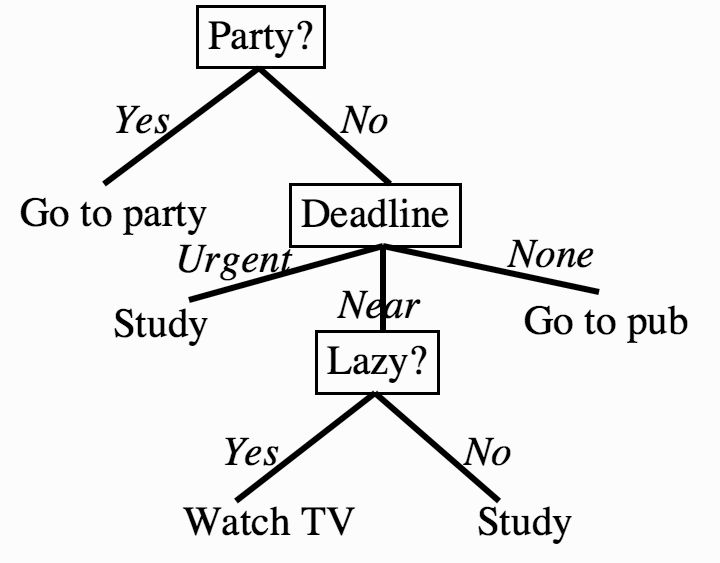
**Q 16. Explain the concept of classification using Decision Tree.**

* Classification by decision trees has grown in popularity over recent years.
* The idea of a decision tree is that we break classification down into a set of choices about each feature in turn, starting at the root (base) of the tree and progressing down to the leaves, where we receive the classification decision.
* Trees can even be turned into a set of if-then rules, suitable for use in a rule induction system.
* In terms of optimisation and search, decision trees use a greedy heuristic to perform search, evaluating the possible options at the current stage of learning and making the one that seems optimal at that point.

For example – “Decision regarding how to spend a day”, may be dependent on various factors like whether there is any deadline of predefined schedule, whether the is feeling lazy, want to party. Following diagram shows the decision tree built on all there possible alternatives of “Spending a day”.



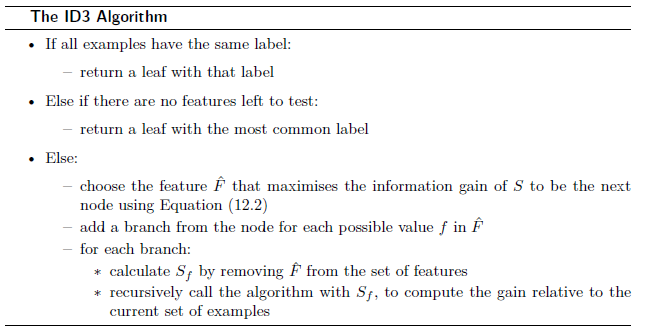
One of the reasons that decision trees are popular is that we can turn them into a set of logical disjunctions (if ... then rules).

then go into program code very simply—the first part of the tree above can be turned into:

• if there is a party then go to it

• if there is not a party and you have an urgent deadline then study…. etc.

**Q17. Write and explain the ID3 – decision tree construction algorithm.**



The ID3 algorithm begins with the original set (S) as the root node

. On each iteration of the algorithm, it iterates through every

unused attribute of the set (S) and calculates the entropy H(S)

or the information gain IG(S) of that attribute. It then selects the attribute

which has the smallest entropy (or largest information gain) value.

The set (S) is then split or partitioned by the selected attribute to

produce subsets of the data. (For example, a node can be split into child nodes based upon the subsets of the population whose ages are less than 50, between 50 and 100, and greater than 100.) The algorithm continues to recurse on each subset, considering only attributes never selected before.

