

# Basic Data Science Interview Questions

## 1. What is Data Science?

**Data Science** is a field of computer science that explicitly deals with turning data into information and extracting meaningful insights out of it. The reason why Data Science is so popular is that the kind of insights it allows us to draw from the available data has led to some major innovations in several products and companies. Using these insights, we are able to determine the taste of a particular customer, the likelihood of a product succeeding in a particular market, etc.

## 2. Differentiate between Data Analytics and Data Science

Data Analytics	Data Science
Data Analytics is a subset of Data Science.	Data Science is a broad technology that includes various subsets such as Data Analytics, Data Mining, Data Visualization, etc.
The goal of data analytics is to illustrate the precise details of retrieved insights.	The goal of data science is to discover meaningful insights from massive datasets and derive the best possible solutions to resolve business issues.
Requires just basic programming languages.	Requires knowledge in advanced programming languages.
It focuses on just finding the solutions.	Data Science not only focuses on finding the solutions but also predicts the future with past patterns or insights.
A data analyst's job is to analyse data in order to make decisions.	A data scientist's job is to provide insightful data visualizations from raw data that are easily understandable.

### 3. What do you understand about linear regression?

[Linear regression](#) helps in understanding the linear relationship between the dependent and the independent variables. Linear regression is a supervised learning algorithm, which helps in finding the linear relationship between two variables. One is the predictor or the independent variable and the other is the response or the dependent variable. In Linear Regression, we try to understand how the dependent variable changes w.r.t the independent variable. If there is only one independent variable, then it is called simple linear regression, and if there is more than one independent variable then it is known as multiple linear regression.

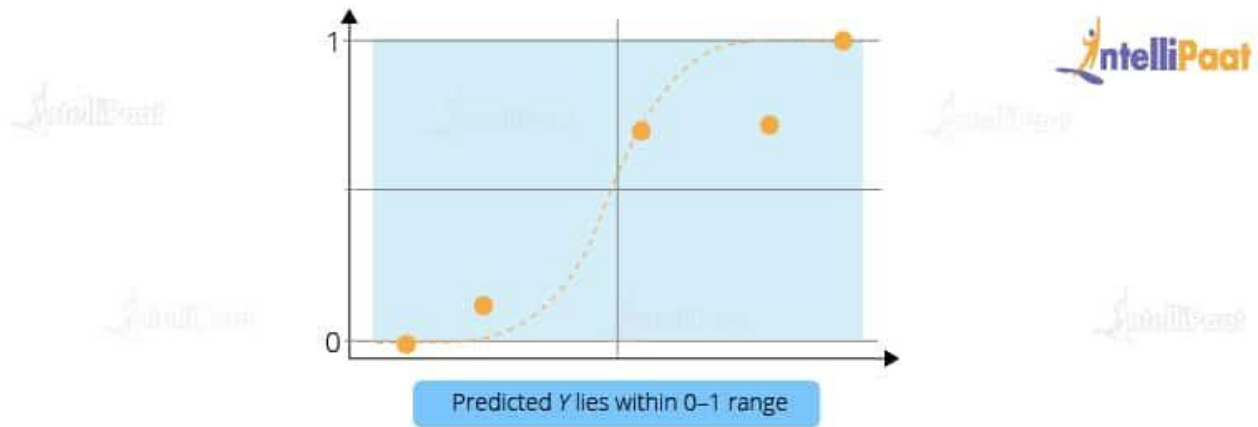
### 4. What do you understand by logistic regression?

[Logistic regression](#) is a classification algorithm that can be used when the dependent variable is binary. Let's take an example. Here, we are trying to determine whether it will rain or not based on temperature and humidity.



Temperature and humidity are the independent variables, and rain would be our dependent variable. So, the logistic regression algorithm actually produces an S shape curve.

Now, let us look at another scenario: Let's suppose that x-axis represents the runs scored by Virat Kohli and the y-axis represents the probability of the team India winning the match. From this graph, we can say that if Virat Kohli scores more than 50 runs, then there is a greater probability for team India to win the match. Similarly, if he scores less than 50 runs then the probability of team India winning the match is less than 50 percent.



So, basically in logistic regression, the Y value lies within the range of 0 and 1. This is how logistic regression works.

## 5. What is a confusion matrix?

The [confusion matrix](#) is a table that is used to estimate the performance of a model. It tabulates the actual values and the predicted values in a 2×2 matrix.

		Predicted	
		Good	Bad
Actual	Good	True Positive (d)	False Negative (c)
	Bad	False Positive (b)	True Negative (a)

True Positive (d): This denotes all of those records where the actual values are *true* and the predicted values are also true. So, these denote all the true positives. False Negative (c): This denotes all those records where the actual values are true, but the predicted values are false. False Positive (b): In this, the actual values are false, but the predicted values are true. True Negative (a): Here, the actual values are *false* and the predicted values are also false. So, if you want to get the correct values, then correct values would basically represent all the true positives and the true negatives. This is how the confusion matrix works.

## **6. What do you understand about the true-positive rate and false-positive rate?**

True positive rate: In Machine Learning, true-positive rates, which are also referred to as sensitivity or recall, are used to measure the percentage of actual positives which are correctly identified. Formula:  $\text{True Positive Rate} = \frac{\text{True Positives}}{\text{Positives}}$  False positive rate: False positive rate is basically the probability of falsely rejecting the null hypothesis for a particular test. The false-positive rate is calculated as the ratio between the number of negative events wrongly categorized as positive (false positive) upon the total number of actual events. Formula:  $\text{False-Positive Rate} = \frac{\text{False-Positives}}{\text{Negatives}}$ .

## **7. How is Data Science different from traditional application programming?**

Data Science takes a fundamentally different approach in building systems that provide value than traditional application development.

In traditional programming paradigms, we used to analyze the input, figure out the expected output, and write code, which contains rules and statements needed to transform the provided input into the expected output. As we can imagine, these rules were not easy to write, especially, for data that even computers had a hard time understanding, e.g., images, videos, etc.

Data Science shifts this process a little bit. In it, we need access to large volumes of data that contain the necessary inputs and their mappings to the expected outputs. Then, we use Data Science algorithms, which use mathematical analysis to generate rules to map the given inputs to outputs.

This process of rule generation is called training. After training, we use some data that was set aside before the training phase to test and check the system's accuracy. The generated rules are a kind of a black box, and we cannot understand how the inputs are being transformed into outputs.

However, If the accuracy is good enough, then we can use the system (also called a model).

As described above, in traditional programming, we had to write the rules to map the input to the output, but in Data Science, the rules are automatically generated or learned from the given data. This helped solve some really difficult challenges that were being faced by several companies.

## 8. Explain the differences between supervised and unsupervised learning.

Supervised and unsupervised learning are two types of Machine Learning techniques. They both allow us to build models. However, they are used for solving different kinds of problems.

Supervised Learning	Unsupervised Learning
Works on the data that contains both inputs and the expected output, i.e., the labeled data	Works on the data that contains no mappings from input to output, i.e., the unlabeled data
Used to create models that can be employed to predict or classify things	Used to extract meaningful information out of large volumes of data
Commonly used supervised learning algorithms: Linear regression, decision tree, etc.	Commonly used unsupervised learning algorithms: K-means clustering, <b>Apriori algorithm</b> , etc.

## 9. What is the difference between the long format data and wide format data?

Long Format Data	Wide Format Data
A long format data has a column for possible variable types and a column for the values of those variables.	Wide data has a column for each variable.
Each row in the long format represents one time point per subject. As a result, each topic will contain many rows of data.	The repeated responses of a subject will be in a single row, with each response in its own column, in the wide format.

This data format is most typically used in R analysis and for writing to log files at the end of each experiment.	This data format is most widely used in data manipulations, stats programs for repeated measures ANOVAs and is seldom used in R analysis.
A long format contains values that do repeat in the first column.	A wide format contains values that do not repeat in the first column.
Use <code>df.melt()</code> to convert wide form to long form	use <code>df.pivot().reset_index()</code> to convert long form into wide form

## 10. Mention some techniques used for sampling. What is the main advantage of sampling?

Sampling is defined as the process of selecting a sample from a group of people or from any particular kind for research purposes. It is one of the most important factors which decides the accuracy of a research/survey result.

**Mainly, there are two types of sampling techniques:**

**Probability sampling:** It involves random selection which makes every element get a chance to be selected. Probability sampling has various subtypes in it, as mentioned below:

- Simple Random Sampling
- Stratified sampling
- Systematic sampling
- Cluster Sampling
- Multi-stage Sampling
- **Non- Probability Sampling:** Non-probability sampling follows non-random selection which means the selection is done based on your ease or any other required criteria. This helps to collect the data easily. The following are various types of sampling in it:
  - Convenience Sampling
  - Purposive Sampling
  - Quota Sampling
  - Referral /Snowball Sampling

## **11. What is bias in Data Science?**

Bias is a type of error that occurs in a Data Science model because of using an algorithm that is not strong enough to capture the underlying patterns or trends that exist in the data. In other words, this error occurs when the data is too complicated for the algorithm to understand, so it ends up building a model that makes simple assumptions. This leads to lower accuracy because of underfitting. Algorithms that can lead to high bias are linear regression, logistic regression, etc.

## **12. What is dimensionality reduction?**

Dimensionality reduction is the process of converting a dataset with a high number of dimensions (fields) to a dataset with a lower number of dimensions. This is done by dropping some fields or columns from the dataset. However, this is not done haphazardly. In this process, the dimensions or fields are dropped only after making sure that the remaining information will still be enough to succinctly describe similar information.

## **13. Why is Python used for Data Cleaning in DS?**

Data Scientists have to clean and transform the huge data sets in a form that they can work with. It is important to deal with the redundant data for better results by removing nonsensical outliers, malformed records, missing values, inconsistent formatting, etc.

Python libraries such as Matplotlib, Pandas, Numpy, Keras, and SciPy are extensively used for [Data cleaning](#) and analysis. These libraries are used to load and clean the data and do effective analysis. For example, a CSV file named "Student" has information about the students of an institute like their names, standard, address, phone number, grades, marks, etc.

## **14. Why is R used in Data Visualization?**

R provides the best ecosystem for data analysis and visualization with more than 12,000 packages in Open-source repositories. It has huge community support, which means you can easily find the solution to your problems on various platforms like StackOverflow.

It has better data management and supports distributed computing by splitting the operations between multiple tasks and nodes, which eventually decreases the complexity and execution time of large datasets.

## **15. What are the popular libraries used in Data Science?**

Below are the popular libraries used for data extraction, cleaning, visualization, and deploying DS models:

- TensorFlow: Supports parallel computing with impeccable library management backed by Google.
- SciPy: Mainly used for solving differential equations, multidimensional programming, data manipulation, and visualization through graphs and charts.
- Pandas: Used to implement the ETL(Extracting, Transforming, and Loading the datasets) capabilities in business applications.
- Matplotlib: Being free and open-source, it can be used as a replacement for MATLAB, which results in better performance and low memory consumption.
- PyTorch: Best for projects which involve Machine Learning algorithms and Deep Neural Networks.

## **16. What is variance in Data Science?**

Variance is a type of error that occurs in a Data Science model when the model ends up being too complex and learns features from data, along with the noise that exists in it. This kind of error could occur if the algorithm used to train the model has high complexity, even though the data and the underlying patterns and trends are quite easy to discover. This makes the model a very sensitive one that performs well on the training dataset but poorly on the testing dataset, and on any kind of data that the model has not yet seen. Variance generally leads to poor accuracy in testing and results in overfitting.

## **17. What is pruning in a decision tree algorithm?**

Pruning a decision tree is the process of removing the sections of the tree that are not necessary or are redundant. Pruning leads to a smaller decision tree, which performs better and gives higher accuracy and speed.



## **18. What is entropy in a decision tree algorithm?**

In a decision tree algorithm, entropy is the measure of impurity or randomness. The entropy of a given dataset tells us how pure or impure the values of the dataset are. In simple terms, it tells us about the variance in the dataset. For example, suppose we are given a box with 10 blue marbles. Then, the entropy of the box is 0 as it contains marbles of the same color, i.e., there is no impurity. If we need to draw a marble from the box, the probability of it being blue will be 1.0. However, if we replace 4 of the blue marbles with 4 red marbles in the box, then the entropy increases to 0.4 for drawing blue marbles.

## **19. What information is gained in a decision tree algorithm?**

When building a decision tree, at each step, we have to create a node that decides which feature we should use to split data, i.e., which feature would best separate our data so that we can make predictions. This decision is made using information gain, which is a measure of how much entropy is reduced when a particular feature is used to split the data. The feature that gives the highest information gain is the one that is chosen to split the data.

## **20. What is k-fold cross-validation?**

In k-fold cross-validation, we divide the dataset into k equal parts. After this, we loop over the entire dataset k times. In each iteration of the loop, one of the k parts is used for testing, and the other k – 1 parts are used for training. Using k-fold cross-validation, each one of the k parts of the dataset ends up being used for training and testing purposes.

