MACHINE LEARNING

Q1 to Q15 are subjective answer type questions, Answer them briefly.

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?**

Ans. R-squared is a better measure of goodness of fit than Residual Sum of Squares (RSS) in regression because it provides a standardized measure of the proportion of the variance in the dependent variable that is explained by the independent variable(s).

R-squared is a statistical measure that ranges from 0 to 1, where 0 indicates that the model explains none of the variance in the dependent variable and 1 indicates that the model explains all of the variance. R-squared is calculated as the ratio of the explained variance to the total variance in the dependent variable. The explained variance is the difference between the total variance and the residual variance, which is the variance that is not explained by the independent variable(s).

RSS, on the other hand, is the sum of the squared residuals, which are the differences between the observed values and the predicted values of the dependent variable. RSS measures the total amount of unexplained variance in the dependent variable, but it does not provide a standardized measure of the goodness of fit, as it depends on the scale of the dependent variable.

Therefore, R-squared is a better measure of goodness of fit than RSS because it provides a standardized measure of the proportion of variance explained by the independent variable(s), which allows for comparison across different models and datasets, and it is independent of the scale of the dependent variable.

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.**

Ans. In regression analysis, TSS (Total Sum of Squares), ESS (Explained Sum of Squares), and RSS (Residual Sum of Squares) are three important measures of the variability in the dependent variable (y).
TSS represents the total variability in the dependent variable, and it is calculated as the sum of the squared differences between each observed value of y and the mean of y:

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

ESS represents the variability in the dependent variable that is explained by the independent variable(s), and it is calculated as the sum of the squared differences between the predicted values of y and the mean of y:

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

RSS represents the variability in the dependent variable that is not explained by the independent variable(s), and it is calculated as the sum of the squared differences between the observed values of y and the predicted values of y:

$RSS = \Sigma(y - yhat)^2$

The relationship between TSS, ESS, and RSS is given by the following equation:

$TSS = ESS + RSS$

This equation shows that the total variability in the dependent variable can be decomposed into two parts: the variability that is explained by the independent variable(s) (ESS) and the variability that is not explained by the independent variable(s) (RSS). The proportion of variability in the dependent variable that is explained by the independent variable(s) can be measured by the R-squared statistic, which is the ratio of ESS to TSS:

$R^2 = ESS / TSS$

The R-squared statistic measures the proportion of the total variability in the dependent variable that is explained by the independent variable(s). A high value of R-squared indicates a good fit between the regression model and the data, whereas a low value of R-squared indicates a poor fit.

3. **What is the need of regularization in machine learning?**

Ans. Regularization is an important technique in machine learning that is used to prevent overfitting and improve the generalization performance of a model. Overfitting occurs when a model is too complex and captures noise in the training data, which results in poor performance on new, unseen data. Regularization helps to prevent overfitting by adding a penalty term to the objective function that the model is trying to minimize.

The main need for regularization in machine learning is to improve the generalization performance of a model. Generalization refers to the ability of a model to perform well on new, unseen data. In many cases, a model may perform well on the training data but poorly on new data, which indicates that the model has overfit the training data. Regularization helps to reduce the complexity of the model and prevent overfitting, which in turn improves the generalization performance of the model.

Regularization also helps to improve the stability and interpretability of the model. A highly complex model may be difficult to interpret and may have unstable coefficients, which can make it difficult to make predictions or draw conclusions from the model. Regularization helps to reduce the complexity of the model and stabilize the coefficients, which makes it easier to interpret the model and make predictions.

There are different types of regularization techniques, such as L1 regularization (also known as Lasso regularization), L2 regularization (also known as Ridge regularization), and Elastic Net regularization. These techniques add a penalty term to the objective function that the model is trying to minimize, which helps to reduce the complexity of the model and prevent overfitting. The choice of regularization technique and the strength of the penalty term depend on the specific problem and the characteristics of the data**.**

4. **What is Gini–impurity index?**

Ans. The Gini impurity index is a measure of the impurity or diversity of a set of examples in a classification problem. It is commonly used in decision trees and random forests to evaluate the quality of a split when building a tree.

