ASSIGNMENT - 5

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 of goodness of fit model in regression and why?
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.
3. What is the need of regularization in machine learning?
4. What is Gini–impurity index?
5. Are unregularized decision-trees prone to overfitting? If yes, why?
6. What is an ensemble technique in machine learning?
7. What is the difference between Bagging and Boosting techniques?
8. What is out-of-bag error in random forests?
9. What is K-fold cross-validation?
10. What is hyper parameter tuning in machine learning and why it is done?
11. What issues can occur if we have a large learning rate in Gradient Descent?
12. Can we use Logistic Regression for classification of Non-Linear Data? If not, why?
13. Differentiate between Adaboost and Gradient Boosting.
14. What is bias-variance trade off in machine learning?
15. Give short description each of Linear, RBF, Polynomial kernels used in SVM.

Ans. 1 R-squared is a better measure of goodness of fit model in regression compared to Residual Sum of Squares (RSS) because it provides a standardized measure of how well the model fits the data. R-squared measures the proportion of variance in the dependent variable that is explained by the independent variables in the model. It is a value between 0 and 1, with 1 indicating a perfect fit between the model and the data.

On the other hand, RSS measures the total amount of unexplained variation in the dependent variable. A low RSS indicates a good fit, but it does not give any indication of how well the model fits the data in relation to the total variation in the dependent variable. Furthermore, the RSS is affected by the sample size, making it difficult to compare models with different sample sizes.

In summary, R-squared is preferred over RSS as it provides a standardized measure of how well the model fits the data, and it is not affected by sample size.

Ans. 2 In regression analysis, Total Sum of Squares (TSS) is the total variation in the dependent variable (Y) around its mean. It is calculated by subtracting the mean of Y from each Y value and taking the sum of squares of these differences:

TSS = Σ(Y - Y_mean)^2

Explained Sum of Squares (ESS) is the variation in Y that is explained by the regression model. It is calculated as the sum of squares of the differences between the predicted values of Y (Y_pred) and the mean of Y (Y_mean):

ESS = Σ(Y_pred - Y_mean)^2

Residual Sum of Squares (RSS) is the variation in Y that is not explained by the regression model. It is calculated as the sum of squares of the differences between the observed values of Y (Y_obs) and the predicted values of Y (Y_pred):

RSS = Σ(Y_obs - Y_pred)^2

The equation relating these three metrics is:

TSS = ESS + RSS

This equation shows that TSS can be partitioned into two components, ESS and RSS. ESS represents the part of the variation in Y that is explained by the regression model, while RSS represents the part of the variation that is not explained by the model (i.e. is due to random error). By comparing ESS and RSS, we can determine whether the regression model is a good fit for the data or not.

Ans. 3 Regularization is an important technique in machine learning used to avoid overfitting in the model. Overfitting occurs when the model is too complex and fits the training data too closely, leading to poor generalization on new, unseen data.

Regularization introduces a penalty term to the loss function, which controls the complexity of the model by adding constraints on the size of the model parameters. This helps to prevent the model from fitting too closely to the training data and encourages generalization on new data.

Regularization techniques like L1 regularization (Lasso) and L2 regularization (Ridge) are commonly used in regression models to shrink the coefficient estimates towards zero, forcing the model to focus only on the most important features and reducing the effect of noise and outliers. Regularization techniques are also used in neural networks to avoid overfitting, where it is referred as weight decay. It involves adding a regularization term to the loss function which penalizes large weights and biases, thereby preventing overfitting and improving the robustness of the model.

In summary, regularization is an important technique in machine learning that helps to control the complexity of the model and avoid overfitting, leading to better generalization and improved performance on new data.

Gini impurity is a measure used in decision tree algorithms to determine the degree of impurity or the homogeneity of a dataset of examples. It is used to decide which attribute will be selected as the splitting attribute at each node in the tree.