## **21. Explain how a recommender system works.**

A recommender system is a system that many consumer-facing, content-driven, online platforms employ to generate recommendations for users from a library of available content. These systems generate recommendations based on what they know about the users' tastes from their activities on the platform.

For example, imagine that we have a movie streaming platform like Netflix or Amazon Prime. If a user has previously watched and liked movies from action and horror genres,

then it means that the user likes watching the movies of these genres. In that case, it would be better to recommend such movies to this user. These recommendations can also be generated based on what users with a similar taste like watching.

## 22. What is a normal distribution?

Data distribution is a visualization tool to analyze how data is spread out or distributed. Data can be distributed in various ways. For instance, it could be with a bias to the left or the right, or it could all be jumbled up.

Data may also be distributed around a central value, i.e., mean, median, etc. This kind of distribution has no bias either to the left or to the right and is in the form of a bell-shaped curve. This distribution also has its mean equal to the median. This kind of distribution is called a [normal distribution](#).

## 23. What is Deep Learning?

Deep Learning is a kind of Machine Learning, in which neural networks are used to imitate the structure of the human brain, and just like how a brain learns from information, machines are also made to learn from the information that is provided to them.

[Deep Learning](#) is an advanced version of neural networks to make the machines learn from data. In Deep Learning, the neural networks comprise many hidden layers (which is why it is called 'deep' learning) that are connected to each other, and the output of the previous layer is the input of the current layer.

## 24. What is an RNN (recurrent neural network)?

A [recurrent neural network](#), or RNN for short, is a kind of Machine Learning algorithm that makes use of the artificial neural network. RNNs are used to find patterns from a sequence of data, such as time series, stock market, temperature, etc. RNNs are a kind of feedforward network, in which information from one layer passes to another layer, and each node in the network performs mathematical operations on the data. These operations are temporal, i.e., RNNs store contextual information about previous computations in the network. It is called recurrent because it performs the same

operations on some data every time it is passed. However, the output may be different based on past computations and their results.

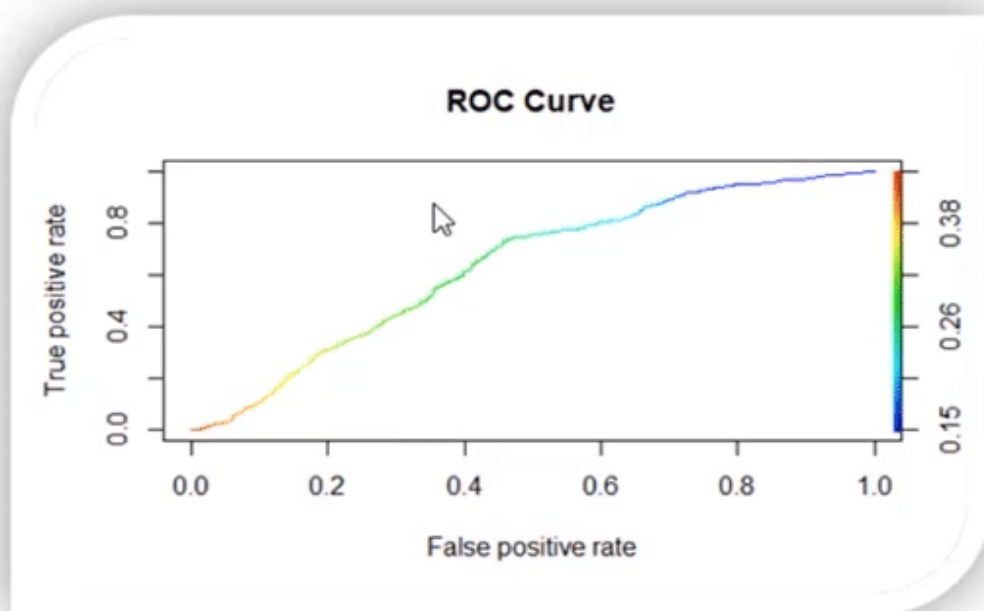
## 25. Explain selection bias.

Selection bias is the bias that occurs during the sampling of data. This kind of bias occurs when a sample is not representative of the population, which is going to be analyzed in a statistical study.

## Intermediate Data Science Interview Questions

## 26. What is the ROC curve?

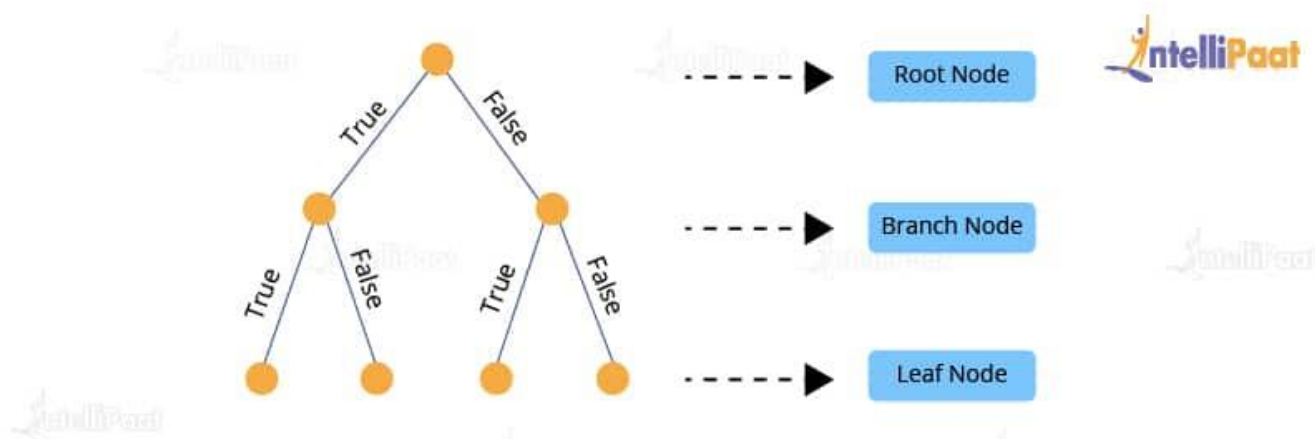
It stands for [Receiver Operating Characteristic](#). It is basically a plot between a true positive rate and a false positive rate, and it helps us to find out the right tradeoff between the true positive rate and the false positive rate for different probability thresholds of the predicted values. So, the closer the curve to the upper left corner, the better the model is.



In other words, whichever curve has greater area under it that would be the better model. You can see this in the below graph:

## 27. What do you understand by a decision tree?

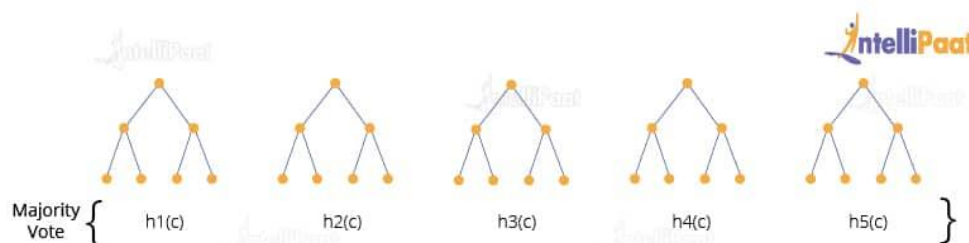
A [decision tree](#) is a supervised learning algorithm that is used for both classification and regression. Hence, in this case, the dependent variable can be both a numerical value and a categorical value.



Here, each node denotes the test on an attribute, and each edge denotes the outcome of that attribute, and each leaf node holds the class label. So, in this case, we have a series of test conditions which give the final decision according to the condition.

## 28. What do you understand by a random forest model?

It combines multiple models together to get the final output or, to be more precise, it combines multiple decision trees together to get the final output. So, decision trees are the building blocks of the random forest model.



**29. Two candidates, Aman and Mohan appear for a Data Science Job interview. The probability of Aman cracking the interview is  $\frac{1}{8}$  and that of Mohan is  $\frac{5}{12}$ . What is the probability that at least one of them will crack the interview?**

The probability of Aman getting selected for the interview is  $\frac{1}{8}$

$$P(A) = \frac{1}{8}$$

The probability of Mohan getting selected for the interview is  $\frac{5}{12}$

$$P(B) = \frac{5}{12}$$

Now, the probability of at least one of them getting selected can be denoted as the Union of A and B, which means

$$P(A \cup B) = P(A) + P(B) - (P(A \cap B)) \dots\dots\dots(1)$$

Where  $P(A \cap B)$  stands for the probability of both Aman and Mohan getting selected for the job.

To calculate the final answer, we first have to find out the value of  $P(A \cap B)$

$$\text{So, } P(A \cap B) = P(A) * P(B)$$

$$\frac{1}{8} * \frac{5}{12}$$

$$\frac{5}{96}$$

Now, put the value of  $P(A \cap B)$  into equation (1)

$$P(A \cup B) = P(A) + P(B) - (P(A \cap B))$$

$$\frac{1}{8} + \frac{5}{12} - \frac{5}{96}$$

So, the answer will be  $\frac{47}{96}$ .

### 30. How is Data modeling different from Database design?

**Data Modeling:** It can be considered as the first step towards the design of a database. Data modeling creates a conceptual model based on the relationship between various data models. The process involves moving from the conceptual stage to the logical model to the physical schema. It involves the systematic method of applying data modeling techniques.

**Database Design:** This is the process of designing the database. The database design creates an output which is a detailed data model of the database. Strictly speaking, database design includes the detailed logical model of a database but it can also include physical design choices and storage parameters.

### 31. What is precision?

**Precision:** When we are implementing algorithms for the classification of data or the retrieval of information, precision helps us get a portion of positive class values that are positively predicted. Basically, it measures the accuracy of correct positive predictions. Below is the formula to calculate precision:

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

### 32. What is a recall?

**Recall:** It is the set of all positive predictions out of the total number of positive instances. Recall helps us identify the misclassified positive predictions. We use the below formula to calculate recall:

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

### 33. What is the F1 score and how to calculate it?

F1 score helps us calculate the harmonic mean of precision and recall that gives us the test's accuracy. If  $F1 = 1$ , then precision and recall are accurate. If  $F1 < 1$  or equal to 0, then precision or recall is less accurate, or they are completely inaccurate. See below for the formula to calculate the F1 score:

$$F1\text{-score} \triangleq 2 \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

### 34. What is a p-value?

P-value is the measure of the statistical importance of an observation. It is the probability that shows the significance of output to the data. We compute the p-value to know the test statistics of a model. Typically, it helps us choose whether we can accept or reject the null hypothesis.

### 35. Why do we use p-value?

We use the p-value to understand whether the given data really describes the observed effect or not. We use the below formula to calculate the p-value for the effect 'E' and the null hypothesis 'H0' is true:

$$P \text{ Value} = P(E \mid H0)$$

### 36. What is the difference between an error and a residual error?

An error occurs in values while the prediction gives us the difference between the observed values and the true values of a dataset. Whereas the residual error is the difference between the observed values and the predicted values. The reason we use the residual error to evaluate the performance of an algorithm is that the true values are never

known. Hence, we use the observed values to measure the error using residuals. It helps us get an accurate estimate of the error.

### 37. Why do we use the summary function?

The summary function in R gives us the statistics of the implemented algorithm on a particular dataset. It consists of various objects, variables, data attributes, etc. It provides summary statistics for individual objects when fed into the function. We use a summary

```
TotalCharges
Min.      : 18.8
1st Qu.   : 401.4
Median    :1397.5
Mean      :2283.3
3rd Qu.   :3794.7
Max.      :8684.8
NA's      :11
```

function when we want information about the values present in the dataset. It gives us the summary statistics in the following form: Here, it gives the minimum and maximum values from a specific column of the dataset. Also, it provides the median, mean, 1st quartile, and 3rd quartile values that help us understand the values better.

### 38. How are Data Science and Machine Learning related to each other?

Data Science and [Machine Learning](#) are two terms that are closely related but are often misunderstood. Both deal with data. However, there are some fundamental distinctions that show us how they are different from each other.

Data Science is a broad field that deals with large volumes of data and allows us to draw insights out of this voluminous data. The entire process of Data Science takes care of multiple steps that are involved in drawing insights out of the available data. This process includes crucial steps such as data gathering, data analysis, data manipulation, data visualization, etc.



Machine Learning, on the other hand, can be thought of as a sub-field of Data Science. It also deals with data, but here, we are solely focused on learning how to convert the processed data into a functional model, which can be used to map inputs to outputs, e.g., a model that can expect an image as an input and tell us if that image contains a flower as an output.

In short, Data Science deals with gathering data, processing it, and finally, drawing insights from it. The field of Data Science that deals with building models using algorithms is called Machine Learning. Therefore, Machine Learning is an integral part of Data Science.

### **39. Explain univariate, bivariate, and multivariate analyses.**

When we are dealing with data analysis, we often come across terms such as univariate, bivariate, and multivariate. Let's try and understand what this means.