The Gini impurity index ranges from 0 to 1, where 0 represents a pure set (all examples belong to the same class) and 1 represents a completely impure set (an equal number of examples in each class). The Gini impurity index is calculated as follows:

Gini impurity = 1 - ∑(p_i)^2

where p_i is the proportion of examples in the i-th class.

When building a decision tree, the Gini impurity index is used to evaluate the quality of each split. A split with a lower Gini impurity index is considered to be better, as it results in a more pure set of examples in each of the resulting subsets. The Gini impurity index is also used in random forests to determine the importance of each feature in the classification task.

In summary, the Gini impurity index is a measure of the diversity or impurity of a set of examples in a classification problem, and it is commonly used in decision trees and random forests to evaluate the quality of a split and determine the importance of each feature**.**

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

Ans. Yes, unregularized decision trees are prone to overfitting. Decision trees are non-parametric models, meaning they have no fixed number of parameters that are determined by the data. Instead, they can adjust themselves to the training data by creating a tree structure that partitions the data into smaller and smaller subsets, each with its own decision rule.

Without any regularization techniques, decision trees will continue to grow and become more complex until they fit the training data perfectly. This can lead to overfitting, where the model performs well on the training data but poorly on new, unseen data.

Overfitting occurs because the decision tree has learned the noise in the training data as well as the signal. As a result, it becomes too specific to the training data and does not generalize well to new data. Regularization techniques such as pruning, setting a minimum number of samples per leaf, and limiting the maximum depth of the tree can help prevent overfitting in decision trees.

6. **What is an ensemble technique in machine learning?**

Ans. An ensemble technique in machine learning is a method that combines multiple models to improve the overall performance and accuracy of the prediction. Ensemble techniques are based on the idea that by combining several models, the resulting model will perform better than any individual model.

The most common ensemble techniques in machine learning are:

Bagging (Bootstrap Aggregating): In bagging, multiple models are trained on different subsets of the training data, where each subset is sampled with replacement (bootstrap sampling). The outputs of these models are then aggregated to produce a final prediction. This technique reduces variance and overfitting and is used for decision trees and other unstable models.

Boosting: Boosting is an iterative technique that creates multiple models sequentially, where each new model focuses on correcting the errors made by the previous model. It assigns higher weights to misclassified samples and lower weights to correctly classified samples, such that the next model focuses more on the misclassified samples.

Stacking: Stacking combines multiple models with different strengths and weaknesses. In stacking, the outputs of several models are used as input to a meta-model that learns how to best combine them.

Ensemble techniques are often used in machine learning to improve the accuracy and stability of predictions, reduce overfitting, and handle data imbalance or noise. They have been used successfully in various applications, including image classification, natural language **processing, and** recommender systems.

7. **What is the difference between Bagging and Boosting techniques?**

Ans. The main difference between Bagging and Boosting techniques in machine learning is the way they combine multiple models to improve prediction accuracy.

Bagging (Bootstrap Aggregating) and Boosting are both ensemble techniques, which means they use multiple models to make predictions. However, they differ in the way they train and combine these models:

Bagging: Bagging is a parallel ensemble technique that involves training multiple models independently on different subsets of the training data. These subsets are sampled with replacement using bootstrap sampling. Each model produces a prediction, and the final prediction is made by aggregating the predictions of all the models. Bagging is mainly used to reduce variance and overfitting.

Boosting: Boosting is a sequential ensemble technique that involves training multiple models iteratively, where each model focuses on correcting the errors made by the previous model. It assigns higher weights to misclassified samples and lower weights to correctly classified samples, such that the next model focuses more on the misclassified samples. The final prediction is made by aggregating the predictions of all the models. Boosting is mainly used to reduce bias and improve accuracy.

In summary, Bagging trains multiple models in parallel, while Boosting trains multiple models sequentially. Bagging reduces variance and overfitting, while Boosting reduces bias and improves accuracy. Both Bagging and Boosting can be used with various machine learning algorithms, such as decision trees, random forests, and neural networks**.**

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

Ans. Out-of-bag (OOB) error in random forests is a technique used to estimate the performance of a random forest model on new, unseen data without the need for a separate validation set.

In a random forest, each tree is built using a bootstrap sample of the training data, which means that some samples are left out of each tree's training set. These left-out samples are called out-of-bag samples.