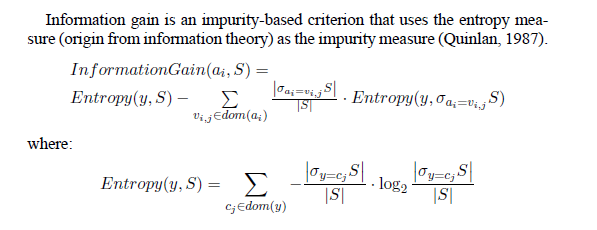
Recursion on a subset may stop in one of these cases:

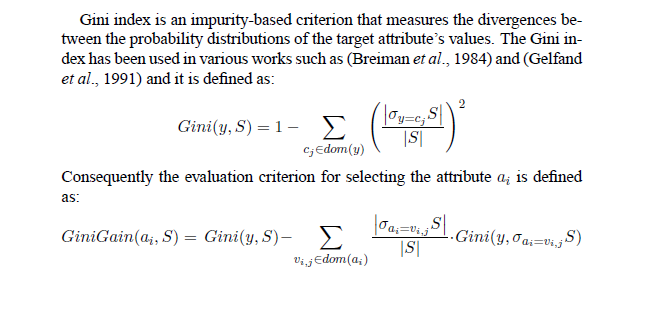
* every element in the subset belongs to the same class; in which case the node is turned into a leaf node and labelled with the class of the examples.
* there are no more attributes to be selected, but the examples still do not belong to the same class. In this case, the node is made a leaf node and labelled with the most common class of the examples in the subset.
* there are no examples in the subset, which happens when no example in the parent set was found to match a specific value of the selected attribute. An example could be the absence of a person among the population with age over 100 years. Then a leaf node is created and labelled with the most common class of the examples in the parent node's set.

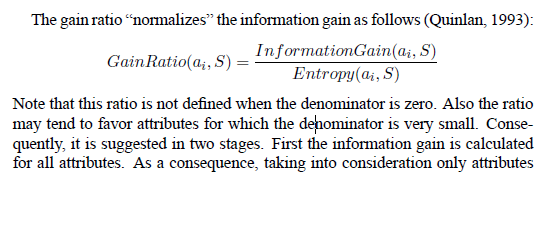
Throughout the algorithm, the decision tree is constructed with each non-terminal node (internal node) representing the selected attribute on which the data was split, and terminal nodes (leaf nodes) representing the class label of the final subset of this branch.

**Q18. Explain the terms- Information Gain, entropy, Gini Index and Information Gain Ratio?**

**Or What is Information Gain? What are different measures**

****

****

****

**Q19. What are different strategies applied to overcome the problem of overfitting in decision tree?**

**Sol:-** Overfitting is a common problem in decision tree models, where the model becomes too complex and starts fitting the training data too closely, resulting in poor generalization to unseen data. To overcome overfitting in decision trees, several strategies can be applied:

1. Pruning: Pruning is a technique used to reduce the complexity of a decision tree by removing unnecessary branches. There are two main types of pruning:

- Pre-pruning: In pre-pruning, the tree is pruned during the construction phase by setting termination conditions. These conditions could include stopping the tree growth after a certain depth, limiting the number of samples in leaf nodes, or requiring a minimum improvement in impurity measures before further splitting.

- Post-pruning: Post-pruning involves growing the decision tree to its maximum size and then pruning it back by removing branches that do not significantly improve the overall performance. This can be done using techniques like reduced-error pruning or cost-complexity pruning.

2. Limiting tree depth: By setting a maximum depth for the decision tree, you can control its complexity and prevent it from becoming too deep. This can help avoid overfitting by limiting the model's ability to memorize noise in the training data.

3. Increasing minimum samples per leaf: Setting a higher threshold for the minimum number of samples required to create a leaf node can help control overfitting. By requiring a minimum number of samples, you can prevent the tree from creating nodes with very few instances, which might be prone to overfitting.

4. Increasing minimum samples for splitting: Similar to the previous strategy, you can increase the minimum number of samples required for a node to be split further. This can prevent the tree from creating overly specific splits that capture noise in the data.

