Machine Learning Assignment

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

Ans.) R-squared is a better measure of goodness of fit for a regression model because it provides a relative measure of how well the model fits the data. R-squared ranges from 0 to 1, where 1 indicates a perfect fit and 0 indicates a poor fit. The Residual Sum of Squares (RSS) is the sum of the squared differences between the predicted and actual values, and it does not provide a relative measure of the model's fit to the data. It can be used to compare the fit of different models, but it doesn't provide information on how well the model fits the data.

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

Ans.) TSS (Total Sum of Squares) is the sum of the squared differences between the actual values of the response variable and its mean. It represents the total variation in the response variable.

ESS (Explained Sum of Squares) is the sum of the squared differences between the predicted values and the mean of the response variable. It represents the variation in the response variable explained by the model's predictor variables.

RSS (Residual Sum of Squares) is the sum of the squared differences between the predicted values and the actual values of the response variable. It represents the variation in the response variable that is not explained by the predictor variables in the model.

The following equation can represent the relationship between these three metrics:

TSS = ESS + RSS

This equation states that the total variation in the response variable can be broken down into the variation explained by the predictor variables (ESS) and the variation not explained by the predictor variables (RSS).

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

Ans.)

- Regularization is a technique used to prevent overfitting in machine learning models.
- Overfitting occurs when a model is too complex and fits the noise or random fluctuations in the training data, leading to poor performance on new unseen data.
- Regularization works by adding a penalty term to the loss function being optimized, which discourages certain model parameters from becoming too large.
- There are different types of regularization, such as L1 and L2 regularization.
- L1 regularization adds a penalty term to the loss function that is proportional to the absolute value of the model parameters. In contrast, L2 regularization adds a penalty term proportional to the model parameters' square.
- The need for regularization is to balance the model's fit to the training data and the generalization of the model to new unseen data. Regularizing the model can avoid overfitting and improve the model's ability to generalize to new data.

In simple terms, regularization helps to keep a balance between how well the model fits the training data and how well it will perform on new unseen data. It helps to keep the model from becoming too complex, which can lead to poor performance on new data.

4. What is Gini–impurity index?

Ans.) The Gini-impurity index measures the impurity of a set of data. It is commonly used in decision tree-based algorithms, such as CART (Classification and Regression Trees), to determine the best feature to split the data on.

The Gini-impurity index is calculated as follows:

Gini = 1 - (p1^2 + p2^2 + ... + pn^2)

where pi is the proportion of the ith class in the dataset.

The Gini-impurity index ranges from 0 to 1, where a value of 0 indicates a perfectly pure dataset, and a value of 1 means the dataset is entirely impure.

In simple terms, the Gini-impurity index is a measure of how mixed the classes are in a particular node, a node with a low Gini impurity is a pure node, i.e., it contains only samples from a single class. In contrast, a node with a high Gini impurity is an impure node, i.e., it includes samples from multiple classes. The goal of building a decision tree is to build a tree such that the impurity of each node is minimized.

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

Ans.) Yes, unregularized decision trees are prone to overfitting. This is because decision trees can fit the training data very well by creating branches representing every possible data partition. If the tree is not pruned or regularized in some way, it will continue to develop branches until it perfectly fits the training data, which leads to overfitting.

When a decision tree is overfitting, it means that it is too complex and can fit the noise or random fluctuations in the training data. This can lead to poor performance on new unseen data.

In simple terms, Unregularized decision trees tend to fit the training data very well, which might lead to overfitting if the tree is not pruned or regularized. The tree will continue to create branches until it perfectly fits the training data, which can lead to poor performance on new unseen data.

6. What is an ensemble technique in machine learning?

Ans.) An ensemble technique in machine learning is a method of combining multiple models to improve the overall performance of the system. The idea behind ensemble techniques is that by combining the predictions of several models, the resulting prediction will be more accurate than the predictions of any individual model.

There are different types of ensemble techniques, such as:

Bagging: which stands for Bootstrap Aggregating, it is an ensemble method that generates multiple versions of the training set by randomly sampling the original data with replacement. Then, it trains a model on each version of the data and combines the predictions of all the models.

Boosting: is an ensemble method that trains several models sequentially, where each model tries to correct the mistakes of the previous model.