- **Univariate analysis:** Univariate analysis involves analyzing data with only one variable or, in other words, a single column or a vector of the data. This analysis allows us to understand the data and extract patterns and trends out of it. Example: Analyzing the weight of a group of people.
- **Bivariate analysis:** Bivariate analysis involves analyzing the data with exactly two variables or, in other words, the data can be put into a two-column table. This kind of analysis allows us to figure out the relationship between the variables. Example: Analyzing the data that contains temperature and altitude.
- **Multivariate analysis:** Multivariate analysis involves analyzing the data with more than two variables. The number of columns of the data can be anything more than two. This kind of analysis allows us to figure out the effects of all other variables (input variables) on a single variable (the output variable).

Example: Analyzing data about house prices, which contains information about the houses, such as locality, crime rate, area, the number of floors, etc.

### **40. How can we handle missing data?**

To be able to handle missing data, we first need to know the percentage of data missing in a particular column so that we can choose an appropriate strategy to handle the situation.

For example, if in a column majority of the data is missing, then dropping the column is the best option, unless we have some means to make educated guesses about the missing values. However, if the amount of missing data is low, then we have several strategies to fill them up.

One way would be to fill them all up with a default value or a value that has the highest frequency in that column, such as 0 or 1, etc. This may be useful if most of the data in that column contains these values.

Another way is to fill up the missing values in the column with the mean of all the values in that column. This technique is usually preferred as the missing values have a higher chance of being closer to the mean than to the mode.

Finally, if we have a huge dataset and a few rows have values missing in some columns, then the easiest and fastest way is to drop those columns. Since the dataset is large, dropping a few columns should not be a problem anyway.

## **41. What is the benefit of dimensionality reduction?**

Dimensionality reduction reduces the dimensions and size of the entire dataset. It drops unnecessary features while retaining the overall information in the data intact. Reduction in dimensions leads to faster processing of the data.

The reason why data with high dimensions is considered so difficult to deal with is that it leads to high time consumption while processing the data and training a model on it. Reducing dimensions speeds up this process, removes noise, and leads to better model accuracy.

## **42. What is a bias-variance trade-off in Data Science?**

When building a model using Data Science or Machine Learning, our goal is to build one that has low bias and variance. We know that bias and variance are both errors that occur due to either an overly simplistic model or an overly complicated model. Therefore, when we are building a model, the goal of getting high accuracy is only going to be accomplished if we are aware of the tradeoff between bias and variance. Bias is an error that occurs when a model is too simple to capture the patterns in a dataset.

To reduce bias, we need to make our model more complex. Although making the model more complex can lead to reducing bias, and if we make the model too complex, it may end up becoming too rigid, leading to high variance. So, the tradeoff between bias and variance is that if we increase the complexity, the bias reduces and the variance increases, and if we reduce complexity, the bias increases and the variance reduces. Our goal is to find a point at which our model is complex enough to give low bias but not so complex to end up having high variance.

### **43. What is RMSE?**

RMSE stands for the root mean square error. It is a measure of accuracy in regression. RMSE allows us to calculate the magnitude of error produced by a regression model. The way RMSE is calculated is as follows:

First, we calculate the errors in the predictions made by the regression model. For this, we calculate the differences between the actual and the predicted values. Then, we square the errors.

After this step, we calculate the mean of the squared errors, and finally, we take the square root of the mean of these squared errors. This number is the RMSE, and a model with a lower value of RMSE is considered to produce lower errors, i.e., the model will be more accurate.

### **44. What is a kernel function in SVM?**

In the [SVM algorithm](#), a kernel function is a special mathematical function. In simple terms, a kernel function takes data as input and converts it into a required form. This transformation of the data is based on something called a kernel trick, which is what gives the kernel function its name. Using the kernel function, we can transform the data that is not linearly separable (cannot be separated using a straight line) into one that is linearly separable.

## 45. How can we select an appropriate value of $k$ in k-means?

Selecting the correct value of  $k$  is an important aspect of k-means clustering. We can make use of the elbow method to pick the appropriate  $k$  value. To do this, we run the k-means algorithm on a range of values, e.g., 1 to 15. For each value of  $k$ , we compute an average score. This score is also called inertia or the inter-cluster variance.

This is calculated as the sum of squares of the distances of all values in a cluster. As  $k$  starts from a low value and goes up to a high value, we start seeing a sharp decrease in the inertia value. After a certain value of  $k$ , in the range, the drop in the inertia value becomes quite small. This is the value of  $k$  that we need to choose for the k-means clustering algorithm.

## 46. How can we deal with outliers?

Outliers can be dealt with in several ways. One way is to drop them. We can only drop the outliers if they have values that are incorrect or extreme. For example, if a dataset with the weights of babies has a value 98.6-degree Fahrenheit, then it is incorrect. Now, if the value is 187 kg, then it is an extreme value, which is not useful for our model.

**In case the outliers are not that extreme, then we can try:**

- A different kind of model. For example, if we were using a linear model, then we can choose a non-linear model
- Normalizing the data, which will shift the extreme values closer to other data points
- Using algorithms that are not so affected by outliers, such as random forest, etc.

## 47. How to calculate the accuracy of a binary classification algorithm using its confusion matrix?

In a binary classification algorithm, we have only two labels, which are True and False. Before we can calculate the accuracy, we need to understand a few key terms:

- True positives: Number of observations correctly classified as True
- True negatives: Number of observations correctly classified as False
- False positives: Number of observations incorrectly classified as True

- False negatives: Number of observations incorrectly classified as False

To calculate the accuracy, we need to divide the sum of the correctly classified observations by the number of total observations. This can be expressed as follows:

## **48. What is ensemble learning?**

When we are building models using Data Science and Machine Learning, our goal is to get a model that can understand the underlying trends in the training data and can make predictions or classifications with a high level of accuracy.

However, sometimes some datasets are very complex, and it is difficult for one model to be able to grasp the underlying trends in these datasets. In such situations, we combine several individual models together to improve performance. This is what is called ensemble learning.

## **49. Explain collaborative filtering in recommender systems.**

Collaborative filtering is a technique used to build recommender systems. In this technique, to generate recommendations, we make use of data about the likes and dislikes of users similar to other users. This similarity is estimated based on several varying factors, such as age, gender, locality, etc.

If User A, like User B, watched and liked a movie, then that movie will be recommended to User B, and similarly, if User B watched and liked a movie, then that would be recommended to User A.

In other words, the content of the movie does not matter much. When recommending it to a user what matters is if other users similar to that particular user liked the content of the movie or not.

## **50. Explain content-based filtering in recommender systems.**

Content-based filtering is one of the techniques used to build recommender systems. In this technique, recommendations are generated by making use of the properties of the content that a user is interested in.

For example, if a user is watching movies belonging to the action and mystery genre and giving them good ratings, it is a clear indication that the user likes movies of this kind. If shown movies of a similar genre as recommendations, there is a higher probability that the user would like those recommendations as well.

In other words, here, the content of the movie is taken into consideration when generating recommendations for users.

## **51. Explain bagging in Data Science.**

Bagging is an ensemble learning method. It stands for bootstrap aggregating. In this technique, we generate some data using the bootstrap method, in which we use an already existing dataset and generate multiple samples of the  $N$  size. This bootstrapped data is then used to train multiple models in parallel, which makes the bagging model more robust than a simple model.

Once all the models are trained, when it's time to make a prediction, we make predictions using all the trained models and then average the result in the case of regression, and for classification, we choose the result, generated by models, that have the highest frequency.

## **52. Explain boosting in Data Science.**

Boosting is one of the ensemble learning methods. Unlike bagging, it is not a technique used to parallelly train our models. In boosting, we create multiple models and sequentially train them by combining weak models iteratively in a way that training a new model depends on the models trained before it.

In doing so, we take the patterns learned by a previous model and test them on a dataset when training the new model. In each iteration, we give more importance to observations in the dataset that are incorrectly handled or predicted by previous models. Boosting is useful in reducing bias in models as well.

### **53. Explain stacking in Data Science.**

Just like bagging and boosting, stacking is also an ensemble learning method. In bagging and boosting, we could only combine weak models that used the same learning algorithms, e.g., logistic regression. These models are called homogeneous learners.

However, in stacking, we can combine weak models that use different learning algorithms as well. These learners are called heterogeneous learners. Stacking works by training multiple (and different) weak models or learners and then using them together by training another model, called a meta-model, to make predictions based on the multiple outputs of predictions returned by these multiple weak models.

### **54. Explain how Machine Learning is different from Deep Learning.**

A field of computer science, Machine Learning is a subfield of Data Science that deals with using existing data to help systems automatically learn new skills to perform different tasks without having rules to be explicitly programmed.

Deep Learning, on the other hand, is a field in Machine Learning that deals with building Machine Learning models using algorithms that try to imitate the process of how the human brain learns from the information in a system for it to attain new capabilities. In Deep Learning, we make heavy use of deeply connected neural networks with many layers.

### **55. What does the word 'Naive' mean in Naive Bayes?**

Naive Bayes is a Data Science algorithm. It has the word 'Bayes' in it because it is based on the Bayes theorem, which deals with the probability of an event occurring given that another event has already occurred.

It has 'naive' in it because it assumes that each variable in the dataset is independent of the other. This kind of assumption is unrealistic for real-world data. However, even with this assumption, it is very useful for solving a range of complicated problems, e.g., spam email classification, etc.

## Advanced Data Science Interview Questions

**56. From the below given 'diamonds' dataset, extract only those rows where the 'price' value is greater than 1000 and the 'cut' is ideal.**

carat	cut	color	clarity	depth	table	price	x	y	z
0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
0.24	Very Good	J	VVS2	62.8	57.0	336	3.94	3.96	2.48
0.24	Very Good	I	VVS1	62.3	57.0	336	3.95	3.98	2.47
0.26	Very Good	H	SI1	61.9	55.0	337	4.07	4.11	2.53
0.22	Fair	E	VS2	65.1	61.0	337	3.87	3.78	2.49

First, we will load the **ggplot2** package:

```
library(ggplot2)
```

Next, we will use the **dplyr** package:

```
library(dplyr)// It is based on the grammar of data manipulation.
```

To extract those particular records, use the below command:

```
diamonds %>% filter(price>1000 & cut=="Ideal")-> diamonds_1000_idea
```

**57. Make a scatter plot between 'price' and 'carat' using ggplot. 'Price' should be on the y-axis, 'carat' should be on the x-axis, and the 'color' of the points should be determined by 'cut.'**

We will implement the scatter plot using **ggplot**.



The ggplot is based on the grammar of data visualization, and it helps us stack multiple layers on top of each other.

So, we will start with the data layer, and on top of the data layer we will stack the aesthetic layer. Finally, on top of the aesthetic layer we will stack the geometry layer.

**Code:**

```
>ggplot(data=diamonds, aes(x=caret, y=price, col=cut))+geom_point()
```

## 58. Introduce 25 percent missing values in this 'iris' dataset and impute the 'Sepal.Length' column with 'mean' and the 'Petal.Length' column with 'median.'

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa

To introduce missing values, we will be using the **missForest** package:

```
library(missForest)
```

Using the prodNA function, we will be introducing 25 percent of missing values:

```
Iris.mis<-prodNA(iris,noNA=0.25)
```

For imputing the 'Sepal.Length' column with 'mean' and the 'Petal.Length' column with 'median,' we will be using the Hmisc package and the impute function:

```
library(Hmisc)
iris.mis$Sepal.Length<-with(iris.mis, impute(Sepal.Length,mean))
iris.mis$Petal.Length<-with(iris.mis, impute(Petal.Length,median))
```

## 59. Implement simple linear regression in R on this 'mtcars' dataset, where the dependent variable is 'mpg' and the independent variable is 'disp.'

mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4

Here, we need to find how 'mpg' varies w.r.t displacement of the column.

We need to divide this data into the training dataset and the testing dataset so that the model does not overfit the data.

So, what happens is when we do not divide the dataset into these two components, it overfits the dataset. Hence, when we add new data, it fails miserably on that new data.

Therefore, to divide this dataset, we would require the **caret** package. This caret package comprises the **createDataPartition()** function. This function will give the true or false labels.

Here, we will use the following code:

```
library(caret)

split_tag<-createDataPartition(mtcars$mpg, p=0.65, list=F)

mtcars[split_tag,]->train

mtcars[-split_tag,]->test
```

```
lm(mpg~data,data=train)->mod_mtcars

predict(mod_mtcars,newdata=test)->pred_mtcars

>head(pred_mtcars)
```

### Explanation:

**Parameters of the createDataPartition function:** First is the column which determines the split (it is the mpg column).

Second is the split ratio which is 0.65, i.e., 65 percent of records will have true labels and 35 percent will have false labels. We will store this in a split\_tag object.

Once we have **split\_tag** object ready, from this entire **mtcars dataframe**, we will select all those records where the split tag value is **true** and store those records in the **training** set.

Similarly, from the mtcars dataframe, we will select all those records where the split\_tag value is **false** and store those records in the **test** set.

So, the split tag will have true values in it, and when we put '-' symbol in front of it, '-split\_tag' will contain all of the false labels. We will select all those records and store them in the test set.

