**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?**

**Answer:-** R-squared is generally considered a better measure of goodness of fit in regression models than the Residual Sum of Squares (RSS).

R-squared provides a standardized measure, indicating the proportion of the variance in the dependent variable that is predictable from the independent variables. It ranges from 0 to 1, making it easier to interpret and compare across different models. An R-squared value closer to 1 indicates a better fit.

In contrast, RSS is the total of the squared differences between observed and predicted values. While it measures the overall discrepancy, it is not standardized and can be influenced by the scale of the data, making it less intuitive for comparison purposes.

Therefore, R-squared is preferred because it offers a more interpretable and comparative measure of how well the regression model explains the variability in 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.**

**Answer:-** In regression, these terms represent how well your model fits the data:

**TSS (Total Sum of Squares):** Reflects the total variability of the dependent variable around its mean. (Imagine spreading all the data points around the average value)

**ESS (Explained Sum of Squares):** Captures the variability explained by the regression line. (How well the fitted line captures the data points)

**RSS (Residual Sum of Squares):** Represents the unexplained variability, the leftover "errors" between the actual data points and the predicted values from the model.

The equation relating them is:

**TSS = ESS + RSS**

This shows that the total variability (TSS) can be broken down into the variation explained by the model (ESS) and the unexplained variation (RSS).

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

**Answer:-** Regularization is crucial in machine learning to prevent overfitting. Overfitting occurs when a model memorizes the noise and irrelevant details in the training data, leading to poor performance on unseen data.

Regularization techniques penalize overly complex models, promoting simpler and more generalizable ones that perform well on new data. This is achieved by:

**Reducing model complexity:** Regularization discourages the model from learning intricate patterns that might not be relevant for unseen data.

**Improving generalization:** By preventing overfitting, regularization ensures the model captures the underlying trends and performs well on future data.

**4.What is Gini–impurity index?**

**Answer:-** The Gini impurity index is a measure of impurity used in decision tree algorithms for classification tasks. It indicates the likelihood of a randomly chosen data point being misclassified if labelled based on the class distribution at a particular node in the tree.

**Range:** 0 (perfectly pure) to 0.5 (maximally impure)

**Lower value:** Preferable, indicating the node is more homogeneous (all instances belong to the same class).

**Higher value:** Suggests the node contains a mixed bag of classes, making classification less certain.

Gini impurity helps decision trees select the best split at each node, ultimately leading to a more accurate model.

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

**Answer:-** Yes, unregularized decision trees are absolutely prone to overfitting. Here's why:

**Greedy Splitting:** Decision trees naturally try to perfectly split the data at each node. This can lead to very complex trees that capture even minor noises and irrelevant details in the training data.

**No Penalty for Complexity:** Without regularization, there's nothing stopping the tree from growing excessively. This complexity makes it good at fitting the training data perfectly, but struggles to generalize to unseen data.

Essentially, unregularized decision trees become too focused on the specific training examples they see and fail to learn the underlying patterns that hold true for new data.

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

**Answer:-** Ensemble techniques in machine learning combine the predictions from multiple models to create a single, more powerful model. It's like consulting multiple experts for a better outcome.

Here's the gist:

- Train several individual models (called base learners).
- Combine their predictions using techniques like voting or averaging.
- The ensemble aims to be more accurate and robust than any single model by leveraging their collective strengths and reducing weaknesses.

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

**Answer:-** Both Bagging and Boosting are ensemble techniques in machine learning that create a powerful model by combining multiple models, but they differ in their approach:

**Bagging (Bootstrap Aggregation):**

- Focuses on reducing variance. Trains multiple models on different bootstrapped copies (samples with replacement) of the original data.
- Models are independent and learn in parallel.
- Final prediction is the average of individual predictions (regression) or majority vote (classification).
- Less prone to overfitting compared to a single model.

**Boosting:**

- Focuses on reducing bias. Trains models sequentially.
- Each model addresses the errors of the previous model.
- Models are weighted based on their performance, with higher weights for better performing models.
- Final prediction is a weighted average of individual predictions.
- Can be more prone to overfitting if not carefully tuned.

In simpler terms:

- Bagging: Averages predictions from a bunch of independent learners to reduce random noise (variance).
- Boosting: Trains a series of learners, each correcting the errors of the previous one, to reduce bias.

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

**Answer:-** Out-of-Bag (OOB) error is a method for estimating the prediction error of a random forest model. Here's the key idea:

- Random forests use bootstrapping, where each tree trains on a random sample of the data (with replacement).
- Data points left out during a tree's training are called out-of-bag (OOB) samples.

- The OOB error is the average error of the predictions made by a tree on the OOB samples for that specific tree.
- By averaging the OOB errors from all trees, we get an estimate of the random forest's performance on unseen data.

**Benefits of OOB Error:**

- Provides a validation score without needing a separate validation set.
- Efficient way to assess model generalizability.

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

**Answer:-** K-fold cross-validation is a technique in machine learning to assess model performance on unseen data. It works like this:

1. Split data into k equal folds.
2. Do k times:
    - Use one fold for testing, remaining k-1 folds for training.
    - Train the model and evaluate on the test fold.
3. Average the performance metrics across all folds.

This provides a more reliable estimate of how well a model generalizes to new data.

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

**Answer:-** Hyperparameter tuning in machine learning is the process of finding the optimal values for a model's hyperparameters. These are settings that control the learning process itself, rather than being learned from the data.

Here's why it's crucial:

- Optimizes Model Performance: Choosing the right hyperparameters can significantly impact a model's accuracy, generalization, and efficiency.

- Prevents Overfitting and Underfitting: Proper tuning helps avoid overfitting (memorizing noise) and underfitting (failing to capture patterns). It finds the sweet spot for model complexity.
- Improves Generalizability: By tuning hyperparameters, you ensure the model learns the underlying trends and performs well on unseen data, not just the training data.

In essence, hyperparameter tuning fine-adjusts the learning process to squeeze the best possible performance out of your machine learning model.

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

**Answer:-** A large learning rate in Gradient Descent can lead to several issues:

- **Overshooting the Minimum:** Imagine climbing down a hill. With a large step size (learning rate), you might jump right past the lowest point (minimum) and end up on the other side. This can cause the model to oscillate around the minimum or even diverge completely.
- **Slow Convergence:** In