the statistical learning method contains only half the observations of the entire data set.

Using this logic, it is not hard to see that LOOCV will give approximately unbiased estimates of the test error since each training set contains n – 1 observations, which is almost as many as the number of observations in the full data set.

And performing k-fold CV for, say, k 5 or k 10 will lead to an intermediate level of bias since each training set contains (k – 1)n/k observations—fewer than in the LOOCV approach, but substantially more than in the validation set approach. Therefore, from the perspective of bias reduction, it is clear that LOOCV is to be preferred to k-fold CV.

However, we know that bias is not the only source for concern in an estimating procedure; we must also consider the procedure's variance. It turns out that LOOCV has higher variance than does k- fold CV with k < n. Why is this the case? When we perform LOOCV, we are in effect averaging the outputs of n fitted models, each of which is trained on an almost identical set of observations; therefore, these outputs are highly (positively) correlated with each other.

In contrast, when we perform k-fold CV with k < n, we are averaging the outputs of k fitted models that are somewhat less correlated with each other since the overlap between the training sets in each model is smaller. Since the mean of many highly correlated quantities has higher variance than does the mean of many quantities that are not as highly correlated, the test error estimate resulting from LOOCV tends to have higher variance than does the test error estimate resulting from k-fold CV.