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 (R^2) is generally considered a better measure of goodness of fit in regression compared to the Residual Sum of Squares (RSS). R-squared measures the proportion of the variance in the dependent variable that is predictable from the independent variables in the model. It provides an indication of how well the independent variables explain the variability of the dependent variable. On the other hand, RSS measures the total squared difference between the observed and predicted values of the dependent variable. While RSS provides insight into the overall fit of the model, it doesn't account for the scale of the dependent variable and doesn't provide a standardized measure of goodness of fit like R-squared does. R-squared is easier to interpret and compare across different models, making it a preferred metric for assessing model performance in regression analysis.

ANS2) In regression analysis, TSS (Total Sum of Squares) represents the total variation in the dependent variable (Y) around its mean. ESS (Explained Sum of Squares) measures the variation explained by the regression model, or in other words, the variation in Y that is attributed to the independent variables in the model. RSS (Residual Sum of Squares) represents the unexplained variation, or the variation in Y that is not accounted for by the independent variables in the model.

The equation relating these three metrics is:

TSS = ESS + RSS

In words, the Total Sum of Squares (TSS) equals the Explained Sum of Squares (ESS) plus the Residual Sum of Squares (RSS). This equation illustrates that the total variation in the dependent variable can be decomposed into the variation explained by the regression model and the unexplained residual variation.

ANS3) Regularization is used in machine learning to prevent over fitting, which occurs when a model learns to perform well on the training data but fails to generalize to unseen data. Regularization techniques add a penalty term to the model's loss function, discouraging overly complex models that may fit the training data too closely. This helps to improve the model's ability to generalize to new, unseen data by promoting simpler and more robust models. Regularization techniques include L1 regularization (Lasso), L2 regularization (Ridge), and elastic net regularization, among others.

ANS4) The Gini impurity index is a measure used in decision tree algorithms to evaluate how well a particular feature splits the data into classes. It quantifies the likelihood of misclassifying a randomly chosen element if it were randomly labeled according to the distribution of labels in the branch.

Mathematically, the Gini impurity index for a set of items is calculated by summing the squared probabilities of each class being chosen, and then subtracting that sum from 1. A lower Gini impurity indicates a purer node, where all elements belong to the same class.

In a decision tree, the feature with the lowest Gini impurity index is usually chosen as the root node for splitting the data.

ANS5) Yes, unregularized decision trees are prone to overfitting. This is because decision trees have the capability to grow very deep and complex, capturing even the noise present in the training data. As a result, they may end up fitting the training data too closely, leading to poor generalization performance on unseen data.Without regularization, decision trees will continue to split nodes until each leaf node contains only instances of a single class (or until it meets some other stopping criterion like minimum samples per leaf). This can result in a highly specific model that memorizes the training data rather than learning underlying patterns.Regularization techniques such as pruning or limiting the maximum depth of the tree are used to prevent this overfitting by promoting simpler tree structures that generalize better to new data.

ANS6) An ensemble technique in machine learning involves combining multiple models to improve predictive performance. Instead of relying on a single model, ensemble methods leverage the diversity and collective wisdom of multiple models to make more accurate predictions.

There are several types of ensemble techniques, including:

1. *Bagging (Bootstrap Aggregating)*: This technique involves training multiple instances of the same base model on different subsets of the training data, often using bootstrap sampling (sampling with replacement). The final prediction is typically an average or voting of the predictions from each model.

2. *Boosting*: Boosting algorithms train a sequence of weak learners (models that are slightly better than random guessing) in such a way that each subsequent model focuses on the mistakes made by the previous ones. Examples include AdaBoost, Gradient Boosting, and XGBoost.

3. *Random Forest*: Random Forest is an ensemble technique that combines the concepts of bagging and decision trees. It builds multiple decision trees using random subsets of the features and then averages their predictions.

4. *Stacking*: Stacking involves training multiple diverse models and then using another model (called a meta-model or blender) to combine their predictions. The meta-model is trained on the predictions of the base models.

Ensemble techniques are often used because they can improve predictive performance, reduce overfitting, and increase robustness compared to individual models.

ANS7) The main difference between Bagging and Boosting techniques lies in how they create diversity among the base models and how they combine their predictions:

1. *Diversity of Base Models*:

   - *Bagging (Bootstrap Aggregating)*: Bagging creates diversity among the base models by training each model on a random subset of the training data, typically using bootstrap sampling (sampling with replacement). This means that each model sees a slightly different subset of the data.

   - *Boosting*: Boosting creates diversity among the base models by sequentially training each model to correct the mistakes made by the previous ones. Each subsequent model focuses more on the instances that were misclassified by the previous models. This iterative process leads to a sequence of models where each one specializes in different areas of the feature space.

2. *Combination of Predictions*:

   - *Bagging*: In bagging, the final prediction is typically made by averaging (for regression) or voting (for classification) the predictions of all base models. Since each model has an equal vote, bagging gives equal importance to all models.

   - *Boosting*: In boosting, the final prediction is typically made by weighted averaging of the predictions of all base models, where the weights are determined based on the performance of each model. Models that perform better on the training data are given higher weights in the final prediction.

