1) **R-squared or Residual Sum of Squares (RSS) which one of these two is a better measure ofgoodness of fit model in regression and why?**

<u>**Ans:**</u>-

R-squared and Residual sum of squares (RSS)are both commonly used measures of goodness of fit for regression models, but they serve different purposes and cannot be directly compared.

R-squared is a statistical measure that represents the proportion of variance in the dependent variable that is explained by independent variable(s) in the model. It ranges from 0 to 1, with higher values indicating a better fit. R-squared is a popular measure because it is easy to interpret and compare across different models, and it can be used to assess the overall fit of model.

On the other hand RSS measures the sum of squared residuals (the different between the predicted values and actual values)in the model. It is a measure of the amount of unexplained variance in the dependent variable, and lower values indicate a better fit. RSS is useful for evaluating the performance of the model in terms of the accuracy of its predictions.

In general , R-squared is a better measure of overall goodness of fit, while RSS is a better measure of the accuracy of model's predictions.

However, it is important to note that R-squared can be misleading if the model is overfitting the date or if it has included irrelevant independent variables. In such cases, RSS may provide a more accurate assessment of the model's performance, therefore it is often useful to consider both measures in combination when evaluating a regression model.

2) **What are TSS (Total Sum of Squares), ESS (Explained Sum of Squares) and RSS (Residual Sumof Squares) in regression. Also mention the equation relating these three metrics with each other.**

<u>**Ans:-**</u>

In regression analysis, TSS(total sum of squares),ESS(Explained sum of squares),and RSS(Residual sum of squares) are used to evaluate the goodness of fit of a regression model.

TSS represents the total variation in the dependent variable(y) and can be calculated as the sum of squares of the difference between each observed value and the mean of y:

$TSS = \Sigma(y_i - \bar{y})^2$

ESS represents the variation in y that can be explained by the independent variable(x) included in the model. ESS can be calculated as the sum of squares of the different between the predicted value of y($\hat{y}$)

And the mean of y:

$ESS = \Sigma(\hat{y}_i - \bar{y})^2$

RSS represents the variation in y that cannot be explained by the independent variables and is due to random error. RSS can be calculated as the sum of squares of the difference between the observed value of y(y_i) and the predicted value of y($\hat{y}_i$):

$RSS = \Sigma(y_i - \hat{y}_i)^2$

The relationship between these three metrics is given by:

TSS = ESS + RSS

In other words, the total sum of squares(TSS) can be partitioned into the explained sum of squares(ESS) and the residual sum of squares(RSS). The ratio of ESS to TSS is often used as a measure of the proportion of variation in y that can be explained by the model, and is called the co-efficient of determination or R-Squared.

3) What is the need of regularization in machine learning?

Regularization is a technique used in machine learning to prevent overfitting of the model to the training data. Overfitting occurs when the model captures noise in the training data and performs poorly on unseen data. Regularization helps to reduce overfitting by adding a penalty term to the objective function of the model, which discourages the model from learning complex relationships in the data that may not be generalizable.

The need of regularization arises when the model has high variance , meaning that it is sensitive to small functions in the training data and can fit noise in the data. Regularization techniques such as L1 regularization (LASSO) and L2 regularization (RIDGE) can help to reduce the variance of the model and improve its generalization performance.

Regularization is particularly important when the number of features in the data is large, as models with many features are more prone to overfitting. Regularization can also be useful in cases where the data is noisy or when the model has limited training data.

Overall, regularization is a key tool in machine learning for improving the performance and robustness of models, and it is widely used in many applications, such as image recognition, natural language processing and predictive modelling.

4) What is Gini–impurity index?

The Gini-impurity index is a measure of impurity or diversity used in decision trees and other machine learning models. It measures the probability of mis classifying a randomly chosen element in a dataset if it were labeled randomly according to the class distribution of the dataset.

The Gini impurity index is calculated as follows:-

1) Compute the probability p(i) of an item in the dataset belonging to a particular class i.
2) Calculate the Gini impurity index Gini(D) for the entire dataset D as follows:

   Gini(D)=1-sum(p(i)^2) for all classes in D.