5. Feature selection: Careful selection of relevant features can help reduce overfitting. Removing irrelevant or redundant features can simplify the decision tree and improve its generalization. Various feature selection techniques, such as information gain, gain ratio, or feature importance measures, can be used to determine the most informative features.

6. Ensemble methods: Instead of using a single decision tree, ensemble methods like random forests or gradient boosting can be employed. These methods combine multiple decision trees to make predictions, which can help reduce overfitting and improve generalization.

7. Cross-validation: Using techniques like k-fold cross-validation during model training and evaluation can provide a more robust estimate of the model's performance and help identify potential overfitting issues.

By applying these strategies, you can mitigate the problem of overfitting in decision tree models and improve their ability to generalize to unseen data.

**Q18. Write and explain with examples a Naive Bayes Classifier**

**Sol:** A Naive Bayes classifier is a probabilistic classifier that applies Bayes' theorem with the assumption of independence between features. It is called "naive" because it simplifies the

modeling process by assuming that the presence or absence of a particular feature is unrelated to the presence or absence of other features.

Here's an explanation of how a Naive Bayes classifier works, along with an example:

Step 1: Training Phase

1. Data Collection: Gather a labeled dataset containing examples of inputs (features) and their corresponding class labels.

2. Feature Extraction: Preprocess the data and extract relevant features from each example. The features can be binary (e.g., presence/absence), categorical, or continuous.

3. Compute Class Prior Probabilities: Calculate the prior probabilities of each class based on the frequency of their occurrence in the training dataset. For example, if there are 100 instances in the training data, and 30 belong to Class A and 70 belong to Class B, the prior probabilities would be P(Class A) = 0.3 and P(Class B) = 0.7.

4. Compute Conditional Feature Probabilities: For each feature and each class, compute the conditional probability of observing a specific feature value given the class. This step involves estimating the likelihood probabilities based on the training data. The specific method used to estimate these probabilities depends on the nature of the features:

- For binary features, such as "word present" or "word absent," the conditional probabilities can be calculated as the fraction of instances in each class where the feature is present or absent.

- For categorical features, such as "color" with categories "red," "blue," and "green," the conditional probabilities can be estimated by calculating the fraction of instances in each class with a specific category.

- For continuous features, such as "age" or "height," the conditional probabilities can be estimated using probability density functions (PDFs), such as Gaussian distributions.

Step 2: Classification Phase

Given a new instance with input features, the Naive Bayes classifier predicts its class label by computing the posterior probabilities for each class and selecting the class with the highest probability.

1. Compute Posterior Probabilities: For each class, calculate the posterior probability by multiplying the prior probability of the class with the conditional probabilities of the observed features given that class. Apply Bayes' theorem to calculate the posterior probabilities:

P(Class | features) = (P(Class) \* P(features | Class)) / P(features)

- `P(Class | features)`: Posterior probability of the class given the observed features.

- `P(Class)`: Prior probability of the class.

- `P(features | Class)`: Conditional probability of the observed features given the class.

- `P(features)`: Probability of observing the features.

2. Select the Most Probable Class: Compare the posterior probabilities for each class and select the class with the highest probability as the predicted class for the given instance.

**Example: Email Classification**

Suppose we want to classify emails as either "spam" or "non-spam" based on two binary features: "contains the word 'offer'" and "contains the word 'lottery'." We have a training dataset with 100 emails, of which 40 are spam and 60 are non-spam.

Training Phase:

- Class prior probabilities: P(spam) = 0.4, P(non-spam) = 0.6

- Conditional feature probabilities:

- P(contains 'offer' | spam) = 0.9 (9 out of 10 spam emails contain 'offer')

- P(contains 'lottery' | spam) = 0.3 (3 out of 10 spam emails contain 'lottery')

- P(contains 'offer' | non-spam) = 0.2 (12 out of 60 non-spam emails contain 'offer')

- P(contains 'lottery' | non-spam) = 0.05 (3 out of 60 non-spam emails contain 'lottery')

Classification Phase:

Suppose we have a new email with the following features: contains 'offer' = True, contains 'lottery' = False.

