1.What is the need of regularization in machine learning?

Ans - Regularization is a technique used in machine learning to prevent overfitting and improve the generalization performance of models. In essence, regularization adds a penalty term to the loss function, discouraging the model from learning overly complex patterns that may not generalize well to unseen data. This helps create simpler, more robust models.

2.What is Gini–impurity index?

Ans - The Gini Index is also known as Gini impurity. It is a measure of how mixed or impure a dataset is. The Gini impurity ranges between 0 and 1, where 0 represents a pure dataset and 1 represents a completely impure dataset(In a pure dataset, all the samples belong to the same class or category.

3.Are unregularized decision-trees prone to overfitting? If yes, why?

Ans - Overfitting happens when any learning processing overly optimizes training set error at the cost test error. While it's possible for training and testing to perform equality well in cross validation, it could be as the result of the data being very close in characteristics, which may not be a huge problem. In the case of decision tree's they can learn a training set to a point of high granularity that makes them easily overfit. Allowing a decision tree to split to a granular degree, is the behavior of this model that makes it prone to learning every point extremely well — to the point of perfect classification — ie: overfitting.

4.What is an ensemble technique in machine learning?

Ans - Ensemble methods are techniques that aim at improving the accuracy of results in models by combining multiple models instead of using a single model. The combined models increase the accuracy of the results significantly. This has boosted the popularity of ensemble methods in machine learning.

- Ensemble methods aim at improving predictability in models by combining several models to make one very reliable model.
- The most popular ensemble methods are boosting, bagging, and stacking.
- Ensemble methods are ideal for regression and classification, where they reduce bias and variance to boost the accuracy of models.
-

5. What is the difference between Bagging and Boosting techniques?

Ans - Bagging and boosting are different ensemble techniques that use multiple models to reduce error and optimize the model. The bagging technique combines

multiple models trained on different subsets of data, whereas boosting trains the model sequentially, focusing on the error made by the previous model.

- The bagging technique combines multiple models trained on different subsets of data, whereas boosting trains models sequentially, focusing on the error made by the previous model.
- Bagging is best for high variance and low bias models while boosting is effective when the model must be adaptive to errors, suitable for bias and variance errors.
- Generally, boosting techniques are not prone to overfitting. Still, it can be if the number of models or iterations is high, whereas the Bagging technique is less prone to overfitting.
- Bagging improves accuracy by reducing variance, whereas boosting achieves accuracy by reducing bias and variance.
- Boosting is suitable for bias and variance, while bagging is suitable for high-variance and low-bias models.

6. What is out-of-bag error in random forests?

Ans - While trying to make a better predictive model, we come across a famous ensemble technique in machine learning algorithms, known as Random Forest in Machine Learning. The Random Forest algorithm comes along with the concept of Out-of-Bag Score(OOB_Score).Random Forest, is a powerful ensemble technique for machine learning and data science, but most people tend to skip the concept of OOB_Score while learning about the algorithm and hence fail to understand the complete importance of Random forest as an ensemble method.

OOB (out-of-bag) errors are an estimate of the performance of a random forest classifier or regressor on unseen data. In scikit-learn, the OOB error can be obtained using the oob_score_ attribute of the random forest classifier or regressor.

The OOB error is computed using the samples that were not included in the training of the individual trees. This is different from the error computed using the usual training and validation sets, which are used to tune the hyperparameters of the random forest.

7. What is K-fold cross-validation?

Ans -K-fold cross validation in machine learning cross-validation is a powerful technique for evaluating predictive models in data science. It involves splitting the dataset into k subsets or folds, where each fold is used as the validation set in turn while the remaining k-1 folds are used for training. This process is repeated k times, and performance metrics such as accuracy, precision, and recall are computed for each fold. By averaging these metrics, we obtain an estimate of the model's generalization performance. This method is essential for model assessment, selection, and hyperparameter tuning, offering a reliable measure of a model's effectiveness. Compared to leave-one-out cross-validation, which uses k equal to the number of samples, K-fold cross-validation is computationally efficient and widely used in practice.