   The Gini impurity index ranges from 0 to 1, with 0 indicating a pure dataset in which all elements belong to the same class, and 1 indicating a dataset in which elements are equally distributed among all classes. Decision tree algorithms use the Gini impurity index to determine which attribute to split on at each node of the tree in order to minimize the impurity or increase the purity of the resulting subsets.

5) Are unregularized decision-trees prone to overfitting? If yes, why?

Yes, unregularized decision trees are prone to overfitting. This is because decision trees have a tendency to create complex trees that can perfectly fit the training data, but may not generalize well to new, unseen data.

An, unregularized decision tree can keep splitting nodes until each leaf contains only one instance, resulting in a tree that is highly specific to the training data. This can cause the tree to capture noise and outliers in the data, leading to poor performance on new data.

Overfitting can also occur when there are two many features or when the data contains many irrelevant or redundant features. In this case, the decision tree can create splits based on noise or spurious correlations rather than meaningful relationships between features and the target variable.

Regularization techniques such as pruning, reducing the depth of the tree, limiting the number of leaf nodes, and using feature selection or extraction can help prevent overfitting in decision trees. By simplifying the tree and reduces its complexity, these techniques can improve the tree’s ability to generalize to new data.

6) What is an ensemble technique in machine learning?

An ensemble technique in machine learning refers to a method that combines multiple models or algorithms to improve the overall performance of a single model. The idea behind an ensemble technique is to leverage the strengths and weakness of different models by combining them in a way that produces a more accurate prediction.

Ensemble techniques are particularly useful when working with complex datasets that have a large no of features or when the data is noisy or contains outliers. The most common types of ensemble techniques include:

1.**Bagging(Boot strap aggregating):-**

This method involves training multiple instances of the same model on diff, subsets of training data, and combining their predictions. The goal is to reduce variance and improve model stability.

2.**Boosting:-**

This technique involves sequentially training weak models and combining their predictions. The goal is to reduce bias and improve model accuracy .

3.Stacking:-

This method involves training multiple models and using their predictions as input to a higher-level model. The goal is to combine the strengths of different models and produce a more accurate prediction.

Ensemble techniques have been shown to be effective in a wide range of applications, including image classification natural language processing, and recommender systems.

7) What is the difference between Bagging and Boosting techniques?

Bagging and boosting are both ensemble technique used in machine learning to improve the accuracy of models by combining multiple weaker models to create a strong model.

However, there are some key differences between Bagging and Boosting:

1.Method:-

Bagging is parallel ensemble method, while boosting is a sequential ensemble method.

2.Sampling Technique:-

In Bagging, multiple subsets of the data are created using bootstrapping and each subset is used to train a sperate model. In Boosting, each model is trained on a modified version of the data where the miss classified data points from the previous model are given more weight.

3.Model Weighting:-

In Bagging, all the models have equal weighting in the final prediction, while in boosting, the models are weighted based on their performance on the training data.

4.Model Types: -

Bagging can be used with any type of model, while boosting is typically used with decision trees.

5.Bias variance trade off: -

Bagging helps reduce the variance of the model by reducing overfitting, while boosting helps reduce bias by focusing on the misclassified data points.

6.Performance: -

Bagging typically results in a lower variance and slightly better accuracy than boosting, while boosting often results in lower bias and better performance on the test data.

In summary, Bagging and Boosting are two different ensemble techniques with different methods, sampling techniques model weighting, and performance characteristics. Bagging is a parallel ensemble method that reduces variance, while Boosting is a sequential ensemble method that reduce bias.

8) What is out-of-bag error in random forests?

In random forest, the out-of-bag (OOB) error is an estimate of the generalization error of the model. It is calculated by using the instances in the training dataset that were not included in the bootstrap sample used to train each tree in the forest.

The OOB error is a convenient way to estimate the performance of a random forest model without the need for a separate validation dataset. For each instance in the training dataset, the OOB error is calculated by aggregating the predictions made by all the trees in the forest that did not include that instance in their bootstrap sample.