The OOB error is then calculated by evaluating the predictions of each tree on its corresponding out-of-bag samples. The OOB error is the average error rate across all trees in the forest, and it can be used as an estimate of the generalization error of the random forest model on new, unseen data.

The advantage of using OOB error estimation is that it allows for the evaluation of model performance without the need for a separate validation set, which can be especially useful when the dataset is small or when a separate validation set is not available. It also makes the random forest algorithm computationally efficient as the algorithm does not need to split the data into separate training and validation sets.

9. **What is K-fold cross-validation?**

Ans. K-fold cross-validation is a technique used to evaluate the performance of a machine learning model by dividing the dataset into K equally sized subsets or folds. The model is trained on K-1 of these folds, and the remaining fold is used for validation. This process is repeated K times, with each fold being used once as the validation set.

The general steps involved in K-fold cross-validation are as follows:

Split the dataset into K equally sized folds.
For each fold:
Use K-1 folds for training the model.
Use the remaining fold for validation.
Calculate the evaluation metric(s) (e.g., accuracy, precision, recall) for the model on the validation set.
Calculate the average of the evaluation metric(s) across all K folds.
The advantage of using K-fold cross-validation is that it provides a more reliable estimate of model performance than a single train-test split because the evaluation is performed multiple times. It also ensures that the model is trained on all the data available and is not biased towards a specific subset.

The choice of K depends on the size of the dataset, with a common choice being K=10 for smaller datasets and K=5 for larger datasets. However, the choice of K should also take into account the computational resources available, as running K-fold cross-validation can be computationally expensive for large datasets or complex models.

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

Ans. Hyperparameter tuning in machine learning is the process of selecting the optimal hyperparameters for a given model to achieve the best performance on a specific dataset.

Hyperparameters are parameters of a machine learning algorithm that are set before the learning process begins and cannot be learned directly from the data. Examples of hyperparameters include the learning rate in a neural network, the number of trees in a random forest, or the regularization parameter in a logistic regression model.

Hyperparameter tuning is done because the performance of a machine learning model depends heavily on the choice of hyperparameters. Choosing suboptimal hyperparameters can lead to poor model performance, including overfitting or underfitting the data. On the other hand, selecting optimal hyperparameters can lead to improved model performance, higher accuracy, and better generalization.

Hyperparameter tuning can be done using several techniques, such as grid search, random search, or Bayesian optimization. In grid search, a predefined set of hyperparameters is used, and the model is trained and evaluated for each combination of hyperparameters in the set. In random search, hyperparameters are sampled randomly from a predefined distribution, and the model is trained and evaluated for each set of hyperparameters. Bayesian optimization uses probabilistic models to guide the search for optimal hyperparameters.

Hyperparameter tuning is an important step in machine learning model development, as it can significantly impact the performance and generalization of the model.

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

Ans. If we have a large learning rate in gradient descent, it can lead to several issues:

Divergence: A large learning rate can cause the gradient descent algorithm to overshoot the optimal values, leading to divergence instead of convergence. This means that the loss function will continue to increase rather than decrease.

Slow convergence: A large learning rate can cause the gradient descent algorithm to oscillate back and forth around the optimal values, leading to slow convergence. This happens because the algorithm is unable to find the optimal values and keeps bouncing around them.

Instability: A large learning rate can cause the gradient descent algorithm to become unstable and unpredictable, making it difficult to determine the optimal values.

Overshooting the minimum: With a large learning rate, the gradient descent algorithm may take steps that are too large, overshooting the minimum of the cost function. This means that the algorithm may never converge to the minimum, and instead may oscillate around it.

In summary, a large learning rate in gradient descent can lead to instability, slow convergence, overshooting the minimum, and divergence, making it difficult to find the optimal parameters for a machine learning model. It is important to choose an appropriate learning rate that balances the speed of convergence and stability of the algorithm.

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

Ans. Logistic regression is a linear classification algorithm and is best suited for datasets where the decision boundary between classes is linear. Therefore, it may not be the best choice for classifying nonlinear data.

