1. Recognize the differences between supervised, semi-supervised, and unsupervised learning.

Answer: Supervised learning aims to learn a function that, given a sample of data and desired outputs, approximates a function that maps inputs to outputs.

Semi-supervised learning aims to label unlabeled data points using knowledge learned from a small number of labeled data points.

Unsupervised learning does not have (or need) any labeled outputs, so its goal is to infer the natural structure present within a set of data points.

2. Describe in detail any five examples of classification problems.

Answer:

Here is the list of real-life examples of machine learning classification problems:

1. Customer behaviour prediction: Customers can be classified in different categories based on their buying patterns, web store browsing patterns etc.
2. Web text prediction: Classifies web text or assign tag to web text based on pre-determined categories learned from the past data
3. Ad click through rate prediction – Binary classification – Whether one or more ads on the website will be clicked or not.
4. Product categorization – Multinomial classification – Categorize the products sold by different retailers in same categories irrespective of categories assigned to the product by the respective retailers. This use case is relevant for ecommerce aggregators.
5. Malware classification – Multinomial classification – Classify the new / emerging malwares on the basis of comparable features of similar malwares
6. Customer churn prediction – Binary classification – Whether a customer will churn or not in near future
7. Deduction validation classification – Binary classification – Whether a deduction claimed by the buyer on a given invoice is valid or invalid deduction
8. Blocked order release recommendation – Binary classification – Classifies whether an order placed by customer should be blocked or not based on the buyer credit exposure
9. Document classification – Multinomial classification – Classifies documents in different categories

3. Describe each phase of the classification process in detail.

Answer: The classification process consists of the following steps:

1. Data Acquisition and Segmentation: e.g., the biomedical data is recorded from the human body and then pre-processed.
2. Data Pre-processing – Data pre-processing is a technique that is utilized to transform the raw data into a useful and effective format. The data may contain noise—many irrelevant and missing parts—which should be eliminated. Atmospheric correction, noise removal, image transformation, main component analysis, etc.;
3. Feature Extraction: Detection and extraction of an object, including detection of position and other characteristics of a moving object image obtained by a camera; while in extraction, estimating the trajectory of the detected object in the image plane;
4. Dimension Reduction: Dimension reduction is employed to eliminate the redundant information from the feature vector, generating a reduced feature vector. The appropriate structure in the raw data is described by the feature vector.
5. Training – selection of the particular attribute which best describes the pattern;
6. Data Classification– this step categorizes detected objects into predefined classes by using a suitable method that compares the image patterns with the target patterns. a classifier classifies the reduced feature vector.
7. Accuracy evaluation: Accuracy assessment of the model.

4. Go through the SVM model in depth using various scenarios.

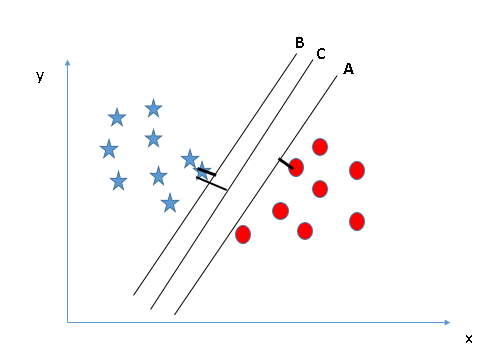
Answer: Support Vector Machine” (SVM) is a supervised machine learning algorithm that can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate. Support Vectors are simply the coordinates of individual observation. The SVM classifier is a frontier that best segregates the two classes (hyper-plane/ line).