8. **What is hyper parameter tuning in machine learning and why it is done?**

Ans - Hyperparameter tuning is an essential part of controlling the behavior of a machine learning model. If we don't correctly tune our hyperparameters, our estimated model parameters produce suboptimal results, as they don't minimize the loss function. This means our model makes more errors. In practice, key indicators like the accuracy or the confusion matrix will be worse.

Hyperparameter tuning consists of finding a set of optimal hyperparameter values for a learning algorithm while applying this optimized algorithm to any data set. That combination of hyperparameters maximizes the model's performance, minimizing a predefined loss function to produce better results with fewer errors. Note that the learning algorithm optimizes the loss based on the input data and tries to find an optimal solution within the given setting. However, hyperparameters describe this setting exactly.

9. . **What issues can occur if we have a large learning rate in Gradient Descent?**

Ans - The learning rate is an important hyperparameter that greatly affects the performance of gradient descent. It determines how quickly or slowly our model learns, and it plays an important role in controlling both convergence and divergence of the algorithm. When the learning rate is too large, gradient descent can suffer from divergence. This means that weights increase exponentially, resulting in exploding gradients which can cause problems such as instabilities and overly high loss values. On the other hand, if the learning rate is too small, then gradient descent can suffer from slow convergence or even stagnation—which means it may not reach a local minimum at all unless many iterations are performed on large datasets.

In order to avoid these issues with different learning rates for each parameter/variable, we use adaptive techniques such as Ad grad and Adam which adjust their own learning rates throughout training based on real-time observations of parameters during optimization (i.e., they control exploration/exploitation trade-offs). These adaptive measures ensure better results than standard gradient descent while avoiding potential

pitfalls in terms of either massive gains or slow losses due to misconfigured static global learning rates like those used with traditional gradient descent algorithms.

10**. Can we use Logistic Regression for classification of Non-Linear Data? If not, why?**

Ans - Logistic regression is known and used as a linear classifier. It is used to come up with a hyper*plane* in feature space to separate observations that belong to a class from all the other observations that do *not* belong to that class. The decision boundary is thus *linear*. Robust and efficient implementations are readily available (e.g. scikit-learn) to use logistic regression as a linear classifier.

While logistic regression makes core assumptions about the observations such as IID (each observation is independent of the others and they all have an identical probability distribution), the use of a linear decision boundary is *not* one of them. The linear decision boundary is used for reasons of simplicity following the Zen mantra – when in doubt simplify. In those cases where we suspect the decision boundary to be nonlinear, it may make sense to formulate logistic regression with a nonlinear model and evaluate how much better we can do.

11 . **Differentiate between Adaboost and Gradient Boosting.**

Ans - Adaboost and gradient boosting are types of ensemble techniques applied in machine learning to enhance the efficacy of week learners. The concept of boosting algorithm is to crack predictors successively, where every subsequent model tries to fix the flaws of its predecessor. Boosting combines many simple models into a single composite one. By attempting many simple techniques, the entire model becomes a strong one, and the combined simple models are called week learners. So the adaptive boosting and gradient boosting increases the efficacies of these simple model to bring out a massive performance in the machine learning algorithm.

Features	Gradient boosting	Adaboost
Model	It identifies complex observations by huge residuals calculated in prior iterations	The shift is made by up-weighting the observations that are miscalculated prior
Trees	The trees with week learners are constructed using a greedy algorithm based on split points and purity scores. The trees are grown deeper with eight to thirty-two terminal nodes. The week learners should stay a week in terms of nodes, layers, leaf nodes, and splits	The trees are called decision stumps.
Classifier	The classifiers are weighted precisely and their prediction capacity is constrained to learning rate and increasing accuracy	Every classifier has different weight assumptions to its final prediction that depend on the performance.
Prediction	It develops a tree with help of previous classifier residuals by capturing variances in data.	It gives values to classifiers by observing determined variance with data. Here all the week learners possess equal weight and it is usually