If the decision boundary between classes is nonlinear, using logistic regression may lead to underfitting of the data, which means that the model will not be able to capture the underlying patterns in the data. This is because logistic regression models use a linear decision boundary, which is unable to capture the nonlinear relationships between the input features and the output class.

However, there are techniques that can be used to transform the data to make it more linearly separable, such as polynomial feature engineering. In this approach, the input features are transformed by creating polynomial combinations of the original features. This can make the decision boundary nonlinear in the transformed feature space, which can lead to improved performance of logistic regression.

Alternatively, other classification algorithms that are better suited for nonlinear data, such as decision trees, random forests, and support vector machines (SVMs), can be used instead of logistic regression. These algorithms are able to model nonlinear relationships between the input features and the output class and are more flexible than logistic regression in handling complex data.

13. Differentiate between Adaboost and Gradient Boosting.

Ans. Adaboost and Gradient Boosting are both ensemble learning techniques used in machine learning for improving the performance of weak learners. However, they differ in their approach to achieving this goal:

Adaboost (Adaptive Boosting): In Adaboost, a series of weak learners are trained sequentially, with each subsequent learner focusing on the misclassified samples of the previous learner. During each iteration, the algorithm adjusts the weights of the misclassified samples, giving more importance to those samples that were classified incorrectly. This process is repeated until a specified number of weak learners are trained or until the desired accuracy is achieved.

Gradient Boosting: In Gradient Boosting, a series of weak learners are also trained sequentially, but in contrast to Adaboost, each subsequent learner focuses on the residuals (the difference between the actual and predicted values) of the previous learner. During each iteration, the algorithm fits a weak learner to the residuals of the previous learner, which effectively reduces the error of the previous model. This process is repeated until a specified number of weak learners are trained or until the desired accuracy is achieved.

In summary, Adaboost and Gradient Boosting are both ensemble learning techniques that sequentially train a series of weak learners. However, Adaboost focuses on the misclassified samples of the previous learner, while Gradient Boosting focuses on the residuals of the previous learner.

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

Ans. Bias-variance tradeoff is a fundamental concept in machine learning that refers to the tradeoff between a model's ability to fit the training data (low bias) and its ability to generalize to new, unseen data (low variance).

Bias refers to the degree to which a model's predictions differ from the true values, even when trained on the entire training dataset. A model with high bias may underfit the data, resulting in poor performance on both the training and test sets.

Variance refers to the degree to which a model's predictions change when trained on different subsets of the training data. A model with high variance may overfit the training data, capturing noise and random fluctuations in the data, and may perform poorly on new, unseen data.

The bias-variance tradeoff states that as we decrease the bias in a model (by increasing its complexity or flexibility), the variance typically increases, and vice versa. Thus, there is a tradeoff between bias and variance in machine learning models, and finding the optimal balance between the two is important for building accurate and generalizable models.

In summary, the bias-variance tradeoff is the balance between a model's ability to fit the training data and its ability to generalize to new, unseen data, and finding the optimal balance between bias and variance is a critical step in building effective machine learning models.

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

Ans. SVM (Support Vector Machine) is a popular machine learning algorithm used for classification and regression problems. The kernel function used in SVM plays a crucial role in its performance. Here is a short description of some commonly used kernel functions in SVM:

Linear kernel: A linear kernel is a simple, linear function that computes the dot product between two vectors. It is used when the data is linearly separable, meaning that a straight line can be drawn to separate the two classes.

RBF (Radial Basis Function) kernel: The RBF kernel is a nonlinear kernel that is commonly used in SVM for classifying nonlinear data. It maps the data to a high-dimensional feature space and computes the similarity between two samples based on their distance in this space. The RBF kernel has a hyperparameter called the gamma parameter, which determines the shape of the decision boundary.

Polynomial kernel: The polynomial kernel is a nonlinear kernel that maps the data to a high-dimensional feature space using polynomial combinations of the original features. It is used when the decision boundary between the two classes is nonlinear and can be approximated using a polynomial function. The polynomial kernel has a hyperparameter called the degree parameter, which determines the degree of the polynomial function used.

In summary, the linear kernel is used for linearly separable data, the RBF kernel is used for nonlinear data, and the polynomial kernel is used for data with a nonlinear decision boundary that can be approximated using a polynomial function.

FLIP ROBO