We will go ahead and build a model on top of the training set, and for the simple linear model we will require the **lm function**.

```
lm(mpg~data,data=train)->mod_mtcars
```

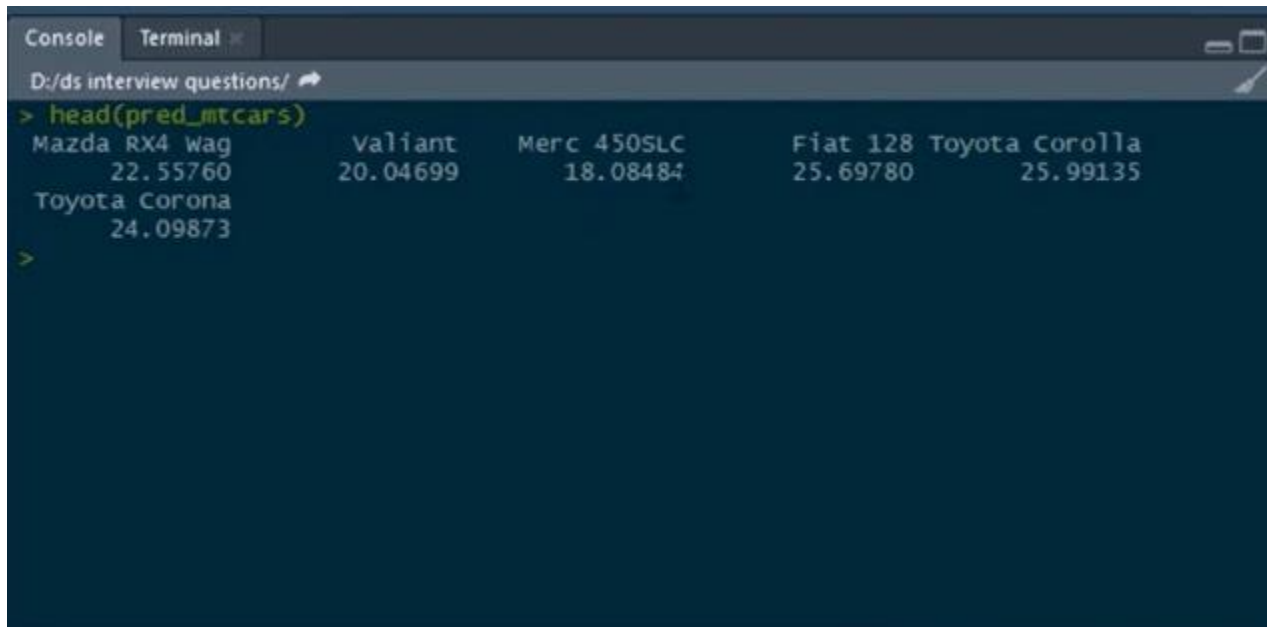
Now, we have built the model on top of the train set. It's time to predict the values on top of the test set. For that, we will use the **predict** function that takes in two parameters: first is the model which we have built and second is the dataframe on which we have to predict values.

Thus, we have to predict values for the test set and then store them in pred\_mtcars.

```
predict(mod_mtcars,newdata=test)->pred_mtcars
```

## Output:

These are the predicted values of mpg for all these cars.



```
Console Terminal
D:/ds interview questions/
> head(pred_mtcars)
Mazda RX4 Wag      Valiant      Merc 450SLC      Fiat 128 Toyota Corolla
  22.55760      20.04699      18.08484      25.69780      25.99135
Toyota Corona
  24.09873
>
```

So, this is how we can build a simple linear model on top of this mtcars dataset.

## 60. Calculate the RMSE values for the model building.

When we build a regression model, it predicts certain  $y$  values associated with the given  $x$  values, but there is always an error associated with this prediction. So, to get an estimate of the average error in prediction, RMSE is used. **Code:**

```
cbind(Actual=test$mpg, predicted=pred_mtcars)->final_data

as.data.frame(final_data)->final_data

error<-(final_data$Actual-final_data$Prediction)

cbind(final_data,error)->final_data

sqrt(mean(final_data$error)^2)
```

**Explanation:** We have the actual and the predicted values. We will bind both into a single dataframe. For that, we will use the **cbind** function:

```
cbind(Actual=test$mpg, predicted=pred_mtcars)->final_data
```

Our actual values are present in the **mpg** column from the test set, and our predicted values are stored in the **pred\_mtcars** object which we have created in the previous question. Hence, we will create this new column and name the column **actual**. Similarly, we will create another column and name it **predicted** which will have predicted values and then store the predicted values in the new object which is **final\_data**. After that, we will convert a matrix into a dataframe. So, we will use the **as.data.frame** function and convert this object (predicted values) into a dataframe:

```
as.data.frame(final_data)->final_data
```

We will pass this object which is **final\_data** and store the result in **final\_data** again. We will then calculate the error in prediction for each of the records by subtracting the predicted values from the actual values:

```
error<-(final_data$Actual-final_data$Prediction)
```

Then, store this result on a new object and name that object as **error**. After this, we will bind this error calculated to the same **final\_data** dataframe:

```
cbind(final_data,error)->final_data //binding error object to this  
final_data
```

Here, we bind the error object to this **final\_data**, and store this into **final\_data** again. **Calculating RMSE:**

```
Sqrt(mean(final_data$error)^2)
```

**Output:**

```
[1] 4.334423
```

**Note:** Lower the value of RMSE, the better the model.

## 61. Implement simple linear regression in Python on this 'Boston' dataset where the dependent variable is 'medv' and the independent variable is 'lstat.'

### Simple Linear Regression

```
import pandas as pd

data=pd.read_csv('Boston.csv')    //loading the Boston dataset

data.head()  //having a glance at the head of this data

data.shape
```

Let us take out the dependent and the independent variables from the dataset:

```
data1=data.loc[:,['lstat','medv']]

data1.head()
```

### Visualizing Variables

```
import matplotlib.pyplot as plt

data1.plot(x='lstat',y='medv',style='o')

plt.xlabel('lstat')

plt.ylabel('medv')

plt.show()
```

Here, 'medv' is basically the median values of the price of the houses, and we are trying to find out the median values of the price of the houses w.r.t to the lstat column.

We will separate the dependent and the independent variable from this entire dataframe:

```
data1=data.loc[:,['lstat','medv']]
```

The only columns we want from all of this record are 'lstat' and 'medv,' and we need to store these results in data1.

Now, we would also do a visualization w.r.t to these two columns:

```
import matplotlib.pyplot as plt

data1.plot(x='lstat',y='medv',style='o')

plt.xlabel('lstat')

plt.ylabel('medv')

plt.show()
```

### Preparing the Data

```
X=pd.DataFrame(data1['lstat'])

Y=pd.DataFrame(data1['medv'])

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=100)

from sklearn.linear_model import LinearRegression

regressor=LinearRegression()

regressor.fit(X_train,y_train)

print(regressor.intercept_)
```

Output:

```
34.12654201
```

```
print(regressor.coef_)//this is the slope
```

Output:

```
[[-0.913293]]
```

By now, we have built the model. Now, we have to predict the values on top of the test set:

```
y_pred=regressor.predict(X_test)//using the instance and the predict  
function and pass the X_test object inside the function and store this in  
y_pred object
```

Now, let's have a glance at the rows and columns of the actual values and the predicted values:

```
Y_pred.shape, y_test.shape
```

Output:

```
((102,1),(102,1))
```

Further, we will go ahead and calculate some metrics so that we can find out the Mean Absolute Error, Mean Squared Error, and RMSE.

```
from sklearn import metrics import NumPy as np  
  
print('Mean Absolute Error: ', metrics.mean_absolute_error(y_test, y_pred))  
  
print('Mean Squared Error: ', metrics.mean_squared_error(y_test, y_pred))  
  
print('Root Mean Squared Error: ',  
np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

Output:

```
Mean Absolute Error: 4.692198
```

```
Mean Squared Error: 43.9198
```

```
Root Mean Squared Error: 6.6270
```



## 62. Implement logistic regression on this 'heart' dataset in R where the dependent variable is 'target' and the independent variable is 'age.'

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
56	0	1	140	294	0	0	153	0	1.3	1	0	2	1
44	1	1	120	263	0	1	173	0	0.0	2	0	3	1
52	1	2	172	199	1	1	162	0	0.5	2	0	3	1
57	1	2	150	168	0	1	174	0	1.6	2	0	2	1

For loading the dataset, we will use the **read.csv** function:

```
read.csv("D:/heart.csv") -> heart  
  
str(heart)
```

In the structure of this dataframe, most of the values are integers. However, since we are building a logistic regression model on top of this dataset, the final **target column is supposed to be categorical**. It cannot be an integer. So, we will go ahead and convert them into a factor. Thus, we will use the **as.factor** function and convert these integer values into categorical data.

We will pass on **heart\$target** column over here and store the result in **heart\$target** as follows:

```
as.factor(heart$target) -> heart$target
```

Now, we will build a logistic regression model and see the different probability values for the person to have heart disease based on different age values.

To build a logistic regression model, we will use the **glm** function:

```
glm(target~age, data=heart, family="binomial") -> log_mod1
```

Here, **target~age** indicates that the target is the dependent variable, and the age is the independent variable, and we are building this model on top of the dataframe.

**family="binomial"** means we are basically telling R that this is the logistic regression model, and we will store the result in **log\_mod1**.

We will have a glance at the summary of the model that we have just built:

```
summary(log_mod1)
```

```
> summary(log_mod1)

Call:
glm(formula = target ~ age, family = "binomial", data = heart)

Deviance Residuals:
    Min       1Q   Median       3Q      Max 
-1.7125 -1.1773  0.8296  1.0685  1.5947 

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  3.03623    0.75639   4.014 5.97e-05 ***
age         -0.05235    0.01363  -3.841 0.000122 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 417.64  on 302  degrees of freedom
Residual deviance: 401.86  on 301  degrees of freedom
AIC: 405.86

Number of Fisher Scoring iterations: 4
```

We can see **Pr** value here, and there are three stars associated with this Pr value. This basically means that we can reject the null hypothesis which states that there is no relationship between the age and the target columns. But since we have three stars over here, this null hypothesis can be rejected. There is a strong relationship between the age column and the target column.

Now, we have other parameters like null deviance and residual deviance. Lower the deviance value, the better the model.

This null deviance basically tells the deviance of the model, i.e., when we don't have any independent variable and we are trying to predict the value of the target column with only the intercept. When that's the case, the null deviance is 417.64.

Residual deviance is wherein we include the independent variables and try to predict the target columns. Hence, when we include the independent variable which is age, we see that the residual deviance drops. Initially, when there are no independent variables, the null deviance was 417. After we include the age column, we see that the null deviance is reduced to 401.

This basically means that there is a strong relationship between the age column and the target column and that is why the deviance is reduced.

As we have built the model, it's time to predict some values:

```
predict(log_mod1, data.frame(age=30), type="response")

predict(log_mod1, data.frame(age=50), type="response")

predict(log_mod1, data.frame(age=29:77), type="response")
```

Now, we will divide this dataset into train and test sets and build a model on top of the train set and predict the values on top of the test set:

```
>library(caret)

Split_tag<- createDataPartition(heart$target, p=0.70, list=F)

heart[split_tag,]->train

heart[-split_tag,]->test

glm(target~age, data=train,family="binomial")->log_mod2

predict(log_mod2, newdata=test, type="response")->pred_heart

range(pred_heart)
```

## 63. Build an ROC curve for the model built

The below code will help us in building the ROC curve:

```
library(ROCR)

prediction(pred_heart, test$target)-> roc_pred_heart

performance(roc_pred_heart, "tpr", "fpr")->roc_curve

plot(roc_curve, colorize=T)
```

## 64. Build a confusion matrix for the model where the threshold value for the probability of predicted values is 0.6, and also find the accuracy of the model.

Accuracy is calculated as:

**Accuracy = (True positives + true negatives)/(True positives+ true negatives + false positives + false negatives)**

To build a confusion matrix in R, we will use the table function:

```
table(test$target,pred_heart>0.6)
```

Here, we are setting the probability threshold as 0.6. So, wherever the probability of pred\_heart is greater than 0.6, it will be classified as 0, and wherever it is less than 0.6 it will be classified as 1.

Then, we calculate the accuracy by the formula for calculating **Accuracy**.

