Machine Learning

Q1. What is Machine Learning?

Machine learning is the study of computer algorithms that improve automatically through experience. It is seen as a subset of artificial intelligence. Machine Learning explores the study and construction of algorithms that can learn from and make predictions on data. You select a model to train and then manually perform feature extraction. Used to devise complex models and algorithms that lend themselves to a prediction which in commercial use is known as predictive analytics.

Q2. What is Supervised Learning?

Supervised learning is the machine learning task of inferring a function from labeled training data. The training data consist of a set of training examples.

Algorithms: Support Vector Machines, Regression, Naive Bayes, Decision Trees, K-nearest Neighbor Algorithm and Neural Networks

E.g. If you built a fruit classifier, the labels will be "this is an orange, this is an apple and this is a banana", based on showing the classifier examples of apples, oranges and bananas.

Q3. What is Unsupervised learning?

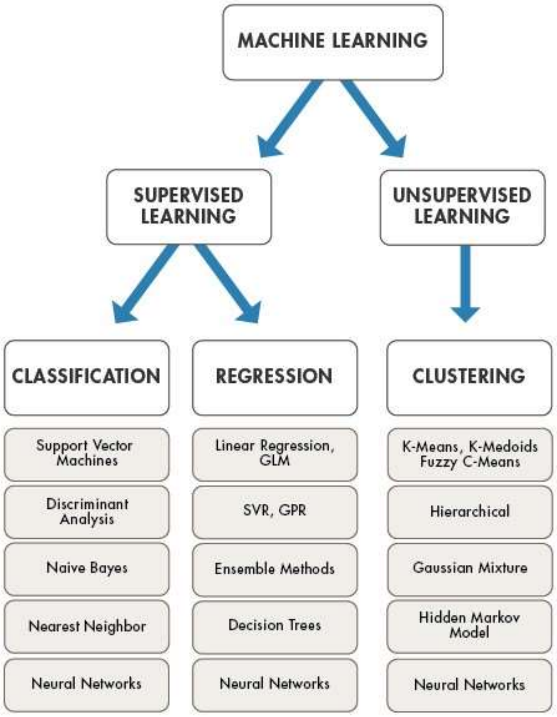
Unsupervised learning is a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labelled responses.

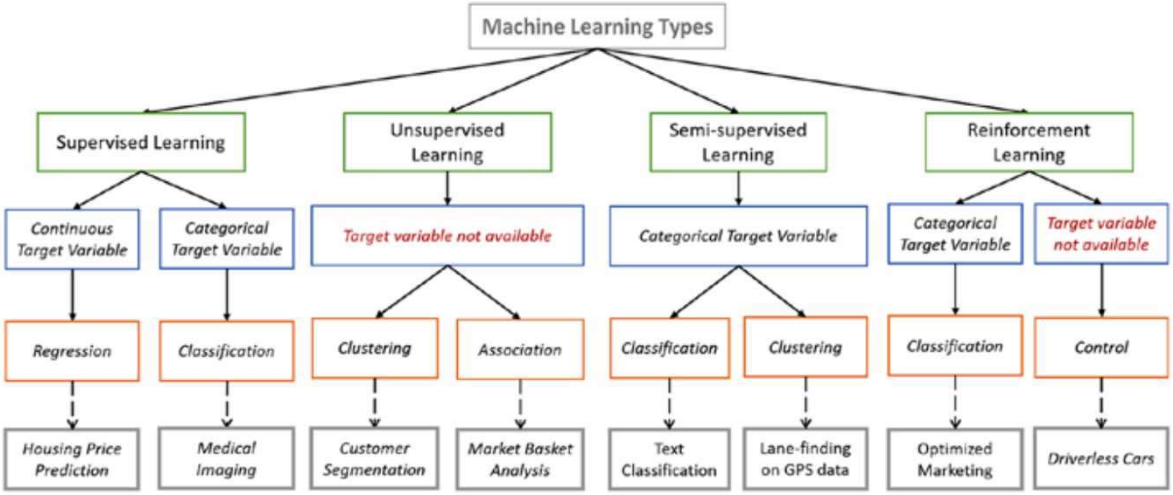
Algorithms: Clustering, Anomaly Detection, Neural Networks and Latent Variable Models

E.g. In the same example, a fruit clustering will categorize as "fruits with soft skin and lots of dimples", "fruits with shiny hard skin" and "elongated yellow fruits".

Q4. What are the various algorithms?

There are various algorithms. Here is a list.





Q5. What is 'Naive' in a Naive Bayes?

https://en.wikipedia.org/wiki/Naive_Bayes_classifier

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. Bayes' theorem states the following relationship, given class variable y and dependent feature vector $x_1$through x_n:

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) P(x_1, \dots, x_n \mid y)}{P(x_1, \dots, x_n)}$$

Using the naive conditional independence assumption that each x_i is independent:
for all i, this relationship is simplified to:

$$P(x_i | y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i | y),$$

Since $P(x_1, \dots, x_n)$ is constant given the input, we can use the following classification rule:

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \dots, x_n)}$$

and we can use Maximum A Posteriori (MAP) estimation to estimate $P(y)$ and $P(y|x_i)$; the former is then the relative frequency of class y in the training set.

$$P(y \mid x_1, \dots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$
$$\Downarrow$$
$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

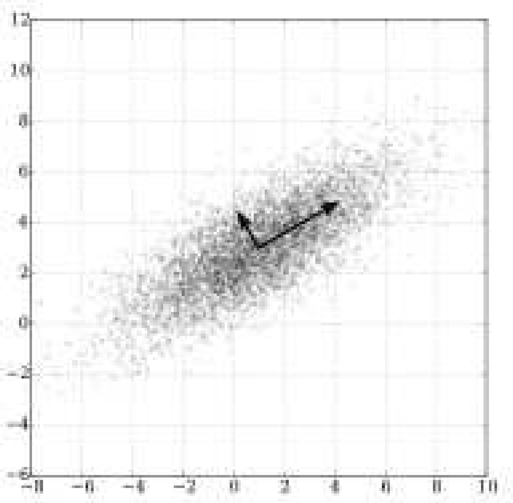
The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(y|x_i)$: can be Bernoulli, Binomial, Gaussian, and so on.

Q6. What is PCA? When do you use it?

https://en.wikipedia.org/wiki/Principal_component_analysis
https://blog.umetrics.com/what-is-principal-component-analysis-pca-and-how-it-is-used
https://blog.umetrics.com/why-preprocesing-data-creates-better-data-analytics-models

Principal component analysis (PCA) is a statistical method used in Machine Learning. It consists in projecting data in a higher dimensional space into a lower dimensional space by maximizing the variance of each dimension.

The process works as following. We define a matrix A with n rows (the single observations of a dataset – in a tabular format, each single row) and p columns, our features. For this matrix we construct a variable space with as many dimensions as there are features. Each feature represents one coordinate axis. For each feature, the length has been standardized according to a scaling criterion, normally by scaling to unit variance. It is determinant to scale the features to a common scale, otherwise the features with a greater magnitude will weigh more in determining the principal components. Once plotted all the observations and computed the mean of each variable, that mean will be represented by a point in the center of our plot (the center of gravity). Then, we subtract each observation with the mean, shifting the coordinate system with the center in the origin. The best fitting line resulting is the line that best accounts for the shape of the point swarm. It represents the maximum variance direction in the data. Each observation may be projected onto this line in order to get a coordinate value along the PC-line. This value is known as a score. The next best-fitting line can be similarly chosen from directions perpendicular to the first. Repeating this process yields an orthogonal basis in which different individual dimensions of the data are uncorrelated. These basis vectors are called **principal components**.



PCA is mostly used as a tool in exploratory data analysis and for making predictive models. It is often used to visualize genetic distance and relatedness between populations.

Q7. Explain SVM algorithm in detail.

https://en.wikipedia.org/wiki/Support_vector_machine

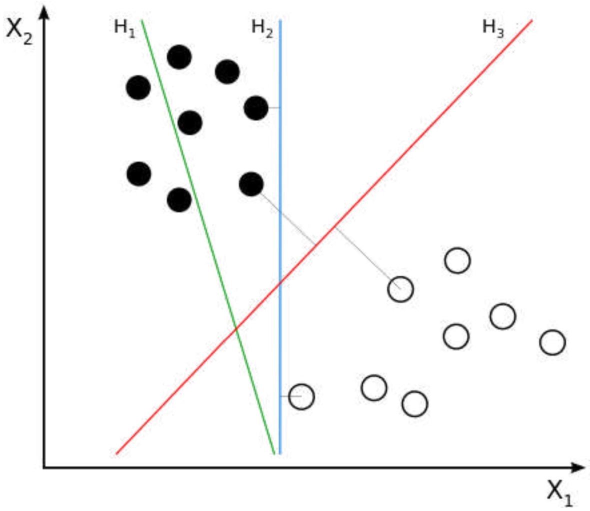
Classifying data is a common task in machine learning. Suppose some given data points each belong to one of two classes, and the goal is to decide which class a new data point will be in. In the case of support-vector machines, a data point is viewed as a p-dimensional vector (a list of p numbers), and we want to know whether we can separate such points with a $(p-1)$-dimensional hyperplane. This is called a linear classifier. There are many hyperplanes that might classify the data. One reasonable choice as the best hyperplane is the one that represents the largest separation, or margin, between the two classes. So, we choose the hyperplane so that the distance from it to the nearest data point on each side is maximized. If such a hyperplane exists, it is known as the maximum-margin hyperplane and the linear classifier it defines is known as a maximum-margin classifier; or equivalently, the perceptron of optimal stability. The best hyper plane that divides the data is H_3.
We have n data $(x_1, y_1), \dots, (x_n, y_n)$ and p different features $x_i = (x_i^1, \dots, x_i^p)$ and y_i is either 1 or -1.
The equation of the hyperplane H_3 is as the set of points x satisfying:

$$w \cdot x - b = 0$$

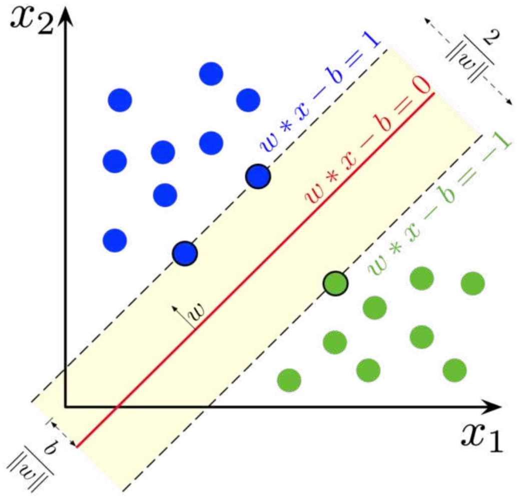
where w is the (not necessarily normalized) normal vector to the hyperplane. The parameter $\frac{b}{\|w\|}$ determines the offset of the hyperplane from the origin along the normal vector w.

So, for each i, either x_i is in the hyperplane of 1 or -1. Basically, x_i satisfies:

$$w \cdot x_i - b \geq 1 \quad or \quad w \cdot x_i - b \leq -1$$



- *SVMs are helpful in text and hypertext categorization, as their application can significantly reduce the need for labeled training instances in both the standard inductive and transductive settings. Some methods for shallow semantic parsing are based on support vector machines.*
- *Classification of images can also be performed using SVMs. Experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback.*
- *Classification of satellite data like SAR data using supervised SVM.*
- *Hand-written characters can be recognized using SVM.*



Q8. What are the support vectors in SVM?

In the diagram, we see that the sketched lines mark the distance from the classifier (the hyper plane) to the closest data points called the support vectors (darkened data points). The distance between the two thin lines is called the margin.

To extend SVM to cases in which the data are not linearly separable, we introduce the *hinge loss* function,

$$\max\left(0, 1 - y_i(w \cdot x_i - b)\right)$$

This function is zero if x lies on the correct side of the margin. For data on the wrong side of the margin, the function's value is proportional to the distance from the margin.

Q9. What are the different kernels in SVM?

There are four types of kernels in SVM.

1. LinearKernel
2. Polynomial kernel
3. Radial basis kernel
4. Sigmoid kernel

Q10. What are the most known ensemble algorithms?

https://towardsdatascience.com/the-ultimate-guide-to-adaboost-random-forests-and-xgboost-7f9327061c4f

The most popular trees are: AdaBoost, Random Forest, and eXtreme Gradient Boosting (XGBoost).

AdaBoost is **best used** in a dataset with low noise, when computational complexity or timeliness of results is not a main concern and when there are not enough resources for broader hyperparameter tuning due to lack of time and knowledge of the user.

Random forests should not be used when dealing with time series data or any other data where look-ahead bias should be avoided, and the order and continuity of the samples need to be ensured. This algorithm can handle noise relatively well, but more knowledge from the user is required to adequately tune the algorithm compared to AdaBoost.

The main advantages of XGBoost is its lightning speed compared to other algorithms, such as AdaBoost, and its regularization parameter that successfully reduces variance. But even aside from the regularization parameter, this algorithm leverages a learning rate (shrinkage) and subsamples from the features like random forests, which increases its ability to generalize even further. However, XGBoost is more difficult to understand, visualize and to tune compared to AdaBoost and random forests. There is a multitude of hyperparameters that can be tuned to increase performance.

Q11. Explain Decision Tree algorithm in detail.

https://en.wikipedia.org/wiki/Decision_tree_learning
https://www.kdnuggets.com/2019/02/decision-trees-introduction.html/2
https://medium.com/@naeemsunesara/giniscore-entropy-and-information-gain-in-decision-trees-cbc08589852d

A decision tree is a supervised machine learning algorithm mainly used for Regression and Classification. It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision tree can handle both categorical and numerical data. The term Classification and Regression Tree (CART) analysis is an umbrella term used to refer to both of the above procedures.

Some techniques, often called *ensemble* methods, construct more than one decision tree:

- **Boosted trees** Incrementally building an ensemble by training each new instance to emphasize the training instances previously mis-modeled. A typical example is AdaBoost. These can be used for regression-type and classification-type problems.
- **Bootstrap aggregated** (or bagged) decision trees, an early ensemble method, builds multiple decision trees by repeatedly resampling training data with replacement, and voting the trees for a consensus prediction.
    - A **random forest** classifier is a specific type of bootstrap aggregating.
- **Rotation forest** – in which every decision tree is trained by first applying principal component analysis (PCA) on a random subset of the input features.

A special case of a decision tree is a decision list, which is a one-sided decision tree, so that every internal node has exactly 1 leaf node and exactly 1 internal node as a child (except for the bottommost node, whose only child is a single leaf node). While less expressive, decision lists are arguably easier to understand than general decision trees due to their added sparsity, permit non-greedy learning methods and monotonic constraints to be imposed.

Notable decision tree algorithms include:

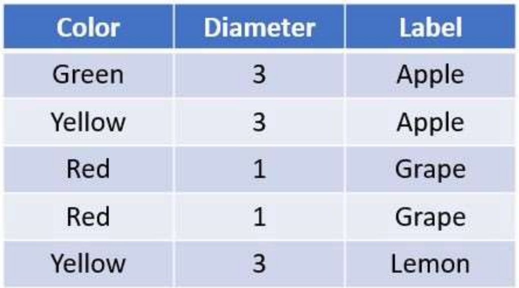
- ID3 (Iterative Dichotomiser 3)
- C4.5 (successor of ID3)
- CART (Classification and Regression Tree)
- Chi-square automatic interaction detection (CHAID). Performs multi-level splits when computing classification trees.
- MARS: extends decision trees to handle numerical data better.
- Conditional Inference Trees. Statistics-based approach that uses non-parametric tests as splitting criteria, corrected for multiple testing to avoid overfitting. This approach results in unbiased predictor selection and does not require pruning.

Q12. What are Entropy and Information gain in Decision tree algorithm?

https://www.saedsayad.com/decision_tree.htm
https://medium.com/@naeemsunesara/giniscore-entropy-and-information-gain-in-decision-trees-cbc08589852d

There are a lot of algorithms which are employed to build a decision tree, ID3 (Iterative Dichotomiser 3), C4.5, C5.0, CART (Classification and Regression Trees) to name a few but at their core all of them tell us what questions to ask and when.

The below table has color and diameter of a fruit and the label tells the name of the fruit. How do we build a decision tree to classify the fruits?



Color	Diameter	Label
Green	3	Apple
Yellow	3	Apple
Red	1	Grape
Red	1	Grape
Yellow	3	Lemon

Here is how we will build the tree. We will start with a node which will ask a true or false question to split the data into two. The two resulting nodes will each ask a true or false question again to split the data further and so on.

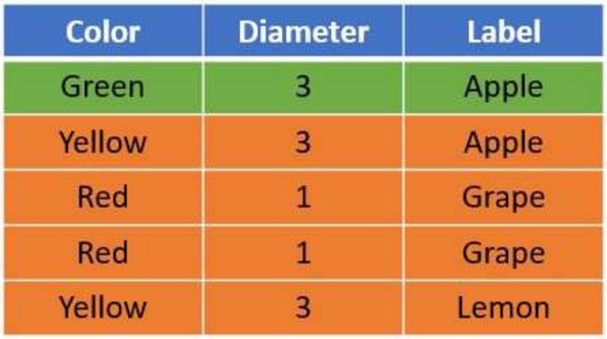
There are 2 main things to consider with the above approach:

- Which is the best question to ask at each node
- When do we stop splitting the data further?

Let's start building the tree with the first or the topmost node. There is a list of possible questions which can be asked. The first node can ask the following questions:

- *Is the color green?*
- *Is the color yellow?*
- *Is the color red?*
- *Is the diameter ≥ 3?*
- *Is the diameter ≥ 1?*

Of these possible set of questions, which one is the best to ask so that our data is split into two sets after the first node? Remember we are trying to split or classify our data into separate classes. Our question should be such that our data is partitioned into as unmixed or pure classes as possible. An impure set or class here refers to one which has many different types of objects *for example if we ask the question for the above data, "Is the color green?" our data will be split into two sets one of which will be pure the other will have a mixed set of labels.* If we assign a label to a mixed set, we have higher chances of being incorrect. But how do we measure this impurity?