The OOB error can be used to tune hyperparameters of the random forest model, such as the number of trees, and to compare the performance of different models. It is important to note that the OOB error of the model. Therefore, it is still recommended to evaluate the model on a separate validation set to ensure its performance on unseen data.

9) What is k-fold cross-validation?

K-fold cross-validation is a technique used in machine learning for model evaluation and hyperparameter tuning. It involves splitting the dataset into k equally sized folds or partitions, where K is a user specific integer. The model is then trained and evaluated K times, each time using a different fold as the validation set and the remaining folds as the training set.

<u>The general steps of K fold cross-validation are as follows</u>:-

1. Split the dataset into K equally sized folds.
2. For each fold, train the model on the remaining K-1 fold and evaluate its performance on the current fold(i.e the validation set).
3. Calculate the average the performance across all K iterations as the final evaluation metric.

K- fold cross- validation is particularly useful in situations where the dataset is small, and we need to make the most of the available data. It also helps to reduce the risk of overfitting and provides a more robust estimate of model performance.

Some variation of K-fold cross-validation includes stratified k-fold cross-validation, which preserves the class distribution of the data in each fold, and nested cross-validation, which uses an inner loop to tune the hyperparameters of the model and an outer loop to evaluate its performance.

10) What is Hyper parameter tuning in machine learning and why is done?

Hyper parameter tuning is the process of selecting the optimal set of hyperparameters for a machine learning algorithm. Hyper parameters are parameters of machine learning algorithm that are not learned during training, but are instead set before the training process begins. Example of hyperparameters include learning rate, regularization strength, and the number of hidden layers in a neural network.

Hyper parameter tuning is done to improve the performance of machine learning model. By choosing the best hyperparameters, the model can generalize better of unseen data, reduce overfitting, and improves it overall accuracy. However, since hyperparameters are not learned from the data, they need to be selected carefully by the machine learning engineer or data scientist, based on their domain knowledge and intuition.

Hyper parameter tuning can be done manually or using automated methods such as grid search, random search, and Bayesian optimization. In manual tuning, the engineer tries out different hyper parameters and evaluates the performance of the model on a validation set. In automated tuning, a search algorithm is used to explore the hyperparameter space and find the optimal set of hyperparameters.

Hyper parameters tuning is an important step in the machine learning workflow, as it can have a significant impact on the performance of the model.

11) What issues can occur if we have a large learning rate in gradient Descent?

In Gradient Descent, the learning rate determines the step size that the algorithm takes in each iteration towards the optimal solution. If the learning rate is too large, the algorithm many fail to converge to the optimal solution, and may oscillate back an forth between different areas of the parameter space. This can cause several issues, including:-

1.**Overshooting the minimum: -**

A large learning rate may cause the algorithm to overshoot the minimum of the cost functions, resulting in the algorithm bouncing back and forth across the minimum without ever reaching it.

2.**Slow convergence: -**

A large rate may cause the algorithm to Oscillate between two parts, leading to slower convergence to the optimal solution.

3.Divergence: -

A very large learning rate can cause the algorithm to diverge and move away from the optimal solution.

4.Unstable Behaviour: -

A very large learning rate may cause the cost function to fluctuate widely, making it difficult for the algorithm to converge to a stable solution.

To avoid these issues, it is important to choose an appropriate learning rate that balances convergence speed with stability. This can be done through hyperparameter tuning which involves experimenting with different learning grates and choosing the one that gives the best performance on the validation set. Alternatively, adaptive learning rate algorithm like AdaGrad, RMSProp, and Adam can be used to adjust the learning rate dynamically during training.

12) Can we use logistic Regression for classification of NoN-linear Data? If not why?

Logistic Regression is a linear model and works well when the relationship between the input features and the output variable is linear. However if the relationship between the input features and the output variable is non-linear, Logistic Regression may not be an appropriate model. In such cases, the decision boundary of Logistic Regression will be a linear function and may not capture the non-linear relationship between the input features and the output variable.

To handle non-linear data, one can use non-linear models such as decision trees, random forests, support vector machines(SVMs), or neural networks. These models can capture non-linear relationships between the input features and the output variable by using non-linear decision boundaries.