```
Console Terminal x
D:/ds interview questions/ ↵
> table(test$target, pred_heart>0.6)

  FALSE TRUE
0     31   10
1     32   17
> (31+17)/(31+17+10+32)
[1] 0.5333333
>
```

**65. Build a logistic regression model on the 'customer\_churn' dataset in Python. The dependent variable is 'Churn' and the independent variable is 'MonthlyCharges.' Find the log\_loss of the model.**

First, we will load the pandas dataframe and the customer\_churn.csv file:

```
customer_churn=pd.read_csv("customer_churn.csv")
```

customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService
7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL
5575-GNVDE	Male	0	No	No	34	Yes	No	DSL
3668-QPYBK	Male	0	No	No	2	Yes	No	DSL
7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL
9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic
9305-CDSKC	Female	0	No	No	8	Yes	Yes	Fiber optic
1452-KIOVK	Male	0	No	Yes	22	Yes	Yes	Fiber optic
6713-OKOMC	Female	0	No	No	10	No	No phone service	DSL
7892-POOKP	Female	0	Yes	No	28	Yes	Yes	Fiber optic

After loading this dataset, we can have a glance at the head of the dataset by using the following command:

```
customer_churn.head()
```

Now, we will separate the dependent and the independent variables into two separate objects:

```
x=pd.DataFrame(customer_churn['MonthlyCharges'])

y=customer_churn['Churn']

#Splitting the data into training and testing sets

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test=train_test_split(x,y,test_size=0.3,
random_state=0)
```

Now, we will see how to build the model and calculate **log\_loss**.

```
from sklearn.linear_model, we have to import LogisticRegression

l=LogisticRegression()

l.fit(x_train,y_train)

y_pred=l.predict_proba(x_test)
```

As we are supposed to calculate the log\_loss, we will import it from **sklearn.metrics**:

```
from sklearn.metrics import log_loss

print(log_loss(y_test,y_pred)//actual values are in y_test and predicted
are in y_pred)
```

**Output:**

```
0.5555020595194167
```

**66. Build a decision tree model on 'Iris' dataset where the dependent variable is 'Species,' and all other columns are independent variables. Find the accuracy of the model built.**

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa

To build a decision tree model, we will be loading the **party** package:

```
#party package

library(party)

#splitting the data

library(caret)

split_tag<-createDataPartition(iris$Species, p=0.65, list=F)

iris[split_tag,]->train

iris[~split_tag,]->test

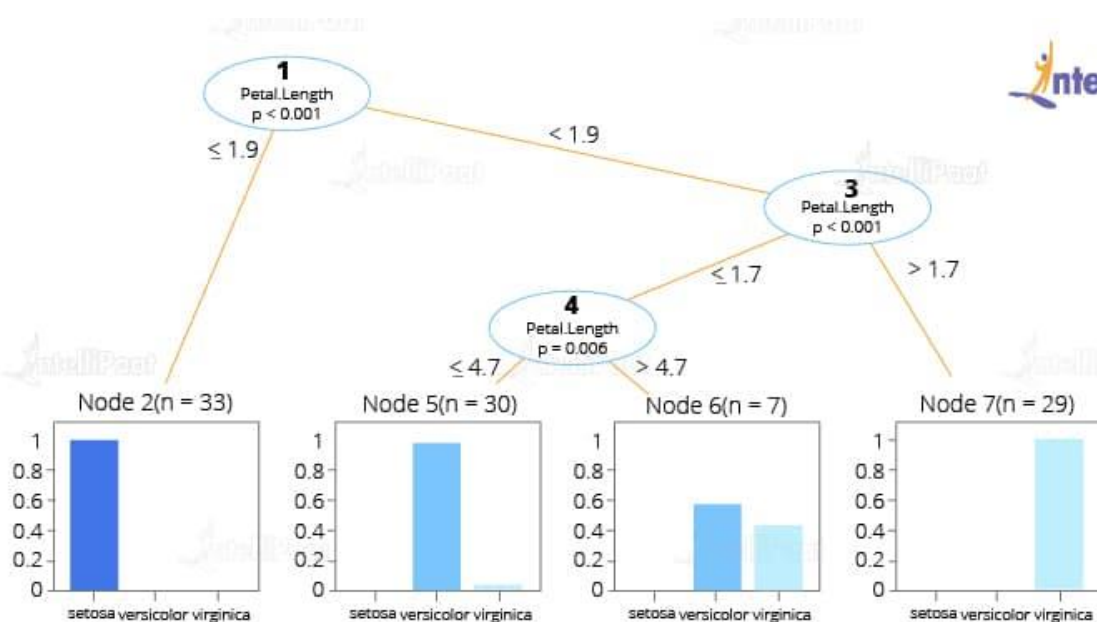
#building model

mytree<-ctree(Species~.,train)
```

Now we will plot the model

```
plot(mytree)
```

**Model:**



#predicting the values

```
predict(mytree, test, type='response') -> mypred
```

After this, we will predict the confusion matrix and then calculate the accuracy using the table function:



```
Console Terminal x
D:/ds interview questions/ ↗
> table(test$Species, mypred)
      mypred
      setosa versicolor virginica
setosa      17         0         0
versicolor  0        16         1
virginica   0         1        16
> (17+16+16)/(17+16+16+1+1)
[1] 0.9607843
>
```

```
table(test$Species, mypred)
```

67. Build a random forest model on top of this 'CTG' dataset, where 'NSP' is the dependent variable, and all other columns are independent variables.

Nmax	Nzeros	Mode	Mean	Median	Variance	Tendency	NSP
2	0	120	137	121	73	1 2	
6	1	141	136	140	12	0 1	
5	1	141	135	138	13	0 1	
11	0	137	134	137	13	1 1	
9	0	137	136	138	11	1 1	
5	3	76	107	107	170	0 3	
6	3	71	107	106	215	0 3	
0	0	122	122	123	3	1 3	
0	0	122	122	123	3	1 3	
1	0	122	122	123	1	1 3	
2	0	150	148	151	9	1 2	

We will load the CTG dataset by using **read.csv**:

```
data<-read.csv("C:/Users/intellipa/Downloads/CTG.csv",header=True)
```

```
str(data)
```

Converting the integer type to a factor

```
data$NSP<-as.factor(data$NSP)

table(data$NSP)

#data partition

set.seed(123)

split_tag<-createDataPartition(data$NSP, p=0.65, list=F)

data[split_tag,]->train

data[~split_tag,]->test

#random forest -1

library(randomForest)

set.seed(222)

rf<-randomForest(NSP~.,data=train)

rf

#prediction

predict(rf,test)->p1
```

Building confusion matrix and calculating accuracy:

```
table(test$NSP,p1)
```

```
Console Terminal
D:/ds interview questions/
> table(test$NSP,p1)
p1
  1  2  3
1 567  8  4
2  20 81  2
3   5  5 51
> (567+81+51)/(567+81+51+2+3+8+4+2+5+5)
[1] 0.9601648
>
```

## 68. Write a function to calculate the Euclidean distance between two points.

The formula for calculating the Euclidean distance between two points (x1, y1) and (x2, y2) is as follows:

$$\sqrt{((x1 - x2)^2 + ((y1 - y2)^2))}$$

Code for calculating the Euclidean distance is as given below:

```
def euclidean_distance(P1, P2):
    return (((P1[0] - P2[0]) ** 2) + ((P1[1] - P2[1]) ** 2)) ** .5
```

## 69. Write code to calculate the root mean square error (RMSE) given the lists of values as actual and predicted.

To calculate the root mean square error (RMSE), we have to:

1. Calculate the errors, i.e., the differences between the actual and the predicted values
2. Square each of these errors
3. Calculate the mean of these squared errors
4. Return the square root of the mean

The code in Python for calculating RMSE is given below:

```
def rmse(actual, predicted):  
    errors = [abs(actual[i] - predicted[i]) for i in range(0, len(actual))]  
    squared_errors = [x ** 2 for x in errors]  
    mean = sum(squared_errors) / len(squared_errors)  
    return mean ** .5
```

## 70. Mention the different kernel functions that can be used in SVM.

In SVM, there are four types of kernel functions:

- Linear kernel
- Polynomial kernel
- Radial basis kernel
- Sigmoid kernel

## 71. How to detect if the time series data is stationary?

Time series data is considered stationary when variance or mean is constant with time. If the variance or mean does not change over a period of time in the dataset, then we can draw the conclusion that, for that period, the data is stationary.

## 72. Write code to calculate the accuracy of a binary classification algorithm using its confusion matrix.

We can use the code given below to calculate the accuracy of a binary classification algorithm:

```
def accuracy_score(matrix):  
    true_positives = matrix[0][0]  
    true_negatives = matrix[1][1]  
    total_observations = sum(matrix[0]) + sum(matrix[1])  
    return (true_positives + true_negatives) / total_observations
```

## **73. What does root cause analysis mean?**

Root cause analysis is the process of figuring out the root causes that lead to certain faults or failures. A factor is a root cause if, after eliminating it, a sequence of operations, leading to a fault, error, or undesirable result, ends up working correctly. Root cause analysis is a technique that was initially developed and used in the analysis of industrial accidents, but now, it is used in a wide variety of areas.

## **74. What is A/B testing?**

A/B testing is a kind of statistical hypothesis testing for randomized experiments with two variables. These variables are represented as A and B. A/B testing is used when we wish to test a new feature in a product. In the A/B test, we give users two variants of the product, and we label these variants as A and B.

The A variant can be the product with the new feature added, and the B variant can be the product without the new feature. After users use these two products, we capture their ratings for the product.

If the rating of product variant A is statistically and significantly higher, then the new feature is considered an improvement and useful and is accepted. Otherwise, the new feature is removed from the product.

## **75. Out of collaborative filtering and content-based filtering, which one is considered better, and why?**

Content-based filtering is better than collaborative filtering for generating recommendations. It does not mean that collaborative filtering generates bad recommendations.

However, as collaborative filtering is based on the likes and dislikes of other users, we cannot rely on it much. Also, users likes and dislikes may change in the future.

For example, there may be a movie that a user likes right now but did not like 10 years ago. Moreover, users who are similar in some features may not have the same taste in the kind of content that the platform provides.

In the case of content-based filtering, we make use of users' own likes and dislikes that are much more reliable and yield more positive results. This is why platforms such as Netflix, Amazon Prime, Spotify, etc. make use of content-based filtering for generating recommendations for their users.

## 76. In the following confusion matrix, calculate precision and recall.

Total = 510		Actual	
Predicted		P	N
	P	156	11
	N	16	327

The formulae for precision and recall are given below.

Precision:

$(\text{True Positive}) / (\text{True Positive} + \text{False Positive})$

Recall:

$(\text{True Positive}) / (\text{True Positive} + \text{False Negative})$

Based on the given data, precision and recall are:

Precision:  $156 / (156 + 11) = 93.4$

Recall:  $156 / (156 + 16) = 90.7$

## 77. Write a function that when called with a confusion matrix for a binary classification model returns a dictionary with its precision and recall.

We can use the below for this purpose:

```
def calculate_precision_and_recall(matrix):
    true_positive = matrix[0][0]
    false_positive = matrix[0][1]
    false_negative = matrix[1][0]
    return {
        'precision': (true_positive) / (true_positive + false_positive),
```