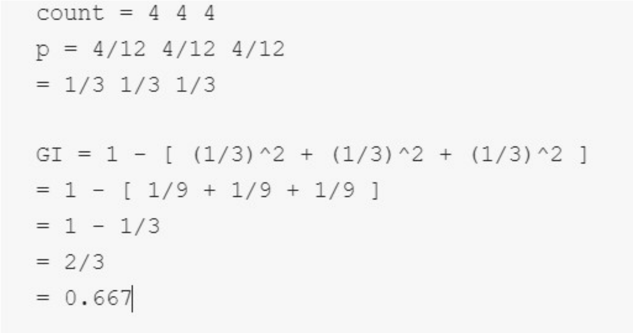
Color	Diameter	Label
Green	3	Apple
Yellow	3	Apple
Red	1	Grape
Red	1	Grape
Yellow	3	Lemon

Gini Impurity and Information Gain - CART

CART (Classification and Regression Trees) → uses ***Gini Index (Classification)*** as metric.

The Gini Impurity (GI) metric measures the homogeneity of a set of items. The lowest possible value of GI is 0.0. The maximum value of GI depends on the particular problem being investigated but gets close to 1.0.

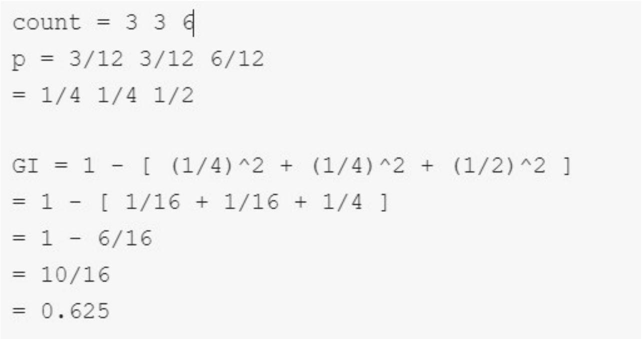
Suppose for example you have 12 items — apples, grapes, lemons. If there are 0 apples, 0 grapes, 12 lemons, then you have minimal impurity (this is good for decision trees) and GI = 0.0. But if you have 4 apples, 4 grapes, 4 lemons, you have maximum impurity and it turns out that GI = 0.667.
I'll show example calculations.
Maximum GI: Apples, Grapes, Lemons



```
count = 4 4 4
p = 4/12 4/12 4/12
= 1/3 1/3 1/3

GI = 1 - [ (1/3)^2 + (1/3)^2 + (1/3)^2 ]
= 1 - [ 1/9 + 1/9 + 1/9 ]
= 1 - 1/3
= 2/3
= 0.667
```

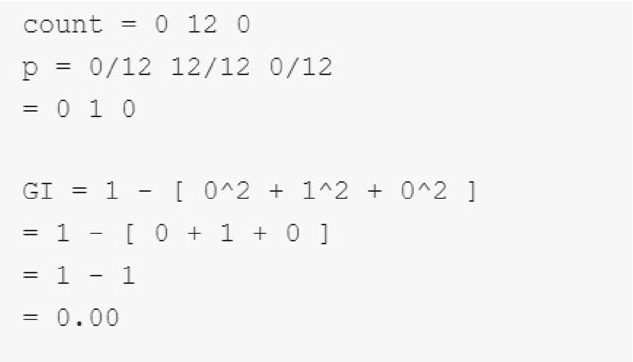
When the number of items is evenly distributed, as in the example above, you have maximum GI but the exact value depends on how many items there are. A bit less than maximum GI:



```
count = 3 3 6
p = 3/12 3/12 6/12
= 1/4 1/4 1/2

GI = 1 - [ (1/4)^2 + (1/4)^2 + (1/2)^2 ]
= 1 - [ 1/16 + 1/16 + 1/4 ]
= 1 - 6/16
= 10/16
= 0.625
```

In the example above, the items are not quite evenly distributed, and the GI is slightly less (which is better when used for decision trees). Minimum GI:



```
count = 0 12 0
p = 0/12 12/12 0/12
= 0 1 0

GI = 1 - [ 0^2 + 1^2 + 0^2 ]
= 1 - [ 0 + 1 + 0 ]
= 1 - 1
= 0.00
```

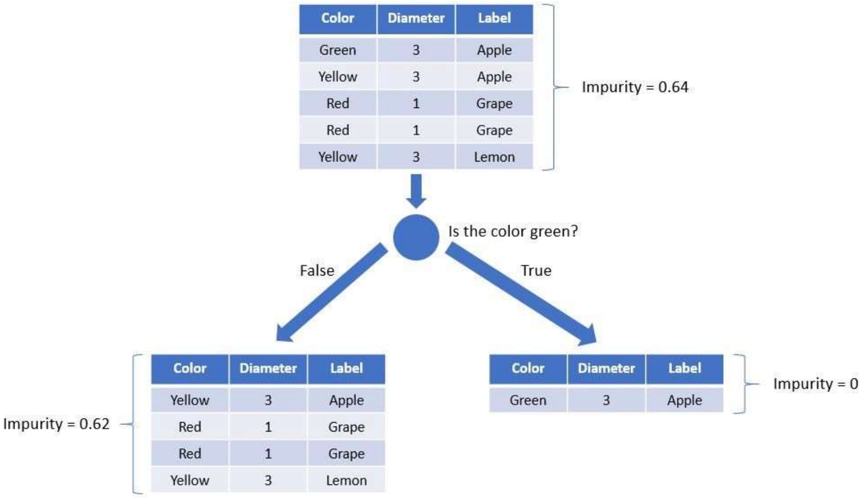
J classes $\{1, 2, \dots J\}$, p_i fraction of elements of class i:

Gini impurity: $I_G(p) = 1 - \sum_{i=1}^{J} p_i^2$

The ***Gini index*** is not at all the same as a different metric called the ***Gini coefficient***. The Gini impurity metric can be used when creating a decision tree but there are alternatives, including ***Entropy Information gain***. The advantage of GI is its simplicity.

Information Gain

Information gain is another metric which tells us how much a question *unmixes* the labels at a node. ***"Mathematically it is just a difference between impurity values before splitting the data at a node and the weighted average of the impurity after the split"***. *For instance, if we go back to our data of apples, lemons and grapes and ask the question "Is the color Green?"*



Color	Diameter	Label
Green	3	Apple
Yellow	3	Apple
Red	1	Grape
Red	1	Grape
Yellow	3	Lemon

Impurity = 0.64

Color	Diameter	Label
Yellow	3	Apple
Red	1	Grape
Red	1	Grape
Yellow	3	Lemon

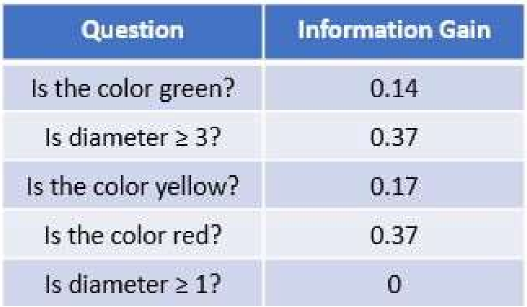
Impurity = 0.62

Color	Diameter	Label
Green	3	Apple

Impurity = 0

$$Information\ Gain = 0.64 - \left(\frac{4}{5} * 0.62 + \frac{1}{5} * 0\right) = 0.144$$

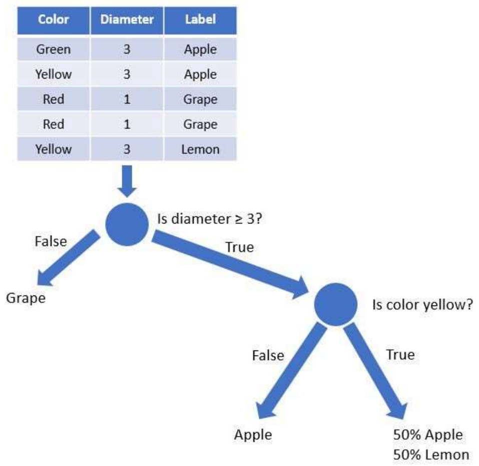
The information gain by asking this question is 0.144. Similarly, we can ask another question from the set of possible questions split the data and compute information gain. This is also called ***(Recursive Binary Splitting).***



Question	Information Gain
Is the color green?	0.14
Is diameter ≥ 3?	0.37
Is the color yellow?	0.17
Is the color red?	0.37
Is diameter ≥ 1?	0

The question where we have the highest information gain *"Is diameter ≥ 3?"* is the best question to ask. Note that the information gain is same for the question *"Is the color red?"* we just picked the first one at random.

Repeating the same method at the child node we can complete the tree. Note that no further questions can be asked which would increase the information gain.

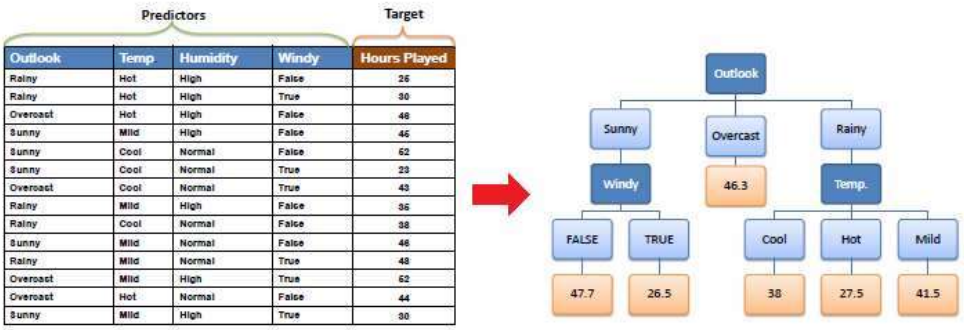


Color	Diameter	Label
Green	3	Apple
Yellow	3	Apple
Red	1	Grape
Red	1	Grape
Yellow	3	Lemon

Also note that the rightmost leaf which says 50% Apple & 50% lemon means that this class cannot be divided further, and this branch can tell an apple or a lemon with 50% probability. For the grape and apple branches we stop asking further questions since the Gini Impurity is 0 for those.