Ans 4. The Gini impurity index is based on the idea of the probability of misclassification. It measures the probability of incorrectly classifying a randomly chosen element in the dataset. If a dataset is completely homogeneous, the Gini impurity is zero, indicating a perfect split. If the dataset is equally divided among all possible classes, then the Gini impurity is 0.5, indicating the worst case scenario.

Mathematically, the Gini impurity is calculated for a given dataset as follows:

Gini impurity = 1 - ∑(pi)^2

where pi is the relative frequency of the ith class in the dataset.

In decision tree algorithms, the attribute with the lowest Gini impurity is chosen as the splitting attribute at each node to divide the dataset into two or more subsets that are as pure as possible. This process is repeated recursively until the purest subsets of the data are achieved, which form the leaf nodes of the decision tree.

Ans. 5 Yes, unregularized decision trees are prone to overfitting. The decision tree algorithm is a non-parametric method, which means that it can model almost any type of relationship between the independent variables and the target variable. When building a decision tree, the tree can grow very deep and produce many nodes --- this can lead to complex decision boundaries and overfitting.

Overfitting can occur in decision trees when the tree is specifically designed to fit to the training data with little regard for the generalization performance of the model. As the model is fitted to the training data, it may begin to capture random noise and outliers in the data which can degrade its performance on new, unseen data.

Regularization techniques can be used to address overfitting in decision trees. Some common approaches to regularization in decision trees include pruning and limit the maximum depth of the tree.

In pruning, the initial tree is grown to its maximum size then pruned back based on various criteria (e.g. cost-complexity, minimum description length). The aim is to simplify a complex complete tree by removing nodes that do not improve the prediction performance.

Limiting the maximum depth of the tree and setting a minimum size of leaf nodes can also avoid overfitting by imposing a constraint on the level of complexity of the tree's decision surface. As a result, the model will reduce the impact of noise and outliers in the training data and produce a better fit on new data.In summary, unregularized decision trees are prone to overfitting due to their flexibility in modeling the training data. Proper use of regularization methods can help to control the complexity of the model and avoid overfitting.

Ans 6. An ensemble technique is a machine learning approach that combines several models to improve the accuracy and robustness of a prediction. The idea behind ensemble techniques is that combining the predictions of multiple models can lead to better performance than any individual model alone.

Ensemble techniques can be classified into two main types: Model averaging and Model boosting.

1. Model Averaging: In Model averaging, a set of weak classifiers (models that are slightly better than random guessing) is combined to produce a strong classifier. The most common techniques in this category include:

- Bagging (Bootstrap Aggregating): In this technique, multiple models are trained on different subsets of the training data, and their predictions are averaged to make the final prediction. The key objective of bagging is to reduce the variance of the model, which can be achieved by resampling the training data.

- Random Forest: This technique is similar to bagging, where multiple decision trees are trained on different subsets of the training data. The trees are then averaged to produce the final predictions.

2. Model Boosting: In Model boosting, the focus is to improve the performance of a weak classifier by training a sequence of models that are weighted based on their errors. The most common techniques in this category include:

- AdaBoost (Adaptive Boosting): In this technique, models are trained sequentially, and the training data for each subsequent model is given more weight than the previous model. The final prediction is a weighted sum of the predictions of all the models.

- Gradient Boosting: This technique involves training multiple models that are designed to compensate for the errors made by the previous models. The final prediction is a weighted sum of the predictions of all the models.

Ensemble techniques have been used successfully in various machine learning tasks, including classification, regression, and anomaly detection. They have proven to be robust and effective in handling noisy and complex datasets.

Ans 7. Bagging (Bootstrap Aggregating) and Boosting are two ensemble learning techniques used in machine learning. They differ in terms of the way they combine the predictions of multiple models.

Bagging and Boosting can both improve the accuracy and robustness of a prediction problem, but they differ in some key ways:

1. Sampling Technique: Bagging relies on sampling with replacement to create different subsets of the dataset, whereas Boosting relies on sampling without replacement to create weighted versions of the dataset.