```
'recall': (true_positive) / (true_positive + false_negative)
}
```

## 78. What is reinforcement learning?

[Reinforcement learning](#) is a kind of Machine Learning, which is concerned with building software agents that perform actions to attain the greatest number of cumulative rewards.

A reward here is used for letting the model know (during training) if a particular action leads to the attainment of or brings it closer to the goal. For example, if we are creating an ML model that plays a video game, the reward is going to be either the points collected during the play, or the level reached in it.

Reinforcement learning is used to build these kinds of agents that can make real-world decisions that should move the model toward the attainment of a clearly defined goal.

## 79. Explain TF/IDF vectorization.

The expression 'TF/IDF' stands for Term Frequency-Inverse Document Frequency. It is a numerical measure that allows us to determine how important a word is to a document in a collection of documents called a corpus. TF/IDF is used often in text mining and information retrieval.

## 80. What are the assumptions required for linear regression?

**There are several assumptions required for linear regression. They are as follows:**

- The data, which is a sample drawn from a population, used to train the model should be representative of the population.
- The relationship between independent variables and the mean of dependent variables is linear.
- The variance of the residual is going to be the same for any value of an independent variable. It is also represented as X.
- Each observation is independent of all other observations.

- For any value of an independent variable, the independent variable is normally distributed.

## **81. What happens when some of the assumptions required for linear regression are violated?**

These assumptions may be violated lightly (i.e., some minor violations) or strongly (i.e., most of the data has violations). Both violations will have different effects on a linear regression model.

Strong violations of these assumptions make the results entirely redundant. Light violations of these assumptions make the results have greater bias or variance.



# Neural Networks Related Questions

## 1. What kind of a neural network will you use in deep learning regression via Keras-TensorFlow? How will you decide the best neural network model for a given problem?

The foremost step when deciding on choosing a [neural network](#) model is to have a good know-how of the data and then decide the best model for it. Also, factoring in whether it is a linearly separable problem or not is important when deciding on a neural network model. So, the task at hand and the data plays a vital role in choosing the best neural network model for a given problem. However, it is always better to start with a simple model like multi-layer perceptron (MLP) that has just one hidden layer unlike CNN, LSTM, or RNN that require configuring the nodes and layers. MLP is considered the simplest neural network because the weight initialization is not sensitive, and there is no need to define a structure for the network beforehand.

## 2. Why do we need autoencoders when there are already powerful dimensionality reduction techniques like Principal Component Analysis?

The curse of dimensionality (the problems that arise when working with high-dimensional data) is a common problem when [working on machine learning](#) or deep learning projects. Curse of Dimensionality causes lots of difficulties while training a model because it requires training a lot of parameters on a scarce dataset leading to issues like overfitting, large training times, and poor generalization. PCA and autoencoders are used to tackle these issues. PCA is an unsupervised technique wherein the actual data is projected to the direction of high variance, while autoencoders are [neural networks](#) used for compressing the data into a low dimensional latent space and then try to reconstruct the actual high dimensional data.

PCA or autoencoders are effective only when the features have some relationship with each other. A general rule of thumb between choosing PCA and Autoencoders is the size

of data. Autoencoders work great for larger datasets, and PCA works well for smaller datasets. Autoencoders are usually preferred when there is a need for modeling non-linearities and relatively complex relationships. Autoencoders can encode a lot of information with fewer dimensions when there is a curvature in low dim structure or non-linearity, making them a better choice over PCA in such scenarios.

Autoencoders are usually preferred for identifying data anomalies rather than for reducing data. Anomalous data points can be identified using the reconstruction error, PCA is not good for reconstructing data particularly when there are non-linear relationships.

### **3. Say you have to build a neural network architecture; how will you decide how many neurons and hidden layers are needed for the network?**

Given a business problem, there is no hard and fast rule to determine the exact number of neurons and hidden layers required to build a neural network architecture. The optimal size of the hidden layer in a neural network lies between the size of the output layers and the size of the input. However, here are some common approaches that have the advantage of making a great start to building a neural network architecture:

- To address any specific real-world predictive modeling problem, the best way is to start with rough systematic experimentation and find out what would work best for any given dataset based on prior experience [working with neural networks](#) on similar real-world problems. Based on the understanding of any given problem domain and one's experience working with neural networks, one can choose the network configuration. The number of layers and neurons used on similar problems is always a great way to start testing the configuration of a neural network.
- It is always advisable, to begin with, simple neural network architecture and then go on to enhance the complexity of the neural network.
- Try working with varying depths of networks and configure deep neural networks only for challenging predictive modeling problems where depth can be beneficial.

#### **4. Why CNN is preferred over ANN for Image Classification tasks even though it is possible to solve image classification using ANN?**

One common problem with using ANN's for image classification is that ANN's react differently to input images and their shifted versions. Let's consider a simple example where you have the picture of a dog in the top left of an image and in another image, there is a picture of a dog at the bottom right. ANN will assume that a dog will always appear in this section of any image, however, that's not the case. ANN's require concrete data points meaning if you are building a deep learning model to distinguish between cats and dogs, the length of the ears, the width of the nose, and other features should be provided as data points while if using CNN for image classification spatial features are extracted from the input images. When there are thousands of features to be extracted, CNN is a better choice because it gathers features on its own, unlike ANN where each individual feature needs to be measured.

Training a neural network model becomes computationally heavy (requiring additional storage and processing capability) as the number of layers and parameters increases. Tuning the increased number of parameters can be a tedious task with ANN, unlike CNN where the time for tuning parameters is reduced making it an ideal choice for image classification problems.

#### **5. Why Sigmoid or Tanh is not preferred to be used as the activation function in the hidden layer of the neural network?**

A common problem with Tanh or Sigmoid functions is that they saturate. Once saturated, the learning algorithms cannot adapt to the weights and enhance the performance of the model. Thus, Sigmoid or Tanh activation functions prevent the neural network from learning effectively leading to a vanishing gradient problem. The vanishing gradient problem can be addressed with the use of Rectified Linear Activation Function (ReLU) instead of sigmoid and using a Xavier initialization.

## 6. Why does the exploding gradient problem happen?

When the model weights grow exponentially and become unexpectedly large in the end when training the model, exploding gradient problem happens. In a neural network with  $n$  hidden layers,  $n$  derivatives are multiplied together. If the weights that are multiplied are greater than 1, then the gradient increases exponentially greater than the usual one and eventually explodes as you propagate through the model. The situation wherein the value of weights is more than 1, makes the output exponentially larger hindering the model training and impacting the overall accuracy of the model. This is called the exploding gradient problem. Exploding gradients is a serious problem because the model cannot learn from its training data resulting in a poor loss. One can deal with the exploding gradient problem either by gradient clipping, weight regularization, or with the use of LSTMs.

## 7. How to fix the constant validation accuracy in CNN model training?

Constant validation accuracy is a common problem when training any neural network because the network just remembers the sample and results in an overfitting problem. Overfitting of a model means that the neural network model works fantastic on the training sample, but the performance of the model sinks in on the validation set. Here are some tips to try to fix the constant validation accuracy in CNN:

- It is always advisable to divide the dataset into training, validation, and test set.
- When working with little data, this problem can be solved by changing the parameters of the neural network by trial and error.
- Increasing the size of the training dataset.
- Use batch normalization.
- Regularization.
- Reduce the network complexity.

## **8. What do you understand by learning rate in a neural network model? What happens if the learning rate is too high or too low?**

Learning rate is one of the most important configurable hyperparameters used in the training of a neural network. The value of the learning rate lies between 0 and 1. Choosing the learning rate is one of the most challenging aspects of training a neural network because it is the parameter that controls how quickly or slowly a neural network model adapts to a given problem and learns. A higher learning rate value means that the model requires few training epochs and results in rapid changes while a smaller learning rate implies that the model will take a long time to converge or might never converge and get stuck on a suboptimal solution. Thus, it is advisable not to use a learning rate that is too low or too high, but instead a good learning rate value should be discovered through trial and error.

## **9. What kind of a network would you prefer – a shallow network or a deep network for voice recognition?**

Every neural network has a hidden layer along with input and output layers. Neural networks that use a single hidden layer are known as shallow neural networks, while those that use multiple hidden layers are referred to as deep neural networks. Both shallow and deep networks are capable of fitting into any function, but shallow networks require a lot of parameters, unlike deep networks that can fit functions even with a limited number of parameters because of several layers. Deep networks are preferred today over shallow networks because at every layer the model learns a novel and abstract representation of the input. Also, they are much more efficient in terms of the number of parameters and computations compared to shallow networks.

## **10. Can you train a neural network model by initializing all biases as 0?**

Yes, there is a possibility that the neural network model will learn even if all the biases are initialized to 0.

## **11. Can you train a neural network model by initializing all the weights to 0?**

No, it is not possible to train a model by initializing all the weights to 0 because the neural network will never learn to perform a given task. Initializing all weights to zeros will cause the derivatives to remain the same for every  $w$  in  $W$  [1] because of which neurons will learn the same features in every iteration. Not just 0, but any kind of constant initialization of weights is likely to produce a poor result.

## **12. Why is it important to introduce non-linearities in a neural network?**

Without non-linearities, a neural network will act like a perceptron regardless of how many layers are there making the output linearly dependent on the input. In other words, having a neural network with  $n$  layers and  $m$  hidden units with linear activation functions is just like having a linear neural network without hidden layers that can only find linear separation boundaries. A neural network without non-linearities cannot find appropriate solutions and classify the data correctly for complex problems.

## **13. Why dropout is effective in deep networks?**

The problem with deep neural networks is that they are most likely to overfit training data with few examples. Overfitting can be reduced by ensembles of networks with different model configurations, but this requires the additional effort of maintaining multiple models and is also computationally expensive. Dropout is one of the easiest and exceptionally successful methods to reduce dependencies in deep neural networks and overcome overfitting problems. When using the dropout regularization method, a single neural network model is used to similar different network architecture by dropping out nodes while training. It is considered an effective method of regularization as it improves generalization errors and is also computationally cheap.

#### **14. A deep learning model finds close to 12 million face vectors. How will you find a new face quickly?**

You will need to know about One-Shot Learning for Face Recognition which is a classification task where one or more examples (faces in this case) are used for classifying new examples (faces) in the future. One needs to know about the method of indexing data to retrieve a new face faster. A new face can be recognized by finding the vectors that are most similar to the input face but in this case, the system would have become super slow if we were to calculate the distance to 12 million vectors. A convenient way would be to index data on real vector space by dividing the data into easy structures for querying (almost like a tree data structure). It is easier to find the vector that is in close proximity with time very quickly whenever new data is available. Techniques like Annoy Indexing, Locality Sensitive Hashing, and Approximate Nearest Neighbours can be used for this purpose.

#### **15. What has fostered the implementation and experimentation of powerful neural network architectures in the industry?**

Flexibility makes deep learning powerful. Neural networks are universal function approximators so even if it is a complex enough problem at hand (where the formula between input and output is not known), a neural network can be approximated. Also, transfer learning (where the trained weights of an existing neural network can be used to initialize the weights of another network that performs similar tasks) makes the application of deep learning much easier under situations when training a neural network from scratch is costly or almost impossible when there is data scarcity.

Faster and powerful computational resources are also a prime reason for the adoption of neural network architectures. One cannot deny the fact that it is faster to train a neural network in just minutes with GPU acceleration which would otherwise take days for the network to learn.

## **16. Can you build deep learning models based solely on linear regression?**

Yes, it is possible to build deep networks using a linear function as the activation function for each layer if the problem is represented by a linear equation. However, a problem that is a composition of linear functions is a linear function and there is nothing extraordinary that can be achieved with the implementation of a deep network because adding more nodes to the network will not increase the predictive power of the [machine learning model](#).

## **17. When training a deep learning model, you observe that after a few epochs the accuracy of the model decreases. How will you address this problem?**

The decrease in the accuracy of a deep learning model after a few epochs implies that the model is learning from the characteristics of the dataset and not considering the features. This is referred to as the overfitting of the deep learning model. You can either use dropout regularization or early stopping to fix this issue. Early stopping as the phrase implies stops training the deep learning model any further the moment you notice a drop in accuracy of the model. Dropout regularization is a technique wherein a few nodes or output layers are dropped so that the remaining nodes have varying weights.

## **18. What is the impact on a model with an improperly set learning rate on weights?**

With images as inputs, an improperly set learning rate can cause noisy features. Having an ill-chosen learning rate determines the prediction quality of a model and can result in an unconverged neural network.

## **19. What do you understand by the terms Batch, Iterations, and Epoch in training a neural network model?**

- Epoch refers to the iteration where the complete dataset is passed forward and backward through the neural network only once.