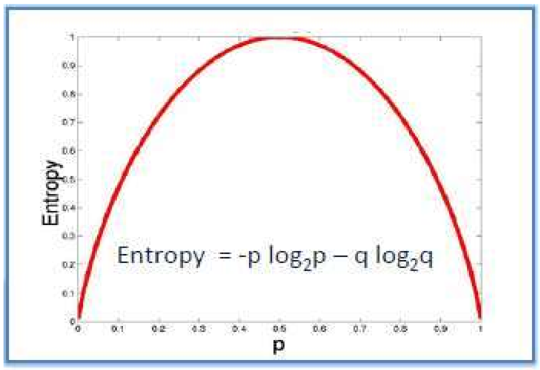
Entropy and Information Gain – ID3

ID3 (Iterative Dichotomiser 3) → uses ***Entropy function*** and ***Information gain*** as metrics.



Predictors				Target
Outlook	Temp	Humidity	Windy	Hours Played
Rainy	Hot	High	False	25
Rainy	Hot	High	True	30
Overcast	Hot	High	False	46
Sunny	Mild	High	False	45
Sunny	Cool	Normal	False	52
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	35
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	46
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	52
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30

If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one.



$$\text{Entropy} = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$

To build a decision tree, we need to calculate two types of entropy using frequency tables as follows:

a) Entropy using the frequency table of one attribute:

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Play Golf	
Yes	No
9	5

Entropy(PlayGolf) = Entropy (5,9)

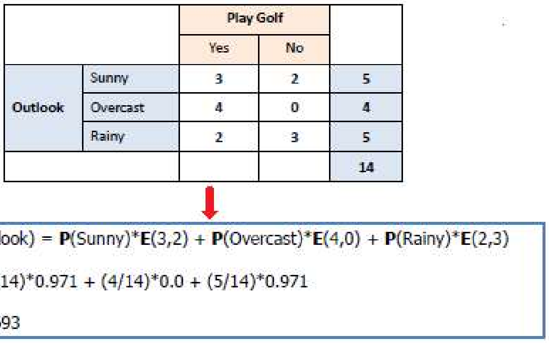
= Entropy (0.36, 0.64)

= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)

= 0.94

b) Entropy using the frequency table of two attributes:

$$E(T,X) = \sum_{c \in X} P(c)E(c)$$



		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

E(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)

= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971

= 0.693

Information Gain

The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).

Step 1: Calculate entropy of the target.

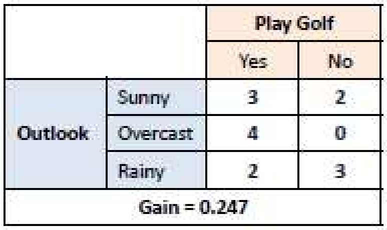
$$\text{Entropy(PlayGolf)} = \text{Entropy}\ (5,9)$$

$$= \text{Entropy}\ (0.36, 0.64)$$

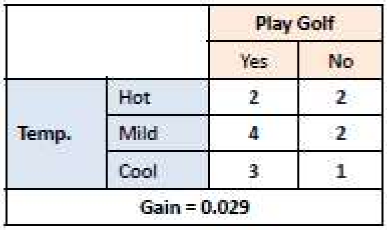
$$= -\ (0.36 \log_2 0.36) - (0.64 \log_2 0.64)$$

$$= 0.94$$

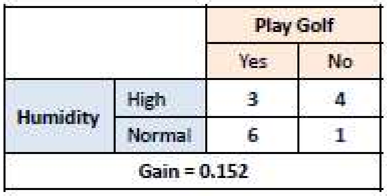
Step 2: The dataset is then split on the different attributes. The entropy for each branch is calculated. Then it is added proportionally, to get total entropy for the split. The resulting entropy is subtracted from the entropy before the split. The result is the Information Gain or decrease in entropy.



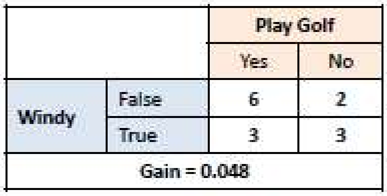
		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			



		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			



		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			



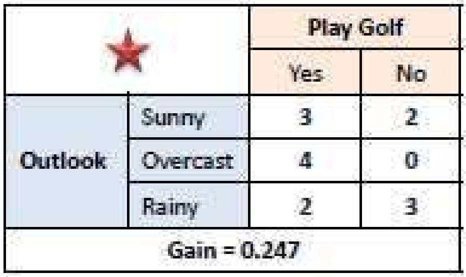
		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

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

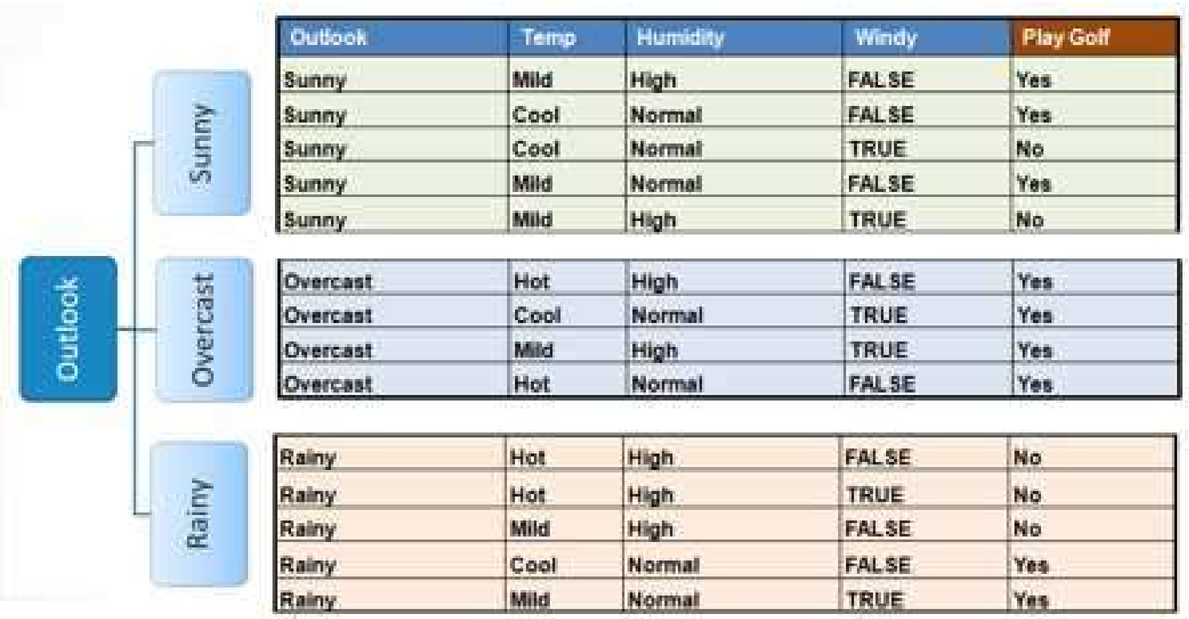
G(PlayGolf, Outlook) = **E**(PlayGolf) – **E**(PlayGolf, Outlook)

= 0.940 – 0.693 = 0.247

Step 3: Choose attribute with the largest information gain as the decision node, divide the dataset by its branches and repeat the same process on every branch.

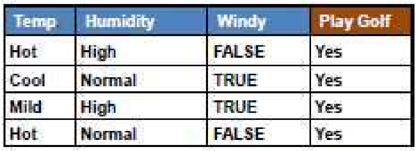


★		**Play Golf**	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

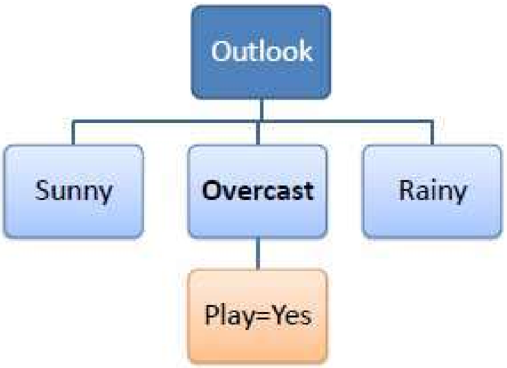


Outlook	Temp	Humidity	Windy	Play Golf
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Overcast	Cool	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes

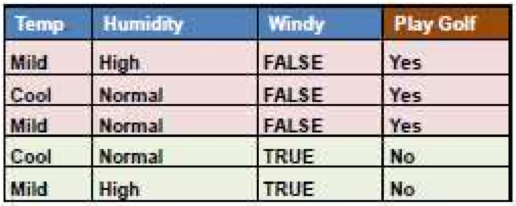
Step 4a: A branch with entropy of 0 is a leaf node.