3. *Training Process*:

   - *Bagging*: Bagging trains base models independently in parallel. Each model is trained on a different subset of the data, and there is no interaction between the models during training.

   - *Boosting*: Boosting trains base models sequentially in a stage-wise manner. Each model is trained based on the performance of the previous models, with more emphasis placed on instances that were misclassified.

In summary, while both Bagging and Boosting techniques aim to improve predictive performance by combining multiple models, they differ in how they create diversity among the models and how they combine their predictions. Bagging focuses on creating diversity through random sampling of the data and combines predictions equally, while Boosting creates diversity by sequentially training models to correct the mistakes of previous models and combines predictions with weighted averaging.

ANS8) In Random Forests, the out-of-bag (OOB) error is an estimate of the model's performance on unseen data. It is calculated using the data points that are not included in the bootstrap samples used to train each individual tree in the forest. During the bootstrap sampling process, approximately one-third of the data points are left out (on average) from each tree's training set. These left-out data points are called out-of-bag samples. Since these samples were not used in the training of the tree, they can be used to estimate the model's performance. For each data point, its prediction is obtained by aggregating the predictions from all trees in the forest where that data point was not included in the bootstrap sample. The OOB error is then calculated as the error rate (or loss) of these aggregated predictions .The OOB error provides an unbiased estimate of the model's performance without the need for a separate validation set, making it a useful tool for evaluating the Random Forest model's generalization ability.

ANS9) K-fold cross-validation is a technique used to assess the performance of a machine learning model and to mitigate issues related to overfitting and selection bias. It involves partitioning the dataset into k equal-sized subsets, or folds. 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.

Here's how K-fold cross-validation works:

1. *Partitioning the Data*: The dataset is divided into k equal-sized subsets (folds). Typically, the data is shuffled randomly before partitioning to ensure that each fold contains a representative sample of the data.

2. *Training and Evaluation*: The model is trained k times. In each iteration, one of the k folds is used as the validation set, and the remaining k-1 folds are used as the training set. The model is trained on the training set and evaluated on the validation set.

3. *Performance Metrics*: After each iteration, a performance metric (such as accuracy, F1-score, or mean squared error) is calculated based on the model's predictions on the validation set.

4. *Aggregate Performance*: Once all k iterations are complete, the performance metrics from each iteration are averaged to obtain a single performance estimate for the model.

K-fold cross-validation helps to provide a more reliable estimate of the model's performance compared to a single train-test split because it uses multiple subsets of the data for both training and validation. It also allows for a more efficient use of the available data, especially when the dataset is limited in size. Common choices for the value of k include 5-fold and 10-fold cross-validation, but other values can be used depending on the size and nature of the dataset.

ANS10) Hyper parameter tuning in machine learning refers to the process of selecting the optimal values for the hyper parameters of a machine learning model. Hyper parameters are parameters that are set before the training process begins and control aspects of the learning process, such as the model's complexity, regularization strength, learning rate, etc. Unlike model parameters, which are learned from the data during training, hyper parameters are set by the data scientist or researcher.

Hyper parameter tuning is done to improve the performance of the model by finding the combination of hyper parameters that results in the best performance on a validation dataset. The process typically involves trying out different combinations of hyper parameters and evaluating the model's performance using a validation set or a cross-validation technique. The goal is to find the hyper parameters that optimize the model's performance metric, such as accuracy, precision, recall, or F1-score.

Hyper parameter tuning is important because the choice of hyper parameters can have a significant impact on the performance of the model. Selecting inappropriate values for hyper parameters can lead to suboptimal performance, including overfitting or underfitting the data. By systematically searching through the hyper parameter space and selecting the best values, hyper parameter tuning helps to improve the generalization ability and robustness of the model.

ANS11) If we have a large learning rate in Gradient Descent, several issues can occur, including:

1. *Divergence*: With a large learning rate, the updates to the model parameters can become too large, causing the optimization process to overshoot the minimum of the loss function. This can lead to the optimization algorithm diverging, meaning it fails to converge to a solution and keeps oscillating or moving away from the optimal parameters.

2. *Oscillations*: A large learning rate can cause the optimization process to oscillate around the minimum of the loss function, especially if the loss surface is rough or irregular. This oscillation can slow down convergence or prevent the algorithm from converging altogether.

3. *Instability*: Large learning rates can make the optimization process unstable, causing the model parameters to fluctuate widely between iterations. This instability can make it difficult to obtain consistent or reliable results.

4. *Skipping Over Optima*: With a large learning rate, the optimization algorithm may skip over smaller, local minima in the loss function and converge to a suboptimal solution or fail to converge at all.

5. *Overfitting*: Large learning rates can lead to overfitting, where the model becomes too specialized to the training data and performs poorly on unseen data. This is because large updates to the model parameters can cause the model to memorize noise or outliers in the training data rather than learning the underlying patterns.