- It is not possible to pass the complete dataset to the network in one go so the dataset is divided into parts. This is referred to as the Batch.
- The total number of batches needed to complete one epoch is referred to as iteration. For example, if you have 60,000 data rows and the batch size is 1000 then each epoch will run 60 iterations.

## 20. **Is it possible to calculate the learning rate for a model a priori?**

For simple models, it could be possible to set the best learning rate value a priori. However, for complex models, it is not possible to calculate the best learning rate through theoretical deductions that can make accurate predictions. Observations and experiences do play a vital role in defining the optimal learning rate.

## 21. **What is the theoretical foundation of neural networks?**

To answer this question, one needs to explain the universal approximation theorem that forms the base on why neural networks work.

*Introducing non-linearity via an activation function allows us to approximate any function. It's quite simple, really. -- Elon Musk --*

According to the Universal Approximation Theorem, a neural network having a single hidden layer containing a finite number of neurons can approximate any continuous function to a reasonable accuracy for inputs in a specific range. However, if the function has large gaps, it is not possible to approximate it. Meaning, if a neural network is trained with inputs between 20 and 30, we cannot be assured that it will work well for inputs between 60 and 70.

## 22. **What are the commonly used approaches to set the learning rate?**

- Using a fixed learning rate value for the complete learning process.
- Using a learning rate schedule.
- Making use of adaptive learning rates.

- Adding momentum to the classical SGD equation.

## 23. **Is there any difference between neural networks and deep learning?**

Ideally, there is no significant difference between deep learning networks and neural networks. Deep learning networks are neural networks but with a slightly complex architecture than they were in 1990s. It is the availability of hardware and computational resources that has made it feasible to implement them now.

## 24. **You want to train a deep learning model on a 10GB dataset, but your machine has 4GB RAM. How will you go about implementing a solution to this deep learning problem?**

One of the possible ways to answer this question would be to say that a neural network can be trained by loading the data into the NumPy array and defining a small batch size, NumPy doesn't load the complete dataset into the memory but creates a complete mapping of the dataset. NumPy offers several tools for compressing large datasets that can be integrated with other NN packages like PyTorch, [TensorFlow](#), or Keras.

## 25. **How will the predictability of a neural network impact if you use a ReLu activation function and then use the Sigmoid function in the final layer of the network?**

The neural network will predict only one class for all types of inputs because the output of a ReLu activation function is always a non-negative result.

## 26. **What are the limitations of using a perceptron?**

A major drawback to using a perceptron is that they can only linearly separable functions and cannot handle non-linear inputs.

## 27. **How will you differentiate between a multi-class and multi-label classification problem?**

In a multi-class classification problem, the classification task has more than two mutually exclusive classes whereas in a multi-label problem each label has a different classification task, however, the tasks are related somehow. For example, classifying a set of images of animals which may be cats, dogs, or bears is a multi-class classification problem that assumes that each sample has only one label meaning an image can be classified as either a cat or a dog but not at the same time. Now imagine that you want to process the below image. The image shown below needs to be classified as both cat and dog because the image shows both the animals. In a multi-label classification problem, a set of labels are assigned to each sample and the classes are not mutually exclusive. So, a pattern can belong to one or more classes in a multi-label classification problem.

## 28. **What do you understand by transfer learning?**

You know how to ride a bicycle, so it will be easy for you to learn to drive a bike. This is transfer learning. You have some skills, and you can learn a new skill that relates to it without having to learn it from scratch. Transfer learning is a process in which the learning can be transferred from one model to another without having to make the model learn everything from scratch. The features and weights can be used for training the new model providing reusability. Transfer learning works well in training a model easily when there is limited data.

## 29. **What is fine-tuning and how is it different from transfer learning?**

In transfer learning, the feature extraction part remains untouched and only the prediction layer is retrained by changing the weights based on the application. On the

contrary in fine-tuning, the prediction layer along with the feature extraction stage can be retrained making the process flexible.

### **30. Why do we use convolutions for images instead of using fully connected layers?**

Each convolution kernel in a CNN acts like its own feature detector and has a partially in-built translation in-variance. Using convolutions lets one preserve, encode, and make use of the spatial information from the image, unlike fully connected layers that do not have any relative spatial information

### **31. What is Gradient Clipping?**

Gradient Clipping is used to deal with the exploding gradient problem that occurs during the backpropagation. The gradient values are forced to a particular minimum or maximum value if the gradient has crossed the expected range. Gradient clipping provides numerical stability while training a neural network, but does not provide any performance improvements.

### **32. What do you understand by end-to-end learning?**

It is a deep learning process where a model gets raw data as the input and all the various parts are trained simultaneously to produce the desired outcome with no intermediate tasks. The advantage of end-to-end learning is that there is no need for implicitly doing [feature engineering](#) which usually leads to a lower bias. A good example that you can quote in the content of end-to-end learning is driverless cars. They use human-provided input as guidance and are trained to automatically learn and process the information using a CNN to complete tasks.

### **33. Are convolutional neural networks translation-invariant?**

Convolutional neural networks are translation invariant only to a certain extent, but pooling can make them translation invariant. Making a CNN completely translation-invariant might not be possible. However, by feeding the right kind of data this can be achieved, although this might not be a feasible solution.

### **34. What is the advantage of using small kernels like 3x3 than using a few large ones.**

Smaller kernels let you use more filters so you can use a greater number of activations functions and let the CNN learn a more discriminative mapping function. Also, smaller kernels capture more spatial context and use fewer computations and parameters making them a better choice over large ones.

### **35. How can you generate a dataset on multiple cores in real-time that can be fed to the deep learning model?**

One of the major challenges today in CV is the need to load large datasets of videos and images but there is not enough memory on the machine. In such situations, data generators act as a magic wand when it comes to loading a dataset that is memory-consuming. You can talk about the various data generators Keras model class provides. When working with big data, in most of the cases it might not be required to load all the data into RAM as it would be memory wastage, could lead to memory overflow, and take a longer time to process. Making use of generative functions is highly beneficial then as they generate the data to be directly fed into the model in each batch for training.

### **36. How do you bring balance to the force when handling imbalanced datasets in deep learning?**

It is next to impossible to have a perfectly balanced real-world dataset when working on deep learning problems so there will be some level of class imbalance within the data that can be tackled either by:

- Weight Balancing
- Over and Under Sampling

### 37. **What are the benefits of using batch normalization when training a neural network?**

- Batch normalization optimizes the network training process making it easier to build and faster to train a deep neural network.
- Batch normalization regulates the values going into each activation function making activation functions more viable because non-linearities that don't seem to work well become viable with the use of batch normalization.
- Batch normalization makes it easier to initialize weights and allows the use of higher learning rates ultimately increasing the speed at which the network trains.

### 38. **Which is better LSTM or GRU?**

LSTM works well for problems where accuracy is critical and sequence is large whereas if you want less memory consumption and faster operations, opt for GRU. Refer here for a detailed answer: </recipes/what-is-difference-between-gru-and-lstm-explain-with-example>

### 39. **RMSProp and Adam optimizer adjust gradients? Does this mean that they perform gradient clipping?**

This does not inherently mean that they perform gradient clipping because gradient clipping involves setting up predetermined values beyond which the gradients cannot go, unlike Adam and RMSProp that make multiplicative adjustments to gradients.

### 40. **Can you name a few hyperparameters used for training a neural network.**

When training any neural networks there are two types of hyperparameters-one that define the structure of the neural network and the other determining how a neural

network is trained. Listed are a few hyperparameters that are set before training any neural network:

- Initialization of weights
- Setting the number of hidden layers
- Learning Rate
- Number of epochs
- Activation Functions
- Batch Size
- Momentum

#### **41. When is multi-task learning usually preferred?**

Multi-task learning with deep neural networks is a subfield wherein several tasks are learned by a shared model. This reduces overfitting, enhances data efficiency, and speeds up the learning process with the use of auxiliary information. Multi-task learning is useful when there is a small amount of data for any given task, and we can benefit from training a deep learning model on a large dataset.

#### **42. Explain the Adam Optimizer in one minute.**

Adaptive momentum or Adam optimizer is an optimization algorithm designed to deal with sparse gradients on noisy problems. Adam optimizer improves convergence through momentum that ensures that a model does not get stuck in saddle point and provides per-parameter updates for faster convergence.

#### **43. Which loss function is preferred for multi-category classification?**

Cross-Entropy loss function.

#### **44. To what kind of problems can the cross-entropy loss function be applied?**

- Binary Classification Problems

- Multi-Label Classification Problems
- Multi-Category Classification Problems

**45. List the steps to implement a gradient descent algorithm.**

- The first step is to initialize random weight and bias.
- Get values from the output layer by passing the input through the neural network.
- Determine the error between the actual and predicted value.
- Based on the neurons that contribute to the error, modify the values to minimize the error.
- Repeat the process until the optimal weights are found for the neural network.

**46. How important is it to shuffle the training data when using batch gradient descent?**

Shuffling the training dataset will not make much of a difference because the gradient is calculated at every epoch using the complete training dataset.

**47. What is the benefit of using max pooling in classification convolutional neural networks?**

The feature maps become smaller after max-pooling in CNN and hence help reduce the computation and give more translation in-variance. Also, we don't lose much semantic information because we're taking the maximum activation.

**48. Can you name a few data structures that are commonly used in deep learning?**

You can talk about computational graphs, tensors, matrices, data frames, and lists.



**49. Can you add an L2 regularization to a recurrent neural network to overcome the vanishing gradient problem?**

This can worsen the vanishing gradient problem because the L2 regularization will shrink weights towards zero.

**50. How will you implement Batch Normalization in RNN?**

It is not possible to use batch normalization in RNN because statistics are computed per batch and thus batch normalization will not consider the recurrent part of the neural network. An alternative to this could be layer normalization in RNN or reparametrizing the LSTM layer that allows the use of batch normalization.

# 10 Deep Learning Questions

## 1. Given that there are so many deep learning algorithms, how will you determine which deep learning algorithm has to be used for a dataset.

Artificial Neural Network Artificial [Neural Network](#) or sometimes called Classic Neural Network is a connection of multilayered perceptrons. This algorithm can be used when the data is properly structured in a tabular form. Both Classification and regression problems can be solved using ANNs Convolutional Neural Networks These networks are the best proven ones to build any prediction model involving image data as input. To put it in general terms, CNN works best on data with spatial relationships and hence these can also produce state-of-the-art results for NLP problems such as topic modelling, document classification and so on. Recurrent Neural Networks RNNs come into picture when we have sequential data where the order of the data entered is also important. RNNs can provide solutions for problems involving [Time Series](#) data. More often, rather than vanilla RNNs, gated networks like LSTMs (Long-short-term memory) and GRUs (Gated Recurrent units) are proven to give much better results. Autoencoders are widely used in the deep learning community these days because of its ability to operate automatically based on its inputs even before taking an activation function and final output decoding. These can be used when we have problems such as feature detection, recommendation systems and other compelling problems.

## 2. How do one-hot encoding and label encoding affect the dimensionality of a dataset?

Label encoding does not really affect the dataset in any way because in label encoding, we only provide labels to each category in the column.

For example,

Place of birth (before label encoding)	Place of birth (after label encoding)
Delhi	0
Hyderabad	1
Chennai	2
Delhi	0

In the above example, we are mapping Delhi -> 0, Hyderabad -> 1, and Chennai -> 2.

In one hot encoding, we create columns to each of the category in the dataset. Thus, the more the number of categories in the column, the more are the columns generated after one hot encoding. Let us consider the very same dataset that we saw above. After one hot encoding it will look like the table shown below

Place of birth (Delhi)	Place of birth (Hyderabad)	Place of birth (Chennai)
1	0	0
0	1	0
0	0	1

1	0	0
---	---	---

If the value is 'Delhi', then only the column meant for 'Delhi' takes the value 1 and the other columns takes the value 0.

Often, we don't consider the last/first category after one hot encoding the variable because it can be clearly understood that if all the existing entries for the category are 0, then it belongs to the category that we dropped. This is much clearly explained with the example below

Place of birth (Delhi)	Place of birth (Hyderabad)
1	0
0	1
0	0
1	0

Here, we already know that there are 3 unique categories in the variable (Delhi, Hyderabad, and Chennai). There are two zeros in the 3rd row which clearly implies that it does not belong to both the categories and the one which remains in Chennai. Therefore, the decoded value for that row is Chennai.

### 3. Why are GPUs important for implementing deep learning models?

Whenever we are trying to build any neural network model, the model training phase is the most resource-consuming job. Each iteration of model training comprises thousands

(or even more) of matrix multiplication operations taking place. This is where GPUs come into the picture. GPUs (Graphics Processing Units) are nothing but CPUs but with more ALUs (Arithmetic logic units) than our normal CPUs which are specifically meant for this kind of heavy mathematical computation.

#### **4. Which is the best algorithm for face detection?**

There are several machine learning algorithms available for face detection, but the best ones are the ones which involve CNNs and deep learning. Some notable algorithms for face detection are the following: FaceNet, Probabilistic Face Embedding ArcFace Cosface Spherface

#### **5. What evaluation approaches do you use to gauge the effectiveness of deep learning models?**

#### **6. When training a neural network, you observe that the loss does not decrease in the first few epochs. What are the possible reasons for this?**

#### **7. What are the commonly used techniques to deal with the overfitting of a deep learning model?**

#### **8. What kind of gradient descent variant is the best for handling data that is too big to handle in RAM simultaneously?**

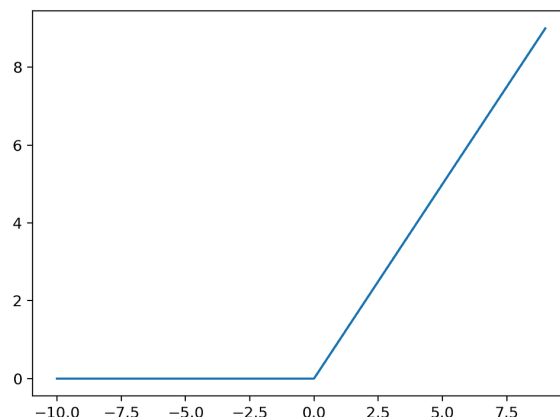