In summary, Logistic Regression is not suitable for classification of non-linear data as it assumes a linear relationship between the input features and the output variable. Non-linear models should be used instead to capture non-linear relationships.

13) Difference between Adaboost and Gradient Boosting?

AdaBoost and Gradient boosting are two popular machine learning algorithms used of building ensemble methods. Here are the difference between them:

<u>Approach: -</u>

AdaBoost is a boosting algorithm that combines multiple weak learners to create a strong learner, while Gradient Boosting is boosting algorithm that iteratively trains decision trees to reduce the residual error.

<u>Weighting:</u>

AdaBoost assigns weights to each data point based on its classification accuracy, and focuses on the misclassified samples to improve the model. In contrast, Gradient Boosting assigns weights to the residual of the model and focuses on reducing the error in the model.

<u>Sequential Vs Parallel</u>:-

AdaBoost builds multiple weak models sequentially, with each model learning from mistakes of the previous model. In contrast, Gradient Boosting builds decision trees in parallel, with each tree learning from the error to the previous trees.

<u>Overfitting: -</u>

AdaBoost can be prone to overfitting if the weak learners use too complex or if there is large amount of noise in the data. Gradient Boosting uses techniques like regularization to prevent Overfitting.

Model complexity: -

AdaBoost uses simple models(weak learners)such as decision stamps, while Gradient Boosting uses more complex models(decision trees) that can capture non-linear relationships in the data.

In summary, Adaboost and Gradient Boosting are both powerful ensemble algorithms used for classification and regression tasks, but difference in their approach, weighting, sequential vs parallel building of models, susceptibility to overfitting, and model complexity.

14. **What is bias- variance trade off in Machine – Learning?**

The Bias-Variance trade-off is a functional concept in machine learning that deals with the relationship between the model's ability to fit the training data(bias) and its ability to generalize to unseen data(variance)

Bias refers to difference between the expected predictions of the model and the true values of the data. A model with high bias has a limited capacity to fit the training data, and may underfit, resulting in poor performance on both training and test data.

Variance, on the other hand, refers to the amount by which the model's prediction changes when trained on different subset of the data. A model with high variance is too complex and can overfit to the training data resulting in high accuracy on the training data but poor performance on the test data.

The goal of a machine learning model is to find the right balance between bias and variance that minimize the overall error. A model that is too simple may have high bias and low variance. While a model that is too complex may have low bias and high variance.

To achieve the right balance several techniques can be used, including, regularization, cross- validation and ensembling. Regularization can be used to reduce the complexity of the model, while cross- validation can be used to estimate the model's performance on unseen data. Ensembling can be used to combine multiple models to reduce the overall variance.

In summary, the bias- variance trade off is an important concept in machine learning that highlight the need to find the right balance between underfitting and overfitting to achieve optimal model performance.

15. Give short description each of linear, RBF, Polynomial kernels used in SVM

SVM(Support vector machine) is a popular supervised learning algorithm used for classification and regression analysis. The kernel functions are essential components of SVM as they are used to transform the input data into a higher-dimensional space where the problem of classification or regression can be more easily solved.

Here are short descriptions of some of the commonly used kernels in SVM:

Linear kernel:-

The linear kernel is a simple kernel that calculates the dot product between the input feature vectors. This kernel is suitable for linearly separable datasets, where the decision boundary is a straight line.

RBF:-

The RBF kernel is a popular kernel used in SVM for non-linearly separable datasets. It maps the input data to an infinite- dimensional space, making it possible to classify non linearly sparable data. The RBF kernel uses a Gaussian function to compute the similarity between the input features vectors.

Polynomial Kernel:-

The polynomial kernel is used to solve non-linear classification problems. It maps the input data to a higher dimensional space, which makes it easier to find a decision boundary. The polynomial kernel function computes the dot product of the feature vectors in a higher dimensional space using a polynomial function.

These kernels can be tuned to optimize the performance of the SVM on different datasets. Choosing the appropriate kernel is important as it can greatly affect the accuracy of the model.