Stacking: is an ensemble method where multiple models are trained on the same data, and their predictions are combined into a new data set, which is then used to train a final model.

In simple terms, Ensemble techniques are methods that combine the predictions of several models to improve the overall performance of the system. It is a technique where multiple models are used together to make a final decision, it helps to overcome the problem of overfitting and improve the accuracy of the model.

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

Ans.) Bagging and Boosting are both ensemble techniques in machine learning, but they work in different ways.

Bagging stands for Bootstrap Aggregating; it is an ensemble method that generates multiple versions of the training set by randomly sampling the original data with replacement. Then, it trains a model on each version of the data and combines the predictions of all the models. The goal of bagging is to reduce the variance of the model by averaging the predictions of multiple models.

On the other hand, Boosting is an ensemble method that trains several models sequentially, where each model tries to correct the mistakes of the previous model. The goal of boosting is to reduce the model's bias by training models that focus on the examples that the earlier models misclassified.

In simple terms, Bagging is a technique that trains multiple models independently and combines their predictions by averaging. At the same time, Boosting is a technique that teaches multiple models sequentially, where each model tries to correct the mistakes of the previous model.

In short, Bagging is used to reduce the variance, while Boosting is used to reduce the bias.

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

Ans.) In a random forest, each tree is trained on a different subset of the data, called a bootstrap sample. The out-of-bag (OOB) error is a measure of the error of a random forest model that is calculated using the data that is not included in the bootstrap sample for each tree.

When a tree is trained on a bootstrap sample, about one-third of the data is not used in the training process and is referred to as the out-of-bag data. The predictions of the tree on the out-of-bag data are recorded, and the overall OOB error is calculated as the average error of all the trees on their out-of-bag data.

The OOB error can be used as an approximation of the model's generalization error, and it can be used as a way to estimate the model's performance on new unseen data.

In simple terms, Out-of-bag error is a measure of the error in Random Forests, calculated using the data not included in the bootstrap sample for each tree. It can be used to approximate the model's generalization error, which is the error on new unseen data. It can be an excellent way to check the model's performance.

9. What is K-fold cross-validation?

Ans.) K-fold cross-validation is a technique for evaluating the performance of a machine-learning model. It is used to estimate the model's performance on new unseen data. The basic idea is to divide the data into k subsets or "folds" of approximately equal size. Then, the model is trained on k-1 of the folds and tested on the remaining one. This process is repeated k times, with a different fold being used as the test set each time. Finally, the performance of the model is averaged over all k iterations.

Cross-validation helps overcome the overfitting problem by providing a more robust estimate of model performance. It also helps to identify whether a model is overfitting or underfitting by comparing the training and cross-validation errors.

In simple terms, K-fold cross-validation is a technique for evaluating the performance of a machine-learning model. It divides the data into k subsets, trains the model on k-1 subsets, and tests it on the remaining one, repeating this process k times. The performance of the model is then averaged over all iterations. It helps to overcome the problem of overfitting by providing a more robust estimate of model performance and helps to identify if a model is overfitting or underfitting.

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

Ans.) Hyperparameter tuning, also known as hyperparameter optimization, is the process of systematically searching for the best combination of hyperparameters for a machine-learning model. Hyperparameters are parameters that are not learned from the data but are set before training the model. Examples of hyperparameters include:

The learning rate in a neural network.

The number of trees in a random forest.

The regularization parameter in a linear regression model.

The goal of hyperparameter tuning is to find the hyperparameters that result in the best performance of the model on a given task. The process of tuning hyperparameters is usually done by training the model multiple times with different combinations of hyperparameters and evaluating the model's performance using a validation set.

Hyperparameter tuning is done to improve the performance of a model on unseen data; it helps to prevent overfitting and underfitting of the model. The model can perform better on unseen data and generalize better with the right set of hyperparameters.

In simple terms, Hyperparameter tuning is the process of finding the best combination of hyperparameters for a machine-learning model. Hyperparameters are parameters set before training the model; they are not learned from the data. The goal of hyperparameter tuning is to improve the model's performance on unseen data. It helps to prevent overfitting and underfitting of the model, and with the right set of hyperparameters, the model can generalize better.