#### **9. How will you explain the success and recent rise in demand for deep learning in the industry?**

#### **10. How do you select the depth of a neural network?**

# Other Deep Learning Technical Interview Questions

1. What is Deep Learning?
2. Which deep learning framework do you prefer to work with – PyTorch or TensorFlow and why? Refer [PyTorch vs Tensorflow](#) for answer
3. Talk about a deep learning project you've worked on and the tools you used
4. Have you used the ReLu activation function in your neural network? Can you explain how does the ReLu activation function works?

Yes, I have used ReLu in my neural networks. ReLu stands for Rectified Linear Unit. Basically, the function returns the input value as it is if it is positive or returns zero if it is negative. If the function is plotted in a line graph, it would look like the graph shown below



The main purpose of formulating this function was to overcome the Vanishing gradient problem caused by preliminary activation functions like Sigmoid and TanH which prevented us from building deeper neural network models. Now a days, this function has become a default activation function for many types of neural network models because models that use this function are easily trainable and don't suffer from the vanishing gradient problem.

**5. How often do you use pre-trained models for your neural network?**

**6. What does the future of video analysis look like with the use of deep learning solutions? How effective/good is video analysis currently?**

**7. Tell us about your passion for deep learning. Do you like to participate in deep learning/machine learning hackathons, write blogs around novel deep learning tools, or attend local meetups, etc...?**

**8. Describe the last time you felt frustrated solving a deep learning challenge, and how did you overcome it?**

**9. What is more important to you the performance of your deep learning model or its accuracy?**

**10. Given the dataset, how will you decide which deep learning model to use and how to implement it?**

- 11. What is the last deep learning research paper you've read?**
- 12. What are the most used neural network paradigms?  
(Hint: Talk about Encoder-Decoder Structures, LSTM, GAN, and CNN)**
- 13. Is it possible to use a neural network as a tool of dimensionality reduction?**
- 14. How deep learning models tackle the curse of dimensionality?**
- 15. What are the pros and cons of using neural networks?**

**Pros:**

Neural networks are highly flexible and can be used for both classification and regression problems and sometimes for problems much more complex than that. Neural networks are highly scalable. We can add as many layers with as many neurons as we want. Neural networks are proven to produce best results when we have a lot of data points. They work best for non-linear data such as image data, text data and so on. They can be used on any data that can be converted to numbers.

**Cons:**

1. The well-known disadvantage of neural networks is their "black box" nature. That is, we don't know how or why our neural network came up with a certain output. For example, when we feed an image of a dog into a neural network and it predicts it to be a duck, we may find it difficult to understand what caused it to arrive at this prediction.
2. Developing a neural network model takes much time.
3. Neural networks are more computationally expensive than traditional algorithms.



4. The amount of computational power needed for a neural network depends mostly on the size of data, depth and complexity of the network.

5. To train a neural network model, it requires much more data than training a traditional machine learning model.

## **16. How is a Capsule Neural Network different from a Convolutional Neural Network?**

## **17. What is a GAN and what are the different types of GAN you've worked with?**

## **18. For any given problem, how do you decide if you have to use transfer learning or also fine-tuning?**

Transfer learning is a method used when a model is developed for one task is reused to work on a second task. Fine tuning is one approach to achieve transfer learning. In Transfer Learning we train the model with a dataset and after we train the same model with another dataset that has a different distribution of classes. In Fine-tuning, an approach of Transfer Learning, we have a dataset, and we make an 80-20 split and use 80% of it in training. Then we train the same model with the remaining 20%. Usually, we change the learning rate to a smaller one, so it does not have a significant impact on the already adjusted weights. To decide which method to choose, one should experiment first by using transfer learning as it is easy and fast, and if it does not suffice the purpose, then use fine tuning.

## **19. Can you share some tricks or techniques that you use to fight to overfit a deep learning model and get better generalization?**

Overfitting of a model is defined when a model performs well on the training data (low bias) and performs badly / poorly on the test data (high variance). In short, the model has learned over a certain pattern of data and is not useful for any other data. Overfitting can

be detected by checking the performance metrics like loss and accuracy of a given model. There are several tips and techniques one can use to reduce the over fitting of a deep learning model:

- Apply regularization.
- Increase the size of training data.
- Reduce number of layers in the hidden layer, this will reduce the networks capacity.
- Add dropout layers.
- Early stopping – try to stop the training before the validation loss increases.
- Make use of data augmentation.

## **20. Explain the difference between Gradient Descent and Stochastic Gradient Descent.**

To begin with, Gradient descent and stochastic gradient descent are popular machine learning and deep learning optimization algorithms which are used for updating a set of parameters in an iterative way to minimize an error function. In gradient descent in order to update parameters, the entire dataset set is to be considered for a particular iteration while in stochastic gradient descent, computation is carried over only one single training sample. For example, if a dataset has 10000 datapoints, then GD will train on all the 10000 datapoints and this will take a longer time, while on the other hand, Stochastic GD will be much faster as we will train on only a single sample and update the parameters. This is because Stochastic gradient descent usually converges faster than gradient descent on large datasets, because updates are more frequent.

## **21. Which one do you think is more powerful – a two-layer NN without any activation function or a two-layer decision tree?**

- When you say a two-layer neural network, it basically contains, one input layer, one hidden layer and one output layer. An activation function is important while dealing with neural networks as they are needed while dealing with complicate and nonlinear complex functional mappings between inputs and response variable.

- When a two-layer neural network has no activation function, it is just a linear network. A Neural Network without Activation function would simply be a Linear regression Model, which has limited power and does not perform good most of the times.
- Two-layer decision tree is just a decision tree with depth of 2.
- So, while comparing between these two models, two-layer neural network (without activation function) is more powerful than the two-layer decision tree, since two-layer neural network will take more attributes into consideration while building a model and in case of 2-layer decision tree, only 2 or 3 attributes will be considered.

## **22. Can you name the breakthrough project that garnered the popularity and adoption of deep learning?**

- The last decade has seen remarkable improvements in the ability of computers to understand the world around them. One of these breakthroughs is an artificial intelligence technique called deep learning.
- Deep learning, unlike machine learning, is based on neural networks, a type of data structure loosely inspired by networks of biological neurons. Neural networks are organized into layers, with inputs from one layer connected to outputs from the next layer.
- Computer scientists have been experimenting neural networks since the 1950s. But two significant breakthroughs—one in 1986, the other in 2012—laid the foundation for today's vast deep learning industry.
- The fortunes of neural networks were revived by a famous 1986 paper (link: [https://www.iro.umontreal.ca/~vincentp/ift3395/lectures/backprop\\_old.pdf](https://www.iro.umontreal.ca/~vincentp/ift3395/lectures/backprop_old.pdf)) that introduced the concept of backpropagation, a practical method to train deep neural networks.
- Backpropagation made deeper networks more computationally tractable, but those deeper networks still required more computing resources than shallower networks.
- Research results in the 1990s and 2000s often suggested diminishing returns to making neural networks more complex. Then a famous 2012 paper(link: <https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a>)

[68c45b-Paper.pdf](#))—which described a neural network dubbed AlexNet after lead researcher Alex Krizhevsky—transformed people's thinking.

- Dramatically deeper networks could deliver breakthrough performance, but only if they were combined with ample computing power and lots and lots of data.

## **23. Differentiate between bias and variance with respect to deep learning models and how can you achieve a balance between the two?**

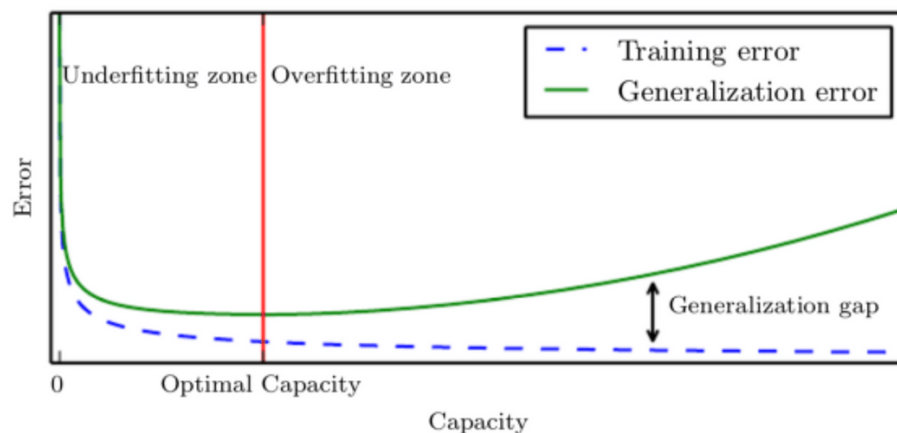
There are mainly two broad types of errors: reducible and irreducible. Among the reducible errors, we have bias and variance. Gaining a proper understanding of these errors helps one build an accurate model by avoiding overfitting and underfitting of the model.

In order to obtain the optimal balance between the two errors, the model must always aim at maintaining a low bias and a low variance. An optimal balance of bias and variance would never overfit or underfit the model.

**Bias** – In the above diagram, the training error (blue dotted line) is high in the initial stage (high bias) and then decreases sustainably (low bias). High bias means that the data is under fitting, and hence, the data must have a low bias to achieve good results. In order to achieve a low bias:

- I. Try increasing the number of iterations / epochs
- II. Try a bigger network

**Variance** – the variance in deep learning is nothing but the difference between the validation error and the training error. In the above figure, we can see that the gap between the training error and validation error is high, i.e., the variance is high. This is the case of overfitting. The model should have low variance and can be achieved by: i. Increasing the training data ii. Using regularization iii. Using different neural network architectures.



## 24. What are your thoughts about using GPT3 for our business?

GPT-3, or the third generation Generative Pre-trained Transformer, is a neural network machine. GPT-3 is a text predictor. Given a text or phrase, GPT-3 returns a human-like response to text completion in natural language. GPT-3 has a wide range of applications serving the industry today. It is a powerful tool that can create applications for responding to customer queries, language translator (say, asking a question in English and expecting an answer in Spanish) etc.

GPT3 can also do everything from creating spreadsheets to building complex CSS or even deploying Amazon Web Services (AWS) instances. So, can using GPT-3 help your business? Well, it can help in many ways. It all depends on what you need it to do, but it is a super versatile deep learning model applied to many applications.

Some more applications of GPT-3 that you can probably use in your business are:

- Generate emails from short descriptions. An application that can expand the given brief description into a formatted and grammatically correct professional email.
- Generate python codes from a description. Generate Flask (Python) API code just by describing the functions in English using GPT-3.
- Generate a deep learning model based on a description. For more details related to GPT-3 applications, visit the following link: <https://www.louisbouchard.ai/can-gpt-3-really-help-you/>

## **25. Can you train a neural network without using back-propagation? If yes, what technique will you use to accomplish this?**

- In a neural network, back propagation is the process of repeatedly adjusting the weights of the layers in the network to minimize the difference between the actual output and the desired output, i.e., the loss.
- These adjusted weights result in making the hidden units of neural network to represent key features of the data. Are there any other ways to carry on the process rather than back propagation?
- Indeed, there are various optimization algorithms that does not require back-propagation to train the neural network.
- Among them are evolutionary optimization and Jeff Hinton's capsule routing. However, none of these methods exhibit a competitive performance against back-propagation based algorithms.

## **26. Describe your research experience in the field of deep learning**

## **27. Explain the working of a perceptron.**

- Perceptron's were developed in the 1950s and 1960s by the scientist Frank Rosenblatt, inspired by earlier work by Warren McCulloch and Walter Pitts.
- A perceptron is one of the simplest ANN (artificial neural network) unit that does certain computations to detect features or business intelligence in the input data.
- Perceptron is based on an artificial neuron called a threshold logic unit (TLU)
- The inputs and output are numbers rather than binary values and each input connection is associated with a weight.
- The TLU computes a weighted sum of its inputs:  $(z = w_1 x_1 + w_2 x_2 + \dots + w_n x_n = w^T x)$ , then applies a step function to that sum and outputs the result:  $hw(x) = \text{step}(z)$ , where  $z = w^T x$ .

- A single TLU can be used for simple linear binary classification.

**28. Differentiate between a feed-forward neural network and a recurrent neural network.**

**29. Why don't we see the exploding or vanishing gradient problem in feed-forward neural networks?**

**30. How do you decide the size of the filter when performing a convolution operation in a CNN?**

- While performing a convolution operation in CNN, filters detect spatial patterns such as edges in images by detecting the changes in the intensity of values of the images.
- There is no particular answer to how many filters, or the best number of filters one can use.
- To decide the filter size, I would say it strongly depends on the type and complexity of the image data.
- A fair number of features is learned from experience after repeatedly working with similar types of datasets.
- In general, the more features you want to capture in an image, the higher the number of filters required in a CNN. The number of filters is a hyper-parameter that can be later tuned.

**31. When designing a CNN, can we find out how many convolutional layers should we use?**

- While designing a CNN, Convolutional layers are the layers where filters are applied to the original image, or to other feature maps in a deep CNN.
- The more convolutional layers the better, since each convolutional layer reduces the number of input features to the fully connected layers. Nonetheless, after about two or three layers the accuracy gain becomes rather small, so you need to decide whether your focus is generalization accuracy or training time.
- All image recognition tasks are different, so the best method is to simply try incrementing the number of convolutional layers one at a time until you are satisfied by the result.

**32. What do you understand by a computational graph?**

**33. Differentiate between PCA and Autoencoders.**

**34. Which one is better for reconstruction linear autoencoder or PCA?**

**35. How is deep learning related to representation learning?**

**36. Explain the Borel Measurable function.**

**37. How are Gradient Boosting and Gradient Descent different from each other?**

**38. In a logistic regression model, will all the gradient descent algorithms lead to the same model if run for a long time?**



39. What is the benefit of shuffling a training dataset when using batch gradient descent?
40. Explain the cross-entropy loss function.
41. Why is cross-entropy preferred as the cost function for multi-class classification problems?
42. What happens if you do not use any activation functions in a neural network?
43. What is the importance of having residual neural networks?
44. There is a neuron in the hidden layer that always results in a large error in backpropagation. What could be the reason for this?
45. Explain the working of forwarding propagation and backpropagation in deep learning.
46. Is there any difference between feature learning and feature extraction?
47. Do you know the difference between the padding parameters valid and the same padding in a CNN?
48. How does deep learning outperform traditional machine learning models in time series analysis?

- 49. Can you explain the parameter sharing concept in deep learning?**
- 50. How many trainable parameters are there in a Gated Recurrent Unit cell and in a Long Short Term Memory cell?**
- 51. What are the key components of LSTM?**
- 52. What are the components of a General Adversarial Network?**