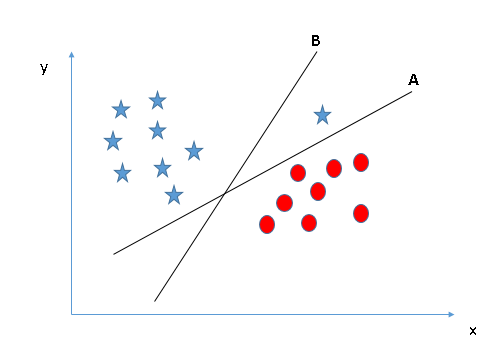
Scenario-1: we have three hyper-planes (a, b, and c). now, identify the right hyper-plane to classify stars and circles. “Select the hyper-plane which segregates the two classes better”. In this scenario, hyper-plane “B” has excellently performed this job.



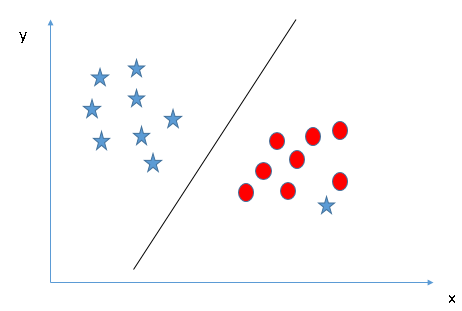
Scenario-2: We have three hyper-planes (a, b, and c) and all are segregating the classes well. now, how can we identify the right hyper-plane? Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as Margin. Below, we can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chance of miss-classification.



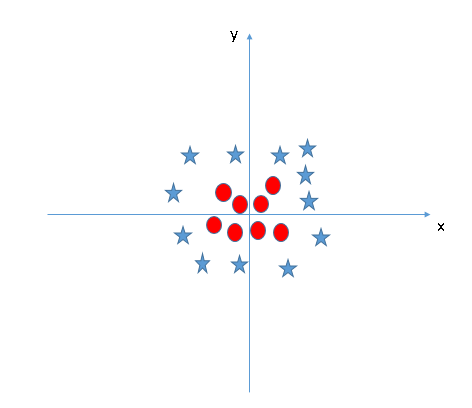
Scenario-3: SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. here, hyper-plane B has a classification error and a has classified all correctly. Therefore, the right hyper-plane is A.



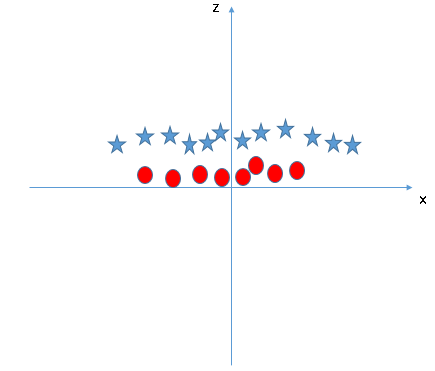
Scenario-4: Below, I am unable to segregate the two classes using a straight line, as one of the stars lies in the territory of other(circle) class as an outlier. one star at other end is like an outlier for star class. the SVM algorithm has a feature to ignore outliers and find the hyper-plane that has the maximum margin. hence, we can say, SVM classification is robust to outliers.



Scenario-5: We can’t have linear hyper-plane between the two classes, so how does SVM classify these two classes. Till now, we have only looked at the linear hyper-plane.



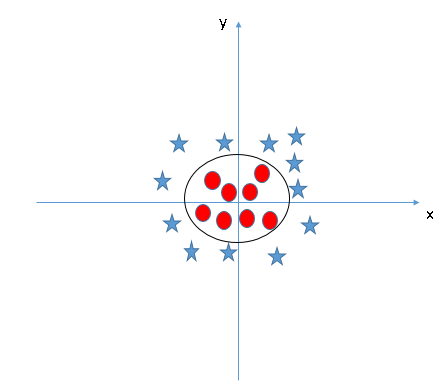
SVM can solve this problem. Easily! It solves this problem by introducing additional feature. Here, we will add a new feature z=x^2+y^2. Now, let’s plot the data points on axis x and z:



In above plot, points to consider are: All values for z would be positive always because z is the squared sum of both x and y. In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z.