2. Model Weighting: In Bagging, multiple models are trained independently on different subsets of the dataset, and their predictions are averaged to produce the final prediction. In Boosting, models are trained sequentially, and the training data for each subsequent model is adjusted based on the errors of the previous models.

3. Focus on Error Reduction: Bagging is designed to reduce the variance of the model since it trains multiple models independently on different subsets of the data, but it can still lead to overfitting. On the other hand, Boosting focuses on reducing the bias of the model by paying more attention to the examples that are incorrectly classified. Boosting is thus more prone to overfitting compared to Bagging.

4. Types: Bagging is mainly used for unstable learners, i.e., high-variance models to improve their performance. Boosting is mainly used for unstable learners, i.e., models that are poor on both, bias and variance.

In summary, Bagging and Boosting are both ensemble methods that aim to improve the accuracy and robustness of machine learning models by combining the predictions of multiple models. The difference lies in the techniques that they use to do this. Bagging mainly focuses on reducing the variance of the model, while Boosting mainly focuses on reducing the bias of the model.

Ans 8. In Random Forests, Out-of-Bag (OOB) error is a metric used to estimate the generalization error of the model. The OOB error is computed using the samples that are not selected for building a particular tree during the random forest training process.

The Random Forest algorithm constructs multiple decision trees by training them on random subsets of the original dataset. These subsets are generated from the original dataset using the bootstrap method, where the samples are selected randomly with replacement. As a result, some samples are left out (not selected) when training each tree, which is known as the OOB samples.

After constructing all the trees, the OOB samples are used to calculate the OOB error. For each OOB sample, the algorithm collects its prediction across all the trees that did not include that sample in their bootstrap sample. The predictions are then averaged for the respective sample.

The OOB error is computed by comparing the predicted classes of the OOB samples with their actual classes. This error is used to estimate the performance of the random forest model on a fresh dataset. Since OOB samples are not used for training the tree, OOB error gives an unbiased estimate of the model performance without the need for a separate validation dataset.

In summary, OOB error is used as a metric to estimate the generalization error of a random forest model. It is computed by comparing the predictions of the OOB samples with their actual classes. It gives an unbiased estimate of model performance without the need for a separate validation dataset.

Ans 9 K-fold cross-validation is a technique used to assess the performance of a machine learning model. In K-fold cross-validation, the original dataset is split into K equal-sized subsets (or folds) randomly. The model is then trained on K-1 subsets (or folds) and validated using the remaining subset (or fold).

The process is repeated K times, and each fold is used once as a validation dataset. The performance of the model is then averaged over all K iterations to obtain a robust estimate of the model performance.

The advantage of K-fold cross-validation is that it uses all the available data for both training and validation, ensuring that the model is tested on all parts of the dataset. It also reduces the variance of the estimate of the model performance compared to other methods of splitting the data into training and testing sets. This method is commonly used in machine learning to evaluate the performance of a model on a given dataset, and to tune hyperparameters before deploying the model to make predictions on new data.

A common variant of K-fold cross-validation is stratified K-fold cross-validation, where the class distribution of the data is preserved in each fold. This ensures that each fold includes a representative sample of each class, which is important when working with imbalanced datasets.

In summary, K-fold cross-validation is a technique used to assess the performance of a machine learning model by splitting the original dataset into K subsets, training the model on K-1 of them and validating the model on the remaining subset. The performance of the model is then averaged over all K iterations. It is a commonly used technique to evaluate model performance and tune hyperparameters in machine learning.

Ans 10. Hyperparameter tuning is the process of selecting the optimal set of hyperparameters for a machine learning model before training the model on the data. Hyperparameters are the settings or configurations that cannot be learned from the data itself but are specified before training the model. These hyperparameters significantly affect the performance of the model.

Hyperparameter tuning is done to identify the optimal set of hyperparameters that result in better model performance. The process of hyperparameter tuning involves selecting a range of values for each of the hyperparameters and then systematically testing the performance of the model on the validation dataset with different combinations of these hyperparameters.