1. Compute Posterior Probabilities:

- P(spam | features) = (0.4 \* 0.9 \* (1 - 0.3)) / P(features)

- P(non-spam | features) = (0.6 \* 0.2 \* (1 - 0.05)) / P(features)

2. Normalize the Probabilities: Calculate P(features) as the sum of the individual probabilities of each class multiplied by their respective conditional probabilities:

- P(features) = (0.4 \* 0.9 \* (1 - 0.3)) + (0.6 \* 0.2 \* (1 - 0.05))

3. Select the Most Probable Class: Compare the posterior probabilities and choose the class with the highest probability. In this case, if P(spam | features) > P(non-spam | features), the email is classified as "spam"; otherwise, it is classified as "non-spam."

Naive Bayes classifiers are computationally efficient, easy to implement, and can perform well even with relatively small training datasets. However, the assumption of feature independence may not hold in all scenarios, and it can limit the classifier's accuracy.

**Q 19. Explain k-NEAREST NEIGHBOR LEARNING method for classification?**

The most basic instance-based method is the k-NEAREST NEIGHBOR algorithm. This algorithm assumes all instances correspond to points in the n-dimensional space Kn. The nearest neighbors of an instance are defined in terms of the standard. k Nearest Neighbors(kNN) is a non-parametric method used for classification . It is a lazy learning algorithm where all computation is deferred until classification. It is also an instance based learning algorithm: Store training examples and delay the processing (“lazy evaluation”) until a new instance must be classified.

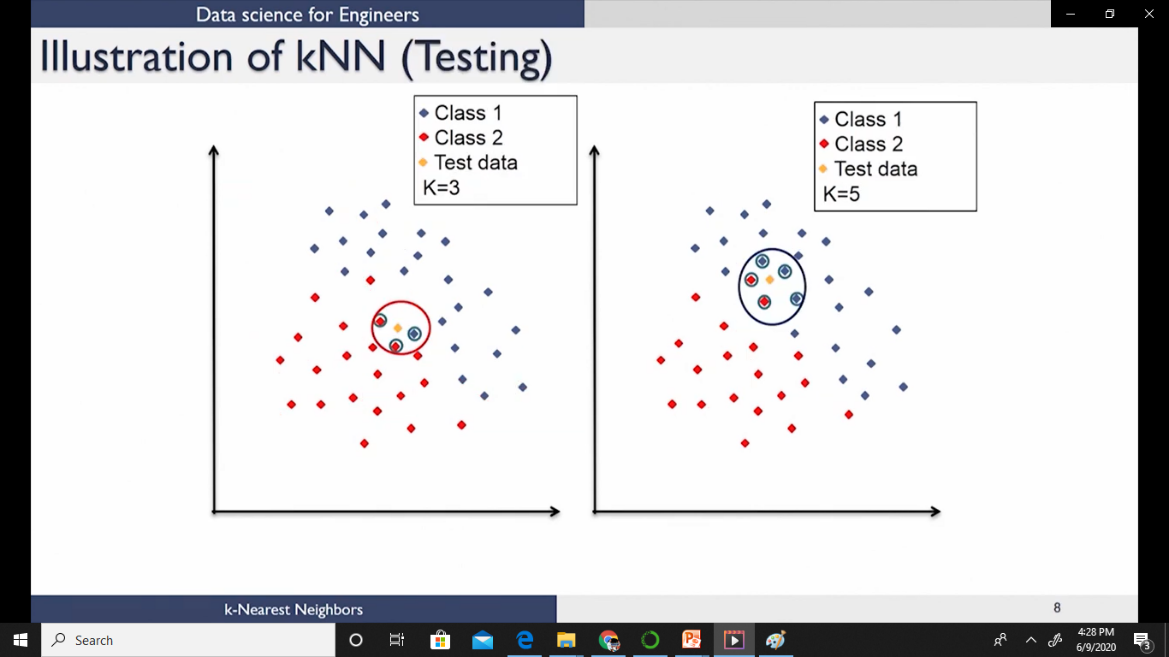
**In K Nearest Neighbors, Input features can be both quantitative and qualitative. The Outputs are categorical values, which typically are the class of the data. It explains a categorical value using the majority votes of nearest neighbors.**