In the SVM classifier, it is easy to have a linear hyper-plane between these two classes. But another burning question which arises is, should we need to add this feature manually to have a hyper-plane. No, the SVM algorithm has a technique called the kernel trick. The SVM kernel is a function that takes low dimensional input space and transforms it to a higher dimensional space i.e. it converts not separable problem to separable problem. It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then finds out the process to separate the data based on the labels or outputs you’ve defined.

When we look at the hyper-plane in original input space it looks like a circle:



5. What are some of the benefits and drawbacks of SVM?

Answer: Benefits and Drawbacks of SVM pros:

Benefits:

1. It works really well with a clear margin of separation
2. It is effective in high dimensional spaces.
3. It is effective in cases where the number of dimensions is greater than the number of samples.
4. It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

Drawbacks:

1. It doesn’t perform well when we have large data set because the required training time is higher
2. It also doesn’t perform very well, when the data set has more noise i.e. target classes are overlapping
3. SVM doesn’t directly provide probability estimates, these are calculated using an expensive five-fold cross-validation. it is included in the related svc method of python scikit-learn library.

6. Go over the KNN model in depth.

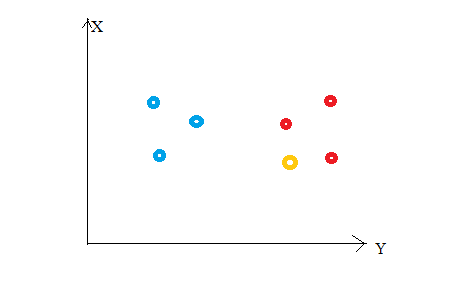
Answer:

K Nearest Neighbour algorithm falls under the Supervised Learning category and is used for classification (most commonly) and regression. It is a versatile algorithm also used for imputing missing values and resampling datasets. As the name (K Nearest Neighbour) suggests it considers K Nearest Neighbours (Data points) to predict the class or continuous value for the new Datapoint.

The algorithm’s learning is:

* Instance-based learning: Here we do not learn weights from training data to predict output (as in model-based algorithms) but use entire training instances to predict output for unseen data.
* Lazy Learning: Model is not learned using training data prior and the learning process is postponed to a time when prediction is requested on the new instance.
* Non-Parametric: In KNN, there is no predefined form of the mapping function.

Consider the following figure. Let us say we have plotted data points from our training set on a two-dimensional feature space. As shown, we have a total of 6 data points (3 red and 3 blue). Red data points belong to ‘class1’ and blue data points belong to ‘class2’. And yellow data point in a feature space represents the new point for which a class is to be predicted. Obviously, we say it belongs to ‘class1’ (red points). Because its nearest neighbours belong to that class.



This is the principle behind K Nearest Neighbours. Here, nearest neighbours are those data points that have minimum distance in feature space from our new data point. And K is the number of such data points we consider in our implementation of the algorithm. Therefore, distance metric and K value are two important considerations while using the KNN algorithm. Euclidean distance is the most popular distance metric. We can also use Hamming distance, Manhattan distance, Minkowski distance as per your need. For predicting class/ continuous value for a new data point, it considers all the data points in the training dataset. Finds new data point’s ‘K’ Nearest Neighbours (Data points) from feature space and their class labels or continuous values.

Then:

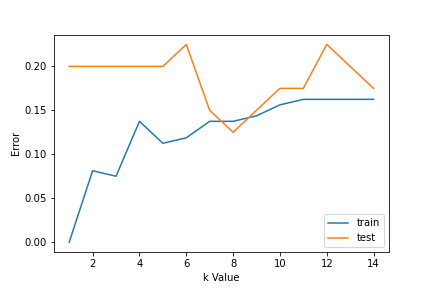
For classification: A class label assigned to the majority of K Nearest Neighbors from the training dataset is considered as a predicted class for the new data point.

For regression: Mean or median of continuous values assigned to K Nearest Neighbours from training dataset is a predicted continuous value for our new data point.