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

Ans.) If the learning rate is set too large in Gradient Descent, it can cause issues such as oscillations, divergence, never converging, overstepping the minimum, and slow convergence. It can cause the algorithm to take large steps in the direction of the gradient, which can cause it to overshoot the optimal solution.

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

Ans.) Logistic Regression is a linear model, it makes the assumption that the relationship between input features and output labels is linear. It is generally unsuitable for the classification of non-linear data; for such cases, models like decision trees, random forests, or SVMs are better suited.

For example, if the data is in a form of a complex, non-linear shape like an "S" or a spiral, it will be difficult for logistic regression to capture the underlying pattern as it only tries to fit a straight line.

13. Differentiate between Adaboost and Gradient Boosting.

Ans.) AdaBoost and Gradient Boosting are both ensemble learning methods used for classification and regression problems, but they work in slightly different ways.

AdaBoost (Adaptive Boosting) is a sequential ensemble method that is based on the concept of boosting. It works by training a series of weak learners (e.g., decision trees with a small depth) on the data and then combining them to form a strong ensemble model. The process starts by training a weak model on the entire data. Then it gives more weight to the misclassified data points and trains another weak model. This process continues, and the weights of the weak models are adjusted to form a final model.

On the other hand, Gradient Boosting is an iterative method that creates a new model for each iteration. It tries to correct the mistakes of the previous model by focusing on the residual errors. Gradient Boosting uses decision trees as its base estimator, and the objective is to minimize the loss function by adding decision trees sequentially.

In simple terms, Adaboost is a sequential ensemble method based on the concept of boosting; it trains a series of weak learners and then combines them to form a strong ensemble model. On the other hand, Gradient Boosting is an iterative method that creates a new model for each iteration; it tries to correct the mistakes of the previous model by focusing on the residual errors, and it uses decision trees as its base estimator.

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

Ans.) The bias-variance trade-off is a fundamental concept in machine learning that refers to the trade-off between the accuracy of a model and its complexity.

Bias refers to the difference between the average predictions of a model and the true values. A model with high bias tends to make the same consistent errors, meaning that it oversimplifies the problem and cannot capture the data's complexity. High-bias models are also known as underfitting models.

Variance, on the other hand, refers to the variability of the predictions made by a model for different samples of the same data. A model with high variance tends to fit the training data very well but performs poorly on unseen data, meaning it overfits the training data.

In simple terms, Bias refers to the difference between the average predictions of a model and the true values; a model with high bias is known as an underfitting model. Variance refers to the variability of predictions made by a model for different samples of the same data; a model with high variance is known as an overfitting model.

The bias-variance trade-off states that as we decrease the bias of a model, its variance increases and vice versa. Machine learning aims to find the right balance between bias and variance, known as the sweet spot, and to achieve the best generalization performance on unseen data.

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

Ans.) Support Vector Machine (SVM) is a popular algorithm for classification and regression problems. One of the key features of SVM is the ability to use different kernel functions to transform the input data into a higher-dimensional space where a hyperplane can separate it.

The following are the three types of kernels commonly used in SVM:

Linear Kernel: The linear kernel is the most straightforward and commonly used function. It transforms the input data into a higher-dimensional space by adding a new dimension for each pair of original features. The dot product between the input vectors can represent the linear kernel.

RBF (Radial Basis Function) Kernel: The RBF kernel is a non-linear kernel function that creates a non-linear decision boundary. It maps the input data into a higher-dimensional space by creating a new dimension for each data point. The distance between the input vectors determines the value of the RBF kernel.

Polynomial Kernel: The polynomial kernel is another non-linear kernel function that creates a non-linear decision boundary. It maps the input data into a higher-dimensional space by creating a new dimension for each pair of original features. The value of the polynomial kernel is determined by the dot product between the input vectors raised to a power.

In short, the Linear kernel is the simplest and most commonly used kernel function, it creates linear decision boundary. RBF (Radial Basis Function) kernel is a non-linear kernel function that creates a non-linear decision boundary by mapping input data into a higher-dimensional space. The polynomial kernel is another non-linear kernel function that creates a non-linear decision boundary by mapping input data into a higher-dimensional space by creating a new dimension for each pair of original features.