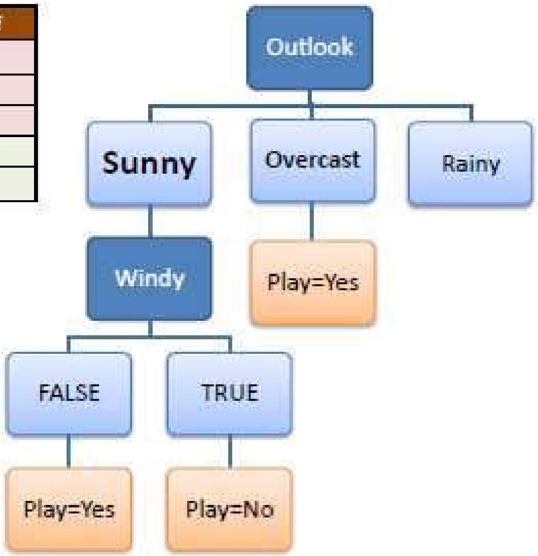
Temp	Humidity	Windy	Play Golf
Hot	High	FALSE	Yes
Cool	Normal	TRUE	Yes
Mild	High	TRUE	Yes
Hot	Normal	FALSE	Yes



Step 4b: A branch with entropy more than 0 needs further splitting.



Temp	Humidity	Windy	Play Golf
Mild	High	FALSE	Yes
Cool	Normal	FALSE	Yes
Mild	Normal	FALSE	Yes
Cool	Normal	TRUE	No
Mild	High	TRUE	No



Step 5: The ID3 algorithm is run recursively on the non-leaf branches, until all data is classified.

Q13. What is pruning in Decision Tree?

Pruning is a technique in machine learning and search algorithms that reduces the size of decision trees by removing sections of the tree that provide little power to classify instances. So, when we remove sub-nodes of a decision node, this process is called pruning or opposite process of splitting.

Q14. What is logistic regression? State an example when you have used logistic regression recently.

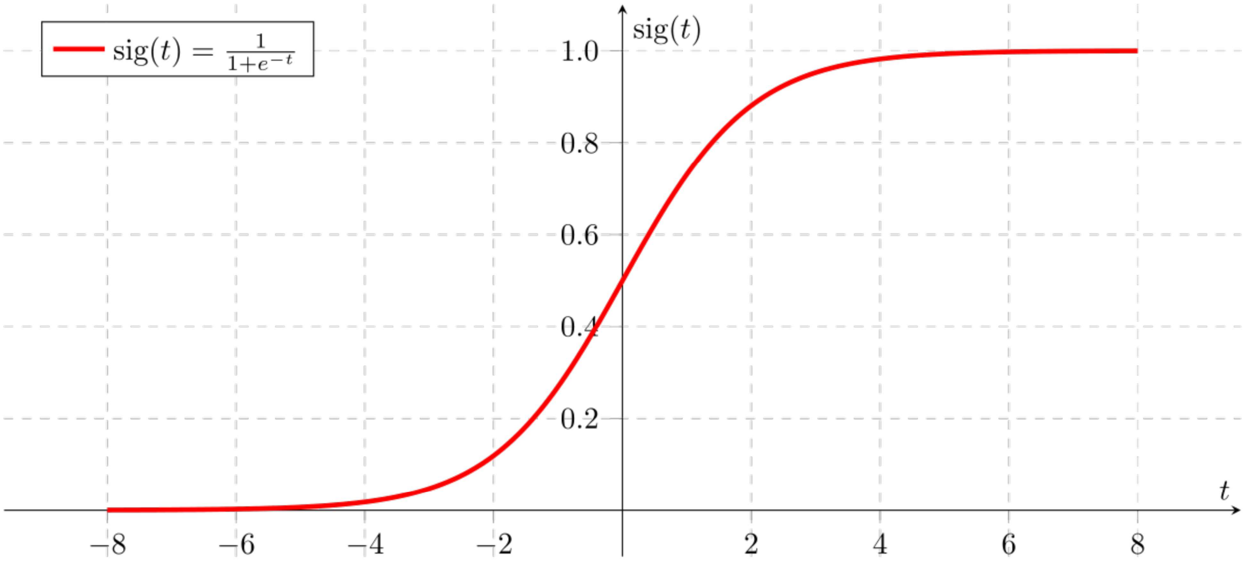
Logistic Regression often referred to as the logit model is a technique to predict the binary outcome from a linear combination of predictor variables. Since we are interested in a probability outcome, a line does not fit the model. Logistic Regression is a classification algorithm that works by trying to learn a function that approximates $P(Y|X)$. It makes the central assumption that $P(X|Y)$ can be approximated as a sigmoid function applied to a linear combination of input features.

- $P(Y=1|X) = p \Rightarrow$ assume $\log \frac{p}{1-p} = b_0 + \sum_{i=1}^{p} b_i X_i \Leftrightarrow \frac{p}{1-p} = e^{b_0 + b^T X}$

$$\Leftrightarrow p = \frac{e^{b_0 + b^T X}}{1 + e^{b_0 + b^T X}}$$

$$\Leftrightarrow p = \frac{1}{1 + e^{-\underbrace{(b_0 + b^T X)}_{z}}} = sig(z)$$

- $P(Y=0|X) = 1 - sig(z)$

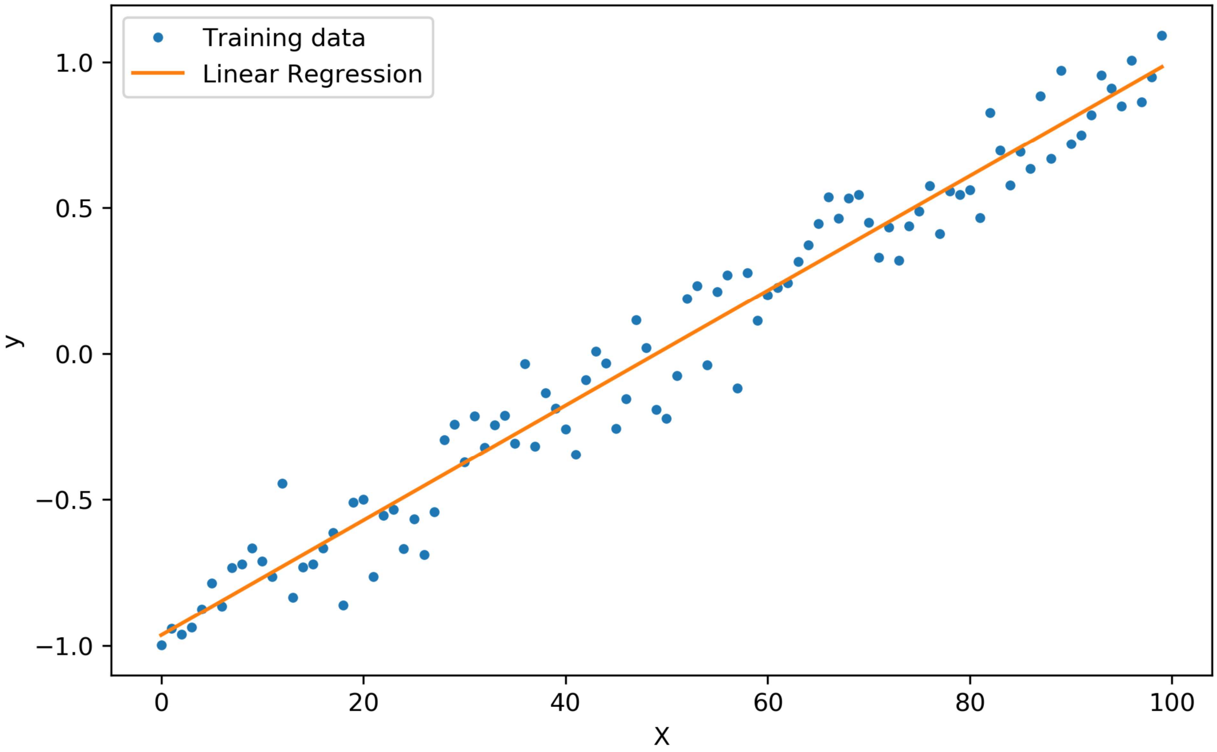


For example, if you want to predict whether a particular political leader will win the election or not. In this case, the outcome of prediction is binary i.e. 0 or 1 (Win/Lose). The predictor variables here would be the amount of money spent for election campaigning of a particular candidate, the amount of time spent in campaigning, etc.

Q15. What is Linear Regression?

Linear regression is a statistical technique where the score of a variable Y is predicted from the score of a second variable X. X is referred to as the predictor variable and Y as the criterion variable.

$$Y = b_0 + b_1 X_1 + \cdots + b_p X_p$$



Q16. What Are the Drawbacks of the Linear Model?

Some drawbacks of the linear model are:

- The assumption of linearity of the model
- It can't be used for count outcomes or binary outcomes.
- There are overfitting or underfitting problems that it can't solve.

Q17. What is the difference between Regression and classification ML techniques?

Both Regression and classification machine learning techniques come under Supervised machine learning algorithms. In Supervised machine learning algorithm, we have to train the model using labelled data set, while training we have to explicitly provide the correct labels and algorithm tries to learn the pattern from input to output. If our labels are discrete values then it will a classification problem, but if our labels are continuous values then it will be a regression problem.