**Algorithm-**

The KNN classification is performed using the following four steps

* Compute the distance metric between the test data point and all the labeled data points
* Order the labeled data points in the increasing order of this distance metric
* Select the top k labeled data points and look at the class labels
* Find the class label that the majority of these k labeled data points have and assign it to

the test data point.



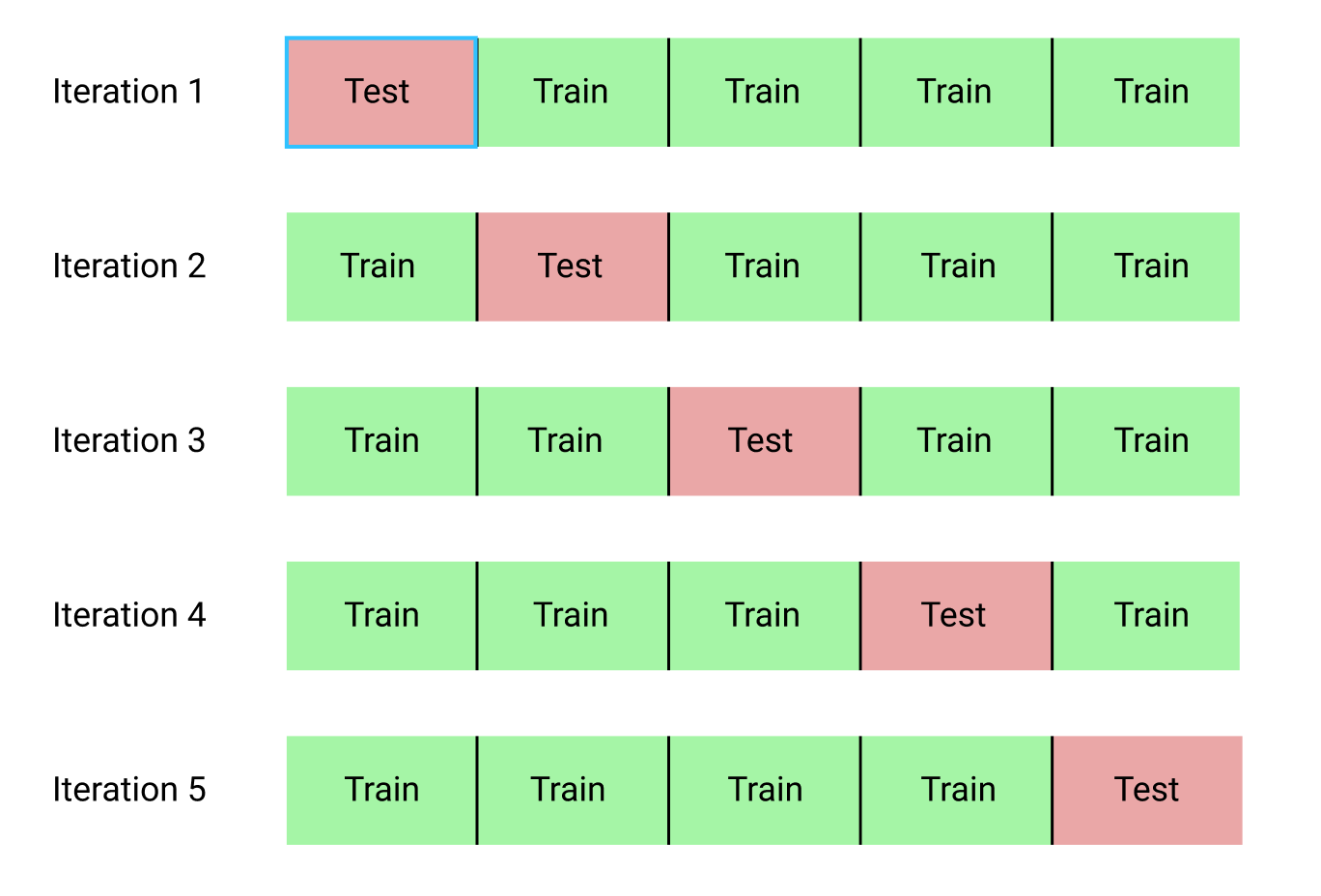
The best choice of k depends on the data