Here, we do not learn weights and store them, instead, the entire training dataset is stored in the memory. Therefore, model representation for KNN is the entire training dataset.

K is a crucial parameter in the KNN algorithm. Some suggestions for choosing K Value are:

* Using error curves: The figure below shows error curves for different values of K for training and test data.



At low K values, there is overfitting of data/high variance. Therefore, test error is high and train error is low. At K=1 in train data, the error is always zero, because the nearest neighbour to that point is that point itself. Therefore, though training error is low test error is high at lower K values. This is called overfitting. As we increase the value for K, the test error is reduced.

But after a certain K value, bias/ underfitting is introduced and test error goes high. So, we can say initially test data error is high (due to variance) then it goes low and stabilizes and with further increase in K value, it again increases (due to bias). The K value when test error stabilizes and is low is considered as optimal value for K. From the above error curve, we can choose K=8 for our KNN algorithm implementation.

* Also, domain knowledge is very useful in choosing the K value.
* K value should be odd while considering binary(two-class) classification.

Data Preparation:

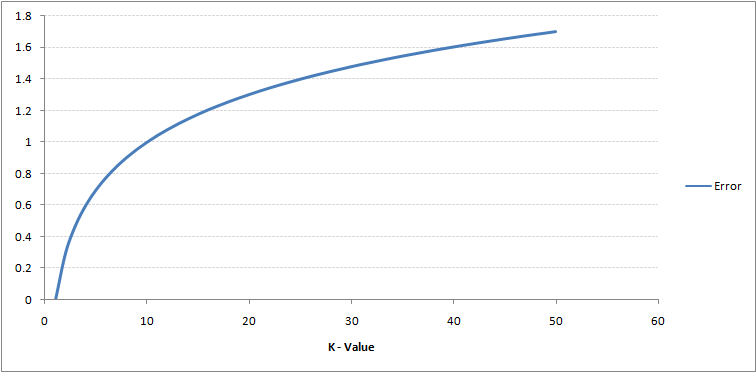
* Data Scaling: To locate the data point in multidimensional feature space, it would be helpful if all features are on the same scale. Hence normalization or standardization of data will help.
* Dimensionality Reduction: KNN may not work well if there are too many features. Hence dimensionality reduction techniques like feature selection, principal component analysis can be implemented.
* Missing value treatment: If out of M features one feature data is missing for a particular example in the training set, then we cannot locate or calculate distance from that point. Therefore, deleting that row or imputation is required.

K Nearest Neighbour (classification) steps:

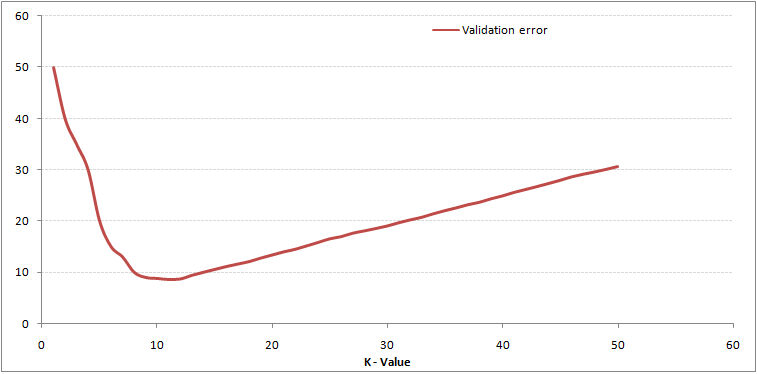
* Load the data
* Initialise the value of k
* For getting the predicted class, iterate from 1 to total number of training data points
* Calculate the distance between test data and each row of training data. Here we will use Euclidean distance as our distance metric since it’s the most popular method. The other metrics that can be used are Chebyshev, cosine, etc.
* Sort the calculated distances in ascending order based on distance values
* Get top k rows from the sorted array
* Get the most frequent class of these rows
* Return the predicted class