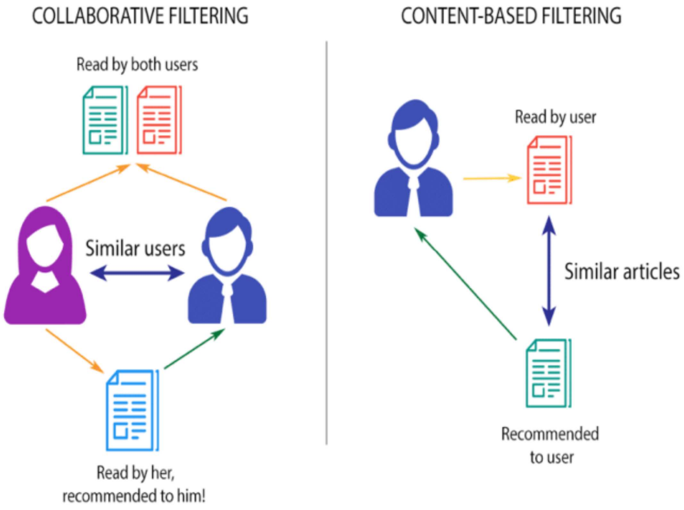
Q18. What are Recommender Systems?

https://en.wikipedia.org/wiki/Recommender_system

Recommender Systems are a subclass of information filtering systems that are meant to predict the preferences or ratings that a user would give to a product. Recommender systems are widely used in movies, news, research articles, products, social tags, music, etc.
Examples include movie recommenders in IMDB, Netflix & BookMyShow, product recommenders in e-commerce sites like Amazon, eBay & Flipkart, YouTube video recommendations and game recommendations in Xbox.

Q19. What is Collaborative filtering? And a content based?

The process of filtering used by most of the recommender systems to find patterns or information by collaborating viewpoints, various data sources and multiple agents. Collaborative filtering is a technique that can filter out items that a user might like on the basis of reactions by similar users. It works by searching a large group of people and finding a smaller set of users with tastes similar to a particular user. It looks at the items they like (usually based on rating) and combines them to create a ranked list of suggestions. Similar users are those with similar rating and on the based on that they get recommendations. In content based, we look only at the item level, recommending on similar items sold.



An example of collaborative filtering can be to predict the rating of a particular user based on his/her ratings for other movies and others' ratings for all movies. This concept is widely used in recommending movies in IMDB, Netflix & BookMyShow, product recommenders in e-commerce sites like Amazon, eBay & Flipkart, YouTube video recommendations and game recommendations in Xbox.

Q20. How can outlier values be treated?

Outlier values can be identified by using univariate or any other graphical analysis method. If the number of outlier values is few then they can be assessed individually but for a large number of outliers, the values can be substituted with either the 99th or the 1st percentile values.
All extreme values are not outlier values. The most common ways to treat outlier values:

1. Change it with a mean or median
2. Standardize the feature, changing the distribution but smoothing the outliers
3. Log transform the feature (with many outliers)
4. Drop the value

5. First/third quartile value if more than 2σ

Q21. What are the various steps involved in an analytics project?

The following are the various steps involved in an analytics project:

1. Understand the Business problem
2. Explore the data and become familiar with it
3. Prepare the data for modeling by detecting outliers, treating missing values, transforming variables, etc.
4. After data preparation, start running the model, analyze the result and tweak the approach. This is an iterative step until the best possible outcome is achieved.
5. Validate the model using a new data set.
6. Start implementing the model and track the result to analyze the performance of the model over the period of time.

Q22. During analysis, how do you treat missing values?

The extent of the missing values is identified after identifying the variables with missing values. If any patterns are identified the analyst has to concentrate on them as it could lead to interesting and meaningful business insights.

If there are no patterns identified, then the missing values can be substituted with mean or median values (imputation) or they can simply be ignored. Assigning a default value which can be mean, minimum or maximum value. Getting into the data is important.

If it is a categorical variable, the default value is assigned. The missing value is assigned a default value. If you have a distribution of data coming, for normal distribution give the mean value.

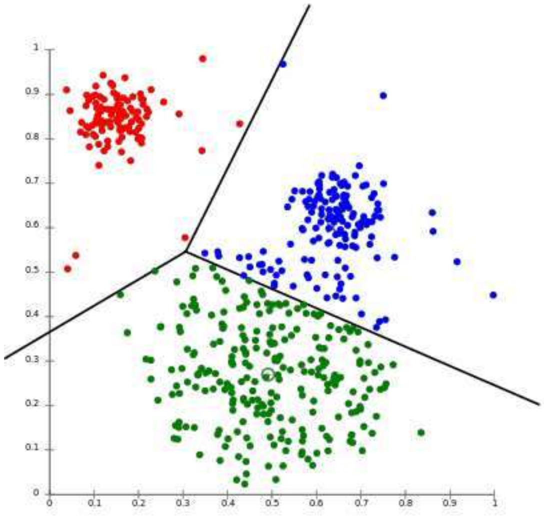
If 80% of the values for a variable are missing, then you can answer that you would be dropping the variable instead of treating the missing values.

Q23. How will you define the number of clusters in a clustering algorithm?

https://stackabuse.com/hierarchical-clustering-with-python-and-scikit-learn/

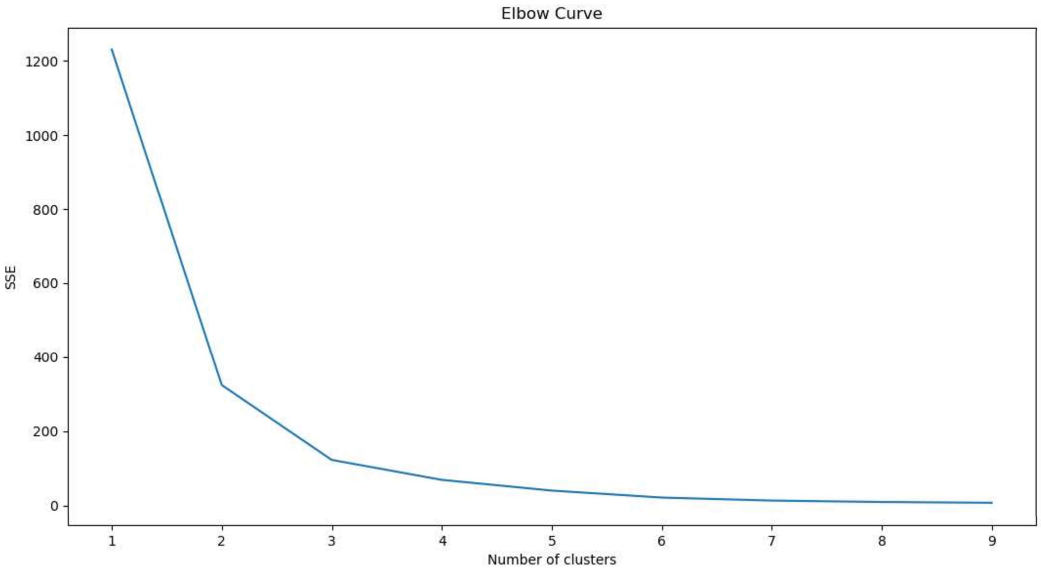
Though the Clustering Algorithm is not specified, this question is mostly in reference to K-Means clustering where “K” defines the number of clusters. The objective of clustering is to group similar entities in a way that the entities within a group are similar to each other, but the groups are different from each other.

For example, the following image shows three different groups.

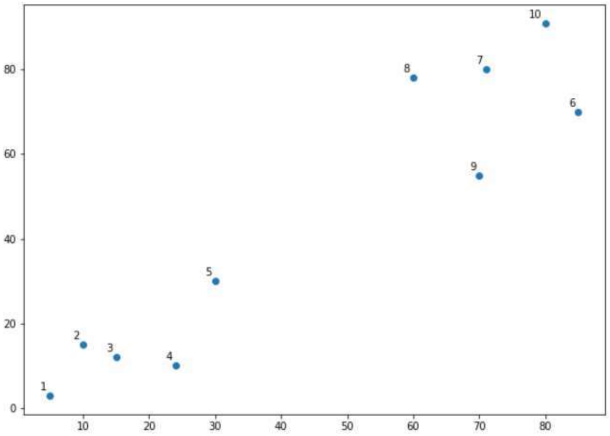


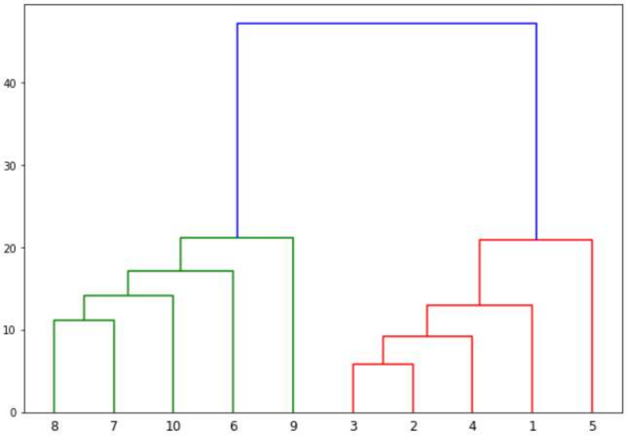
Within Sum of squares is generally used to explain the homogeneity within a cluster. If you plot WSS (as the sum of the squared distance between each member of the cluster and its centroid) for a range of number of clusters, you will get the plot shown below.

- The Graph is generally known as Elbow Curve.
- Red circled a point in above graph i.e. Number of Cluster = 3 is the point after which you don't see any decrement in WSS.
- This point is known as the bending point and taken as K in K – Means.