	The final prediction depends on the maximum vote of the week learners and is weighted by its accuracy.	fixed as the rate for learning which is too minimum in magnitude.
Short-comings	Here, the gradients themselves identify the shortcomings.	Maximum weighted data points are used to identify the shortcomings.
Loss value	Gradient boosting cut down the error components to provide clear explanations and its concepts are easier to adapt and understand	The exponential loss provides maximum weights for the samples which are fitted in worse conditions.
Applications	This method trains the learners and depends on reducing the loss functions of that week learner by training the residues of the model	Its focus on training the prior miscalculated observations and it alters the distribution of the dataset to enhance the weight on sample values which are hard for classification

12. **What is bias-variance trade off in machine learning?**

Ans - The bias is known as the difference between the prediction of the values by the Machine Learning model and the correct value. Being high in biasing gives a large error in training as well as testing data. It recommended that an algorithm should always be low-biased to avoid the problem of underfitting. By high bias, the data predicted is in a straight line format, thus not fitting accurately in the data in the data set. Such fitting is known as the **Underfitting of Data**

If the algorithm is too simple (hypothesis with linear equation) then it may be on high bias and low variance condition and thus is error-prone. If algorithms fit too complex (hypothesis with high degree equation) then it may be on high variance and low bias. In the latter condition, the new entries will not perform well. Well, there is something between both of these conditions, known as a Trade-off or Bias Variance Trade-off. This tradeoff in complexity is why there is a tradeoff between bias and variance. An algorithm can't be more complex and less complex at the same time.

13. . **Give short description each of Linear, RBF, Polynomial kernels used in SVM.**

Ans -Linear Kernel

A linear kernel is a type of kernel function used in machine learning, including in SVMs (Support Vector Machines). It is the simplest and most commonly used kernel function, and it defines the dot product between the input vectors in the original feature space.

The linear kernel can be defined as:

1. K(x, y) = x .y

Where x and y are the input feature vectors. The dot product of the input vectors is a measure of their similarity or distance in the original feature space.

When using a linear kernel in an SVM, the decision boundary is a linear hyperplane that separates the different classes in the feature space. This linear boundary can be useful when the data is already separable by a linear decision boundary or when dealing with high-dimensional data, where the use of more complex kernel functions may lead to overfitting.

Polynomial Kernel

A particular kind of kernel function utilised in machine learning, such as in SVMs, is a polynomial kernel (Support Vector Machines). It is a nonlinear kernel function that employs polynomial functions to transfer the input data into a higher-dimensional feature space.

One definition of the polynomial kernel is:

Where x and y are the input feature vectors, c is a constant term, and d is the degree of the polynomial, K(x, y) = (x. y + c)d. The constant term is added to, and the dot product of the input vectors elevated to the degree of the polynomial.

The decision boundary of an SVM with a polynomial kernel might capture more intricate correlations between the input characteristics because it is a nonlinear hyperplane.

The degree of nonlinearity in the decision boundary is determined by the degree of the polynomial.

The polynomial kernel has the benefit of being able to detect both linear and nonlinear correlations in the data. It can be difficult to select the proper degree of the polynomial, though, as a larger degree can result in overfitting while a lower degree cannot adequately represent the underlying relationships in the data.

In general, the polynomial kernel is an effective tool for converting the input data into a higher-dimensional feature space in order to capture nonlinear correlations between the input characteristics.

Gaussian (RBF) Kernel

The Gaussian kernel, also known as the radial basis function (RBF) kernel, is a popular kernel function used in machine learning, particularly in SVMs (Support Vector Machines). It is a nonlinear kernel function that maps the input data into a higher-dimensional feature space using a Gaussian function.

The Gaussian kernel can be defined as:

1. K(x, y) = exp(-gamma * ||x - y||^2)

Where x and y are the input feature vectors, gamma is a parameter that controls the width of the Gaussian function, and ||x - y||^2 is the squared Euclidean distance between the input vectors.

When using a Gaussian kernel in an SVM, the decision boundary is a nonlinear hyper plane that can capture complex nonlinear relationships between the input features. The width of the Gaussian function, controlled by the gamma parameter, determines the degree of nonlinearity in the decision boundary.

One advantage of the Gaussian kernel is its ability to capture complex relationships in the data without the need for explicit feature engineering. However, the choice of the gamma parameter can be challenging, as a smaller value may result in under fitting, while a larger value may result in over fitting.