The model's performance is evaluated during hyperparameter tuning using a performance metric, such as accuracy, precision, recall, F1-score, or mean squared error (MSE). Grid search and randomized search are two common approaches for hyperparameter tuning. In the grid search approach, a set of hyperparameters are specified, and a grid of hyperparameters is defined with combinations of these hyperparameters. In the randomized search approach, random combinations of hyperparameters are tested to identify the optimal hyperparameters for the model.

Hyperparameter tuning is important because the selection of the optimal hyperparameters can significantly improve the performance of the model on new unseen datasets. A model that is not properly optimized for hyperparameters can result in poor performance on new data, leading to poor predictions or inaccurate results.

In summary, hyperparameter tuning is the process of selecting the best set of hyperparameters for machine learning models before training on new datasets. Hyperparameter tuning is performed to improve the model's performance on new datasets, and the selection process can be done using grid search or random search techniques.

Ans 11. Gradient descent is an optimization algorithm used to minimize the cost (or loss) function for machine learning models. The learning rate is an important hyperparameter in the gradient descent algorithm that controls the step size at each iteration of the optimization process. If the learning rate is too large, several issues can occur in the gradient descent optimization process, as explained below:

1. Divergence of the algorithm: A large learning rate causes the gradient descent algorithm to take very large steps in the direction of the steepest descent. If the learning rate is too large, the algorithm can overshoot the minimum and start oscillating between two opposite directions, failing to converge to the minimum value of the cost function.

2. Slow convergence rate: A large learning rate can cause the optimization process to converge slowly. The gradient descent algorithm takes large steps, which can result in the optimization process 'bouncing around' the minimum, requiring more iterations to reach convergence.

3. Risk of missing the global minimum: If the learning rate is too large, the gradient descent algorithm is more likely to skip the global minimum and converge only to a local minimum.

4. Numerical instability: A large learning rate can lead to numerical instability in the gradient descent algorithm. It can cause the algorithm to encounter a problem called 'overflow,' where the algorithm tries to compute a value that is too large to be stored in memory.

In summary, a large learning rate in gradient descent can cause the optimization process to diverge, converge slowly, miss the global minimum, and cause numerical instability. Therefore, it is important to choose an appropriate learning rate to ensure a smooth and efficient gradient descent optimization process.

Ans.12 Logistic regression is a linear classification algorithm that models the relationship between the dependent variable (binary outcome) and one or more independent variables (predictor variables). Logistic regression assumes a linear relationship between the independent variables and the log-odds of the binary outcome.

However, in cases where the relationship between the predictor variables and the binary outcome is non-linear, logistic regression may not be a suitable algorithm for classification of non-linear data. Logistic regression can only provide a linear decision boundary between two separate classes. If the relationship between the predictor variables and the binary outcome is non-linear, the decision boundary may not be able to separate all the data points into the correct categories.

In such cases, non-linear classification algorithms such as decision trees, support vector machines (SVM), or neural networks may be more appropriate for classification of non-linear data.

There are various techniques to convert non-linear data into linearly-separable data in logistic regression, such as polynomial transformation, interaction effects, or feature engineering. These transformations can help capture non-linear relationships between the predictor variables and the binary outcome and can improve the performance of logistic regression. However, if the non-linearity is too complex, those techniques may not work well, and other non-linear classifiers should be used.

In summary, Logistic regression is a powerful and simple algorithm for binary classification but can be outperformed by non-linear classifiers in cases where the relationship between the predictor variables and the binary outcome is non-linear.

Ans 13. Adaboost and Gradient Boosting are two popular boosting algorithms used for machine learning problems, and while they share some similarities, there are also some important differences between them:

1. Learning Algorithm: Adaboost and Gradient Boosting use different learning algorithms. Adaboost uses decision trees as its base learner, while Gradient Boosting uses decision trees, SVM, or linear regression as its base learner.

2. Sequential vs. Parallel Learning: Adaboost is a sequential learning algorithm, which means that each new model trained in the boosting process is trained using weighted data from previous models. In contrast, Gradient Boosting is a parallel learning algorithm, which means that each new model trained in the boosting process is trained independently of the previous models.