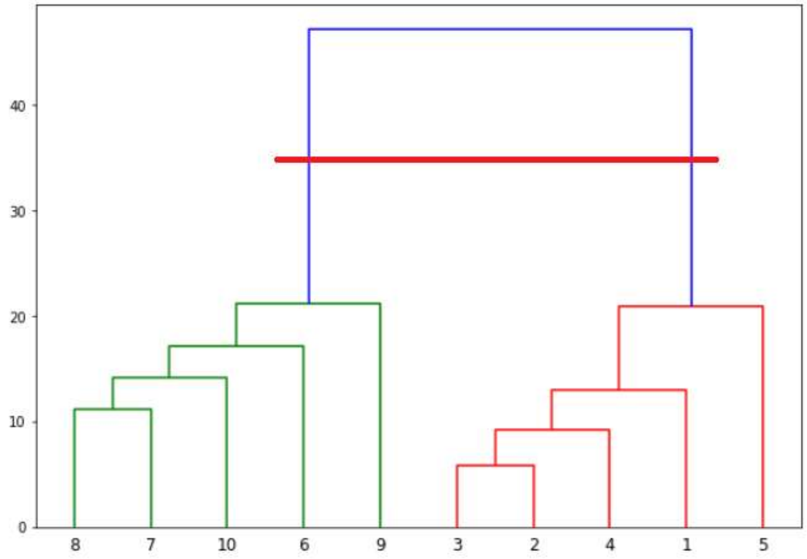


This is the widely used approach but few data scientists also use Hierarchical clustering first to create dendrograms and identify the distinct groups from there.

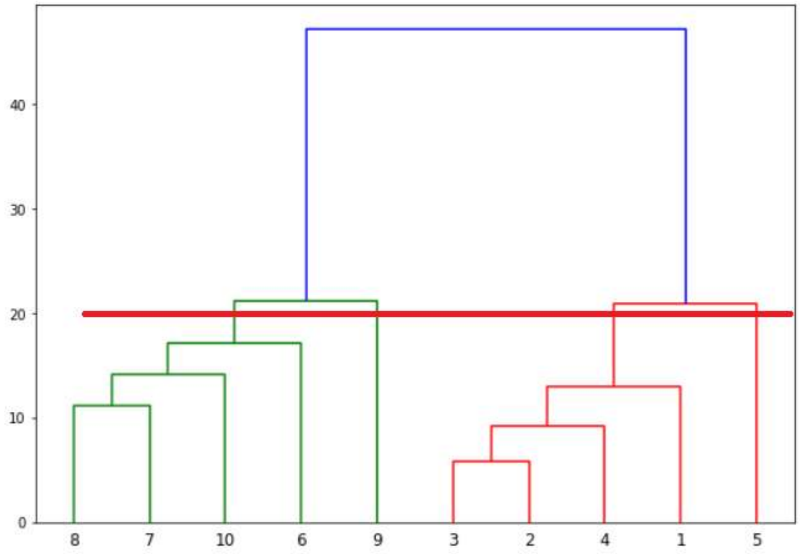




The algorithm starts by finding the two points that are closest to each other on the basis of Euclidean distance. If we look back at Graph1, we can see that points 2 and 3 are closest to each other while points 7 and 8 are closes to each other. Therefore a cluster will be formed between these two points first. In Graph2, you can see that the dendograms have been created joining points 2 with 3, and 8 with 7. The vertical height of the dendogram shows the Euclidean distances between points. From Graph2, it can be seen that Euclidean distance between points 8 and 7 is greater than the distance between point 2 and 3.
The next step is to join the cluster formed by joining two points to the next nearest cluster or point which in turn results in another cluster. If you look at Graph1, point 4 is closest to cluster of point 2 and 3, therefore in Graph2 dendrogram is generated by joining point 4 with dendrogram of point 2 and 3. This process continues until all the points are joined together to form one big cluster.
Once one big cluster is formed, the longest vertical distance without any horizontal line passing through it is selected and a horizontal line is drawn through it. The number of vertical lines this newly created horizontal line passes is equal to number of clusters. Take a look at the following plot:



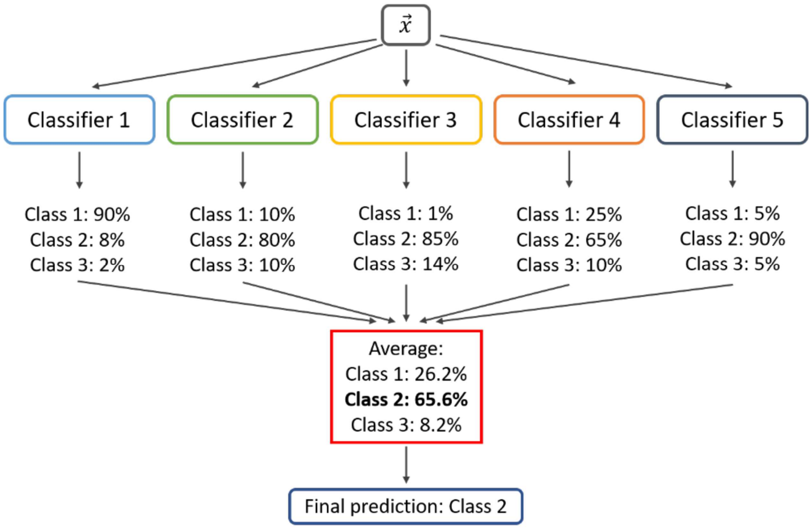
We can see that the largest vertical distance without any horizontal line passing through it is represented by blue line. So we draw a new horizontal red line that passes through the blue line. Since it crosses the blue line at two points, therefore the number of clusters will be 2. Basically the horizontal line is a threshold, which defines the minimum distance required to be a separate cluster. If we draw a line further down, the threshold required to be a new cluster will be decreased and more clusters will be formed as see in the image below:



In the above plot, the horizontal line passes through four vertical lines resulting in four clusters: cluster of points 6,7,8 and 10, cluster of points 3,2,4 and points 9 and 5 will be treated as single point clusters.

Q24. What is Ensemble Learning?

In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. Ensembles are a divide-and-conquer approach used to improve performance. The main principle behind ensemble methods is that a group of “weak learners” can come together to form a “strong



learner". Each classifier, individually, is a "weak learner," while all the classifiers taken together are a "strong learner".

Q25. Describe in brief any type of Ensemble Learning.

https://medium.com/@ruhi3929/bagging-and-boosting-method-c036236376eb

Ensemble learning has many types but two more popular ensemble learning techniques are mentioned below.

Bagging

Bagging tries to implement similar learners on small sample populations and then takes a mean of all the predictions. In generalized bagging, you can use different learners on different population. As you expect this helps us to reduce the variance error.

Pros

- Bagging method helps when we face variance or overfitting in the model. It provides an environment to deal with variance by using N learners of same size on same algorithm.
- During the sampling of train data, there are many observations which overlaps. So, the combination of these learners helps in overcoming the high variance.
- Bagging uses Bootstrap sampling method (Bootstrapping is any test or metric that uses random sampling with replacement and falls under the broader class of resampling methods.)

Cons

- Bagging is not helpful in case of bias or underfitting in the data.
- Bagging ignores the value with the highest and the lowest result which may have a wide difference and provides an average result.

Boosting

Boosting is an iterative technique which adjusts the weight of an observation based on the last classification. If an observation was classified incorrectly, it tries to increase the weight of this observation and vice versa. Boosting in general decreases the bias error and builds strong predictive models. However, they may over fit on the training data.

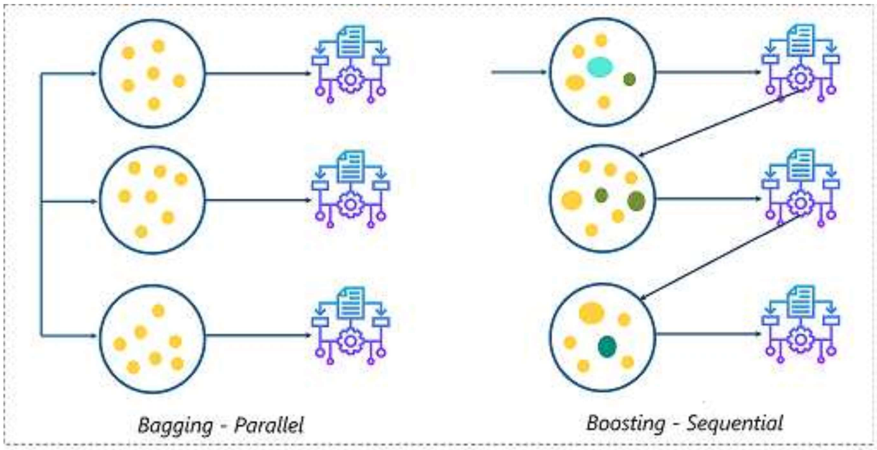
Pros

- Boosting technique takes care of the weightage of the higher accuracy sample and lower accuracy sample and then gives the combined results.
- Net error is evaluated in each learning steps. It works good with interactions.
- Boosting technique helps when we are dealing with bias or underfitting in the data set.
- Multiple boosting techniques are available. *For example: AdaBoost, LPBoost, XGBoost, GradientBoost, BrownBoost*

Cons

- Boosting technique often ignores overfitting or variance issues in the data set.

- It increases the complexity of the classification.
- Time and computation can be a bit expensive.



There are multiple areas where Bagging and Boosting technique is used to boost the accuracy.