7. Discuss the KNN algorithm's error rate and validation error.

Answer: The training error rate and the validation error rate are two parameters we need to access different K-value. Following is the curve for the training error rate with a varying value of K:



As you can see, the error rate at K=1 is always zero for the training sample. This is because the closest point to any training data point is itself. Hence, the prediction is always accurate with K=1. If validation error curve would have been similar, our choice of K would have been 1. Following is the validation error curve with varying value of K:



At K=1, we were overfitting the boundaries. Hence, error rate initially decreases and reaches a minima. After the minima point, it then increase with increasing K. To get the optimal value of K, you can segregate the training and validation from the initial dataset. Now plot the validation error curve to get the optimal value of K. This value of K should be used for all predictions.

8. For kNN, talk about how to measure the difference between the test and training results.

Answer: Distance-based classification algorithms are techniques used for classifying data objects by computing the distance between the test sample and all training samples using a distance function.

The traditional k-NN classification algorithm finds the k-nearest neighbour(s) and classifies numerical data records by calculating the distance between the test sample and all training samples using the Euclidian distance.

In k-NN, the k value represents the number of nearest neighbours. This value is the core deciding factor for this classifier due to the k-value deciding how many neighbours influence the classification. When k=1 then the new data object is simply assigned to the class of its nearest neighbour. The neighbours are taken from a set of training data objects for where the correct classification is already known. k-NN works naturally with numerical data. Various numerical measures have been used such as Euclidean, Manhattan, Minkowsky, City-block, and Chebyshev distances. Amongst these, the Euclidean is the most widely used distance function with k-NN.

9. Create the kNN algorithm.

Answer:

The main steps of k-NN algorithm are

1. Determine the number of nearest neighbours (K values).
2. Compute the distance between test sample and all the training samples.
3. Sort the distance and determine nearest neighbours based on the K-th minimum distance.
4. Assemble the categories of the nearest neighbours.
5. Utilise simple majority of the category of nearest neighbours as the prediction value of the new data object.

10. What is a decision tree, exactly? What are the various kinds of nodes? Explain all in depth.

Answer:

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

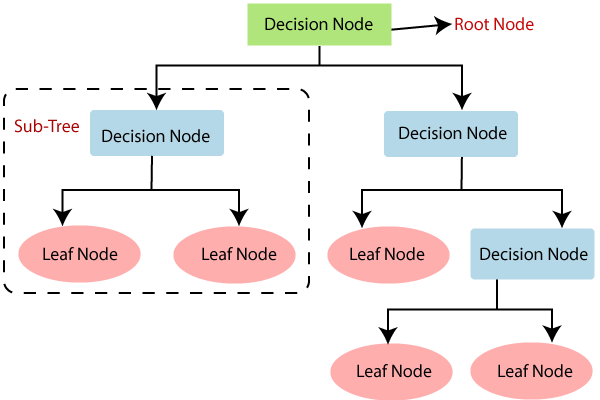
The decisions or the test are performed on the basis of features of the given dataset.

It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.

A decision tree simply asks a question and based on the answer (Yes/No), it further split the tree into subtrees.



There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.

The logic behind the decision tree can be easily understood because it shows a tree-like structure.

Decision Tree Terminologies:

* Root Node: Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
* Leaf Node: Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
* Splitting: Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
* Branch/Sub Tree: A tree formed by splitting the tree.
* Pruning: Pruning is the process of removing the unwanted branches from the tree.
* Parent/Child node: The root node of the tree is called the parent node, and other nodes are called the child nodes.