To mitigate these issues, it's important to choose an appropriate learning rate for the optimization algorithm. Techniques such as learning rate scheduling, adaptive learning rates (e.g., AdaGrad, RMS prop, Adam), and careful monitoring of the optimization process can help ensure stable and efficient training of machine learning models.

ANS12) Logistic Regression is a linear classification algorithm, which means it models the relationship between the independent variables and the log-odds of the dependent variable using a linear function. Therefore, it is not suitable for handling non-linear data because it cannot capture complex non-linear relationships between the features and the target variable.When faced with non-linear data, Logistic Regression may struggle to find a decision boundary that effectively separates the different classes. As a result, it may lead to poor classification performance or inaccurate predictions.To handle non-linear data, more flexible and complex models such as Support Vector Machines with non-linear kernels, Decision Trees, Random Forests, Gradient Boosting Machines, Neural Networks, and kernelized versions of Logistic Regression (such as Kernel Logistic Regression) are often used. These models are capable of capturing non-linear relationships in the data and can provide better classification performance on non-linear datasets.

ANS13) Adaboost and Gradient Boosting are both ensemble learning methods used for classification and regression tasks, but they differ in their approach to training base learners and updating model predictions:

1. *Training Process*:

- *Adaboost (Adaptive Boosting)*: Adaboost works by sequentially training a series of weak learners (e.g., decision trees) on the training data. In each iteration, the algorithm assigns higher weights to the misclassified data points from the previous iteration, focusing on the difficult-to-classify instances. Subsequent weak learners are trained to correct the mistakes made by the previous ones, with each new learner paying more attention to the misclassified instances.

- *Gradient Boosting*: Gradient Boosting, on the other hand, builds an ensemble of weak learners (usually decision trees) sequentially, but it doesn't adjust the weights of data points. Instead, it fits each new weak learner to the residuals (the differences between the observed and predicted values) of the previous model. In each iteration, the model minimizes the loss function by adding a new learner that focuses on reducing the residual errors.

2. *Model Complexity*:

- *Adaboost*: Adaboost typically uses simple weak learners (e.g., shallow decision trees) as base models. Each weak learner is trained to focus on the mistakes of the previous learners, gradually improving the overall performance of the ensemble.

- *Gradient Boosting*: Gradient Boosting can handle more complex weak learners and is often used with decision trees. It builds a strong model by combining multiple weak learners in a way that optimally reduces the loss function, leading to potentially more complex models.

3. *Loss Function*:

- *Adaboost*: Adaboost minimizes the exponential loss function by iteratively re-weighting misclassified instances. It focuses on reducing the classification error by assigning higher weights to misclassified points.

- *Gradient Boosting*: Gradient Boosting minimizes a user-specified loss function (e.g., mean squared error for regression, cross-entropy for classification) by fitting each new weak learner to the residuals of the previous model. It aims to directly optimize the loss function by reducing the residuals in each iteration.

In summary, while both Adaboost and Gradient Boosting are ensemble methods that combine multiple weak learners to create a strong model, they differ in their approach to training base learners and updating model predictions. Adaboost focuses on re-weighting misclassified instances to correct mistakes, while Gradient Boosting directly minimizes a user-specified loss function by fitting new weak learners to the residuals of the previous model.

ANS14) The bias-variance tradeoff is a fundamental concept in machine learning that refers to the tradeoff between the bias of an estimator and its variance.

- *Bias*: Bias refers to the error introduced by approximating a real-world problem with a simplified model. A model with high bias tends to oversimplify the underlying relationships in the data and may under fit the training data. It fails to capture the true patterns in the data, leading to systematic errors in predictions.

- *Variance*: Variance refers to the model's sensitivity to small fluctuations or noise in the training data. A model with high variance is overly complex and captures noise in the training data along with the underlying patterns. As a result, it may fit the training data very well but generalize poorly to new, unseen data.

The bias-variance tradeoff arises because reducing one component (bias or variance) typically increases the other. For example:

- Increasing the complexity of a model (e.g., adding more features or increasing the model capacity) can reduce bias but may increase variance.

- Conversely, decreasing the complexity of a model (e.g., using fewer features or regularization techniques) can reduce variance but may increase bias.

The goal in machine learning is to find the right balance between bias and variance that minimizes the model's overall error on unseen data. This balance depends on the specific characteristics of the dataset and the problem at hand. Techniques such as cross-validation, regularization, and model selection can help find an appropriate tradeoff between bias .

ANS15) Sure, here's a brief description of each kernel used in Support Vector Machines (SVMs):

1. *Linear Kernel*:

   - The linear kernel is the simplest kernel function used in SVM.

   - It computes the dot product between the feature vectors, without any transformation.

   - It works well when the data is linearly separable or when the number of features is very large.

2. *RBF (Radial Basis Function) Kernel*:

- The RBF kernel, also known as the Gaussian kernel, is a popular non-linear kernel used in SVM.

- It computes the similarity between two feature vectors based on the Euclidean distance between them.

- It maps the data into a high-dimensional space where it becomes linearly separable.

- It has two hyper parameters: gamma (γ), which controls the width of the kernel, and C, which controls the regularization strength.