3. Weighting of Data: In Adaboost, the algorithm uses weighted data to train each new model to give more importance to those data points that were misclassified by previous models. In contrast, Gradient Boosting doesn't use any weighting, but it tries to fit a new model to the negative gradient of the loss function of the prior model.

4. Weak Learners: Adaboost focuses on finding weak learners that misclassify the data and then use them to improve the model. In contrast, Gradient Boosting focuses on finding a strong learner by training models iteratively that minimize the loss function.

5. Outlier Handling: Adaboost is highly sensitive to outliers, as they can significantly affect the weights in the Adaboost algorithm. In contrast, Gradient Boosting can handle outliers moderately well due to its residual-based approach.

In summary, Adaboost and Gradient Boosting are two boosting algorithms designed to combine weak learners into a strong learner for machine learning problems. Both methods aim to improve the model's performance iteratively, but they use different learning algorithms, weighting of data, and methods for handling outliers, among other differences.

Ans 14. The bias-variance tradeoff is a fundamental concept in machine learning that refers to the tradeoff between model complexity and generalization performance. In supervised learning, a model is trained on a labeled dataset, and the ultimate goal is to achieve high accuracy on unseen data. The bias-variance tradeoff refers to the ability of a model to generalize to new data by balancing the following two sources of error:

1. Bias: Bias is the error caused by the model's assumptions about the data. A model with high bias makes strong assumptions about the data and is likely to underfit the data, meaning it will be unable to capture the true relationship between the features and the target variable.

2. Variance: Variance is the error caused by the model's sensitivity to small fluctuations in the training data. A model with high variance is likely to overfit the data, meaning it will perform well on the training data but poorly on unseen data.

The bias-variance tradeoff can be visualized as a U-shaped curve, with high bias at one end and high variance at the other end. The optimal tradeoff between bias and variance is achieved when the model has sufficient complexity to capture the true underlying relationship between the features and the target variable, but not so much complexity that it overfits the data.

To achieve the optimal tradeoff between bias and variance, various techniques can be used such as regularized models, feature selection, and early stopping, among others. It's important to find an optimal balance between bias and variance to ensure that the model generalizes well to new data.

Ans .15 Linear, RBF, and Polynomial kernels are three common types of kernel functions used in Support Vector Machines (SVMs) for non-linear classification problems. Here's a brief description of each kernel:

1. Linear Kernel: The linear kernel is the simplest type of kernel and is used when the data is linearly separable. This kernel computes the dot product between pairs of feature vectors in the input space, which creates a linear decision boundary in the transformed feature space. The decision boundary is a straight line that separates the two classes of data.

2. RBF Kernel: The Radial Basis Function (RBF) kernel is a popular kernel used for non-linear classification problems. The RBF kernel computes the similarity between two data points in the input space by measuring the distance between them. The similarity decreases as the distance between the data points increases. The RBF kernel maps the data points to an infinite dimensional feature space where the decision boundary is non-linear and can handle complex data distributions. The RBF kernel introduces a hyperparameter 'gamma' which controls the width of the kernel and hence the flexibility of a model.

3. Polynomial Kernel: The polynomial kernel is another commonly used kernel function in SVMs. Like the RBF kernel, it transforms the data to a higher dimensional space where the decision boundary is non-linear. The polynomial kernel is best used when the data has curved or polynomial relationships. It works by computing the similarity of the data points in the input space by measuring the inner product of the feature space derived from pair-wise interactions between the input features. The polynomial kernel introduces a hyperparameter 'degree' which determines the degree of the polynomial interaction between the input features.

In summary, all three kernels - Linear, RBF, and Polynomial - are used to transform the data into higher dimensional spaces where the decision boundary is non-linear. The choice of a kernel depends on the nature of the data and the complexity of the problem. The linear kernel is simple and fast, but only works for linearly separable data. The polynomial kernel is best used for curved or polynomial relationships, whereas the RBF kernel is a good choice for general non-linear classification problems.