11. Describe the different ways to scan a decision tree.

Answer:

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as Attribute selection measure or ASM. By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

1. Information Gain
2. Gini Index

1. Information Gain:

Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute. It calculates how much information a feature provides us about a class. According to the value of information gain, we split the node and build the decision tree. A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

Information Gain= Entropy(S)- [(Weighted Avg) \*Entropy (each feature)

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

Where,

S= Total number of samples

P(yes)= probability of yes

P(no)= probability of no

2. Gini Index:

Gini index is a measure of impurity or purity used while creating a decision tree in the CART (Classification and Regression Tree) algorithm. An attribute with the low Gini index should be preferred as compared to the high Gini index. It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits. Gini index can be calculated using the below formula:

Gini Index= 1- ∑jPj2

Pruning: Getting an Optimal Decision tree

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree. A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree pruning technology used:

* Cost Complexity Pruning
* Reduced Error Pruning.

12. Describe in depth the decision tree algorithm.

Answer: In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree.

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).

Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

13. In a decision tree, what is inductive bias? What would you do to stop overfitting?

Answer:

The inductive bias (also known as learning bias) of a learning algorithm is the set of assumptions that the learner uses to predict outputs of given inputs that it has not encountered.

Avoiding overfitting:

* Stop growing before it reaches the point where it perfectly classifies the training data.
* Grow full tree, then post-prune

14.Explain advantages and disadvantages of using a decision tree?

Answer:

Advantages of the Decision Tree

* It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
* It can be very useful for solving decision-related problems.
* It helps to think about all the possible outcomes for a problem.
* There is less requirement of data cleaning compared to other algorithms.

Disadvantages of the Decision Tree

* The decision tree contains lots of layers, which makes it complex.
* It may have an overfitting issue, which can be resolved using the Random Forest algorithm.
* For more class labels, the computational complexity of the decision tree may increase.

15. Describe in depth the problems that are suitable for decision tree learning.

Answer:

Decision tree learning is generally best suited to problems with the following characteristics:

1. Instances are represented by attribute-value pairs.
   1. There is a finite list of attributes (e.g. hair colour) and each instance stores a value for that attribute (e.g. blonde).
   2. When each attribute has a small number of distinct values (e.g. blonde, brown, red) it is easier for the decision tree to reach a useful solution.
   3. The algorithm can be extended to handle real-valued attributes (e.g. a floating point temperature).
2. The target function has discrete output values.
   1. A decision tree classifies each example as one of the output values.
      1. Simplest case exists when there are only two possible classes (Boolean classification).
      2. However, it is easy to extend the decision tree to produce a target function with more than two possible output values.
   2. Although it is less common, the algorithm can also be extended to produce a target function with real-valued outputs.
3. Disjunctive descriptions may be required.
   1. Decision trees naturally represent disjunctive expressions.
4. The training data may contain errors.
   1. Errors in the classification of examples, or in the attribute values describing those examples are handled well by decision trees, making them a robust learning method.
5. The training data may contain missing attribute values.
   1. Decision tree methods can be used even when some training examples have unknown values (e.g., humidity is known for only a fraction of the examples).

After a decision tree learns classification rules, it can also be re-represented as a set of if-then rules in order to improve readability.

16. Describe in depth the random forest model. What distinguishes a random forest?

Answer:

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees.

The fundamental difference is that in Random forests, only a subset of features are selected at random out of the total and the best split feature from the subset is used to split each node in a tree, unlike in bagging where all features are considered for splitting a node.

17. In a random forest, talk about OOB error and variable value.

Answer: OOB\_Score is a very powerful Validation Technique used especially for the Random Forest algorithm for least Variance results. While using the cross-validation technique, every validation set has already been seen or used in training by a few decision trees and hence there is a leakage of data, therefore more variance. But, OOB\_Score prevents leakage and gives a better model with low variance, so we use OOB\_score for validating the model. The OOB\_score is computed as the number of correctly predicted rows from the out-of-bag sample. OOB Error is the number of wrongly classifying the OOB Sample.

Random forest can be a very powerful technique for predicting better values if we use the OOB\_score technique. even if OOB\_score takes a bit more time, but the predictions are worth the time consumed in training the random forest model with the OOB\_score parameter set as true.