Larger values of k reduce the effect of noise on classification but makes the decision boundaries between classes less distinct.

Smaller values of k tend to be affected by the noise with clear separation between classes

**Q20. Explain the k-fold cross validation ?**

When a system that has only a limited sample of data Do , is available, several methods can be used to resample Do. Following figure describes a k-fold method in which Do is partitioned into k disjoint, equal-sized subsets. In this k-fold approach, each example from Do is used exactly once in a test set, and k - 1 times in a training set. A second popular approach is to randomly choose a test set of at least 30 examples from Do, use the remaining examples for training, then repeat this process as many times as desired. This randomized method has the advantage that it can be repeated an indefinite number of times, to shrink the confidence interval to the desired width. In contrast, the k-fold method is limited by the total number of examples, by the use of each example only once in a test set, and by our desire to use samples of size at least 30. However, the randomized method has the disadvantage that the test sets no longer qualify as being independently drawn with respect to the underlying instance distribution D. In contrast, the test sets generated by k-fold cross validation are independent because each instance is included in only one test set.



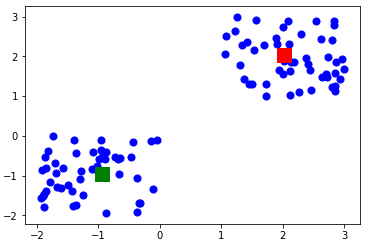
Q.21 Explain K-means clustering Algorithm?

k-Means Clustering-

Let us say we have an image that is stored with 24 bits/pixel and can have up to 16 million colors. Assume we have a color screen with 8 bits/pixel that can display only 256 colors. We want to find the best 256 colors among all 16 million colors such that the image using only the 256 colors in the palette looks as close as possible to the original image. This is color quantization where we map from high to lower resolution. In the general case, the aim is to map from a continuous space to a discrete space; this process is called vector quantization. Of course we can always quantize uniformly, but this wastes the color-map by assigning entries to colors not existing in the image, or would not assign extra entries to colors frequently used in the image. For example, if the image is a seascape, we expect to see many shades of blue and maybe no red. So the distribution of the color-map entries should reflect the original density as close as possible placing many entries in high-density regions, discarding regions where there is no data.

Let us say we have a sample of X= {xt}n t = 1 reference vectors . We have k reference vectors, mj , j = 1,...,k. In our example of color quantization, xt are the image pixel values in 24 bits and mj are the color map entries also in 24 bits, with k = 256. Assume for now that we somehow have the mj ,values; we discuss how to learn them shortly. Then in displaying the image, given the pixel, xt, we represent it with the most similar entry, mi in the color map,

satisfying | xt − mj | = Min j  | xt − mj |



How K-means work-

* To process the learning data, the K-means algorithm in data mining starts with a first group of randomly selected centroids, which are used as the beginning points for every cluster, and then performs iterative (repetitive) calculations to optimize the positions of the centroids.
* It halts creating and optimizing clusters when either:
* The centroids have stabilized — there is no change in their values because the clustering has been successful.
* The defined number of iterations has been achieved.