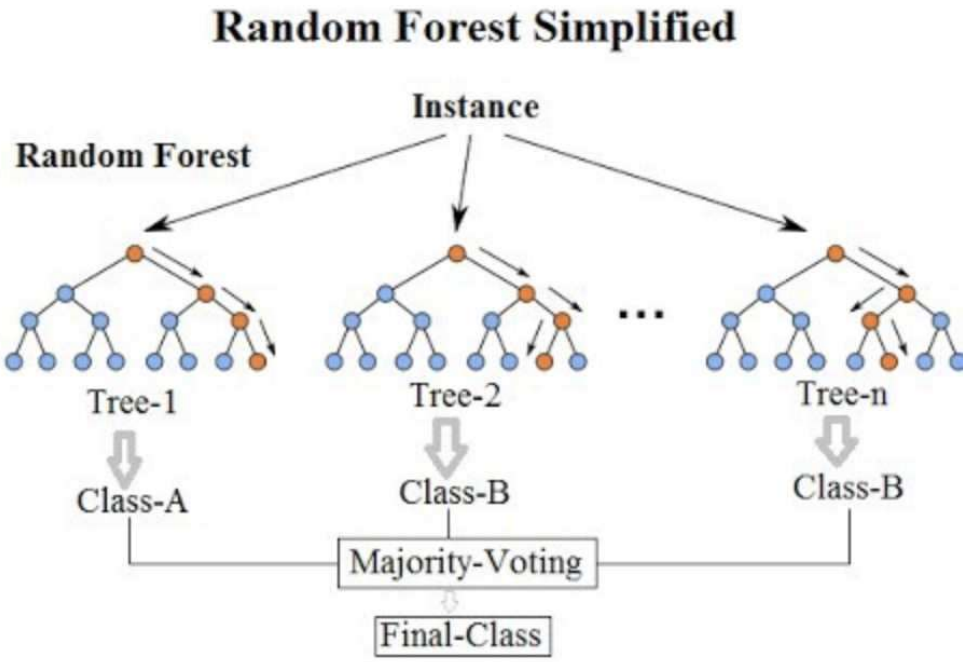
- Banking: Loan defaulter prediction, fraud transaction
- Credit risks
- Kaggle competitions
- Fraud detection
- Recommender system for Netflix
- Malware
- Wildlife conservations and so on.

Q26. What is a Random Forest? How does it work?

Random forest is a versatile machine learning method capable of performing:

- regression
- classification
- dimensionality reduction
- treat missing values
- outlier values

It is a type of ensemble learning method, where a group of weak models combine to form a powerful model. The random forest starts with a standard machine learning technique called a “decision tree” which, in ensemble terms, corresponds to our weak learner. In a decision tree, an input is entered at the top and as it traverses down the tree the data gets bucketed into smaller and smaller sets.



In Random Forest, we grow multiple trees as opposed to a single tree. To classify a new object based on attributes, each tree gives a classification. The forest chooses the classification having the most votes (Over all the trees in the forest) and in case of regression, it takes the average of outputs by different trees.

Q27. How Do You Work Towards a Random Forest?

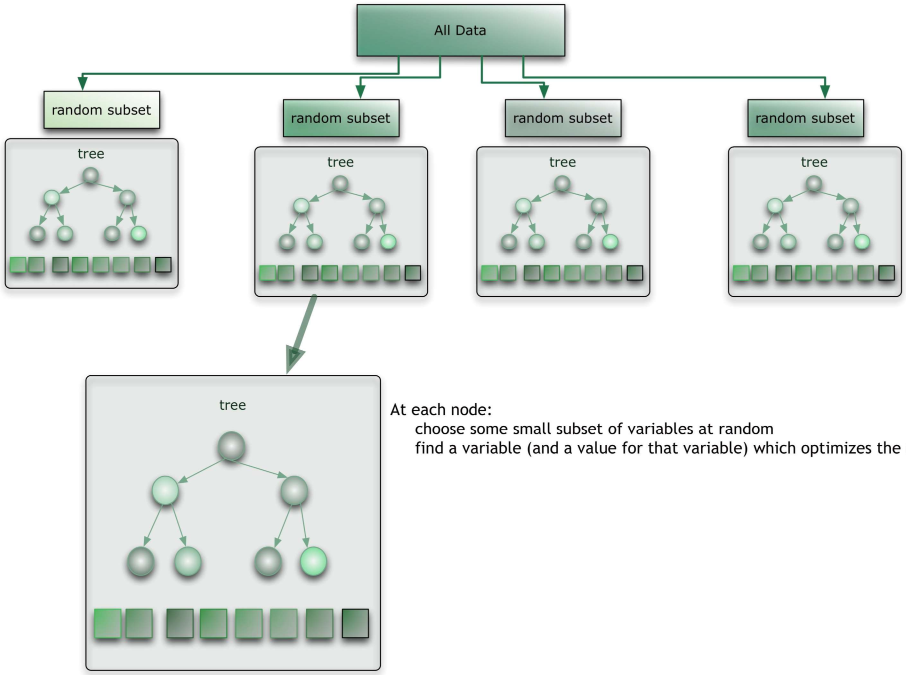
https://blog.citizennet.com/blog/2012/11/10/random-forests-ensembles-and-performance-metrics

The underlying principle of this technique is that several weak learners combined to provide a keen learner. Here is how such a system is trained for some number of trees *T*:

1. Sample *N* cases at random with replacement to create a subset of the data. The subset should be about 66% of the total set.
2. At each node:
    a. For some number *m* (see below), *m* predictor variables are selected at random from all the predictor variables.
    b. The predictor variable that provides the best split, according to some objective function, is used to do a binary split on that node.
    c. At the next node, choose another *m* variables at random from all predictor variables and do the same.

Depending upon the value of *m*, there are three slightly different systems:

- Random splitter selection: $m = 1$
- Breiman’s bagger: m = total number of predictor variables (p)
- Random forest: $m \ll$ number of predictor variables.
    - Brieman suggests three possible values for m: $\frac{1}{2}\sqrt{p},\ \sqrt{p},\ 2\sqrt{p}$



At each node:
choose some small subset of variables at random
find a variable (and a value for that variable) which optimizes the split

When a new input is entered into the system, it is run down all of the trees. The result may either be an average or weighted average of all of the terminal nodes that are reached, or, in the case of categorical variables, a voting majority.

Note that:

- With a large number of predictors $(p \gg 0)$, the eligible predictor set (m) will be quite different from node to node.
- The greater the inter-tree correlation, the greater the random forest error rate, so one pressure on the model is to have the trees as uncorrelated as possible.
- As m goes down, both inter-tree correlation and the strength of individual trees go down. So some optimal value of m must be discovered.
- Strengths: Random forest runtimes are quite fast, and they are able to deal with unbalanced and missing data.
- Weaknesses: Random Forest used for regression cannot predict beyond the range in the training data, and that they may over-fit data sets that are particularly noisy. Of course, the best test of any algorithm is how well it works upon your own data set.

Q28. What cross-validation technique would you use on a time series data set?

Instead of using k-fold cross-validation, you should be aware of the fact that a time series is not randomly distributed data — It is inherently ordered by chronological order.
In case of time series data, you should use techniques like forward=chaining — Where you will be model on past data then look at forward-facing data.

fold 1: training[1], test[2]
fold 2: training[1 2], test[3]
fold 3: training[1 2 3], test[4]
fold 4: training[1 2 3 4], test[5]

Q29. What is a Box-Cox Transformation?

The dependent variable for a regression analysis might not satisfy one or more assumptions of an ordinary least squares regression. The residuals could either curve as the prediction increases or follow the skewed distribution. In such scenarios, it is necessary to transform the response variable so that the data meets the required assumptions. A Box-Cox transformation is a statistical technique to transform non-normal dependent variables into a normal shape. If the given data is not normal then most of the statistical techniques assume normality. Applying a Box-Cox transformation means that you can run a broader number of tests.
A Box-Cox transformation is a way to transform non-normal dependent variables into a normal shape. Normality is an important assumption for many statistical techniques, if your data isn't normal, applying a Box-Cox means that you are able to run a broader number of tests. The Box-Cox transformation is named after statisticians George Box and Sir David Roxbee Cox who collaborated on a 1964 paper and developed the technique.

Q30. How Regularly Must an Algorithm be Updated?

You will want to update an algorithm when:

- You want the model to evolve as data streams through infrastructure
- The underlying data source is changing
- There is a case of non-stationarity (mean, variance change over the time)
- The algorithm underperforms/results lack accuracy

Q31. If you are having 4GB RAM in your machine and you want to train your model on 10GB data set. How would you go about this problem? Have you ever faced this kind of problem in your machine learning/data science experience so far?

First of all, you have to ask which ML model you want to train.
For Neural networks: Batch size with Numpy array will work. Steps:

1. Load the whole data in the Numpy array. Numpy array has a property to create a mapping of the complete data set, it doesn't load complete data set in memory.
2. You can pass an index to Numpy array to get required data.
3. Use this data to pass to the Neural network.
4. Have a small batch size.

For SVM: Partial fit will work. Steps:

1. Divide one big data set in small size data sets.

2. Use a partial fit method of SVM, it requires a subset of the complete data set.
3. Repeat step 2 for other subsets.

However, you could actually face such an issue in reality. So, you could check out the best laptop for Machine Learning to prevent that. Having said that, let’s move on to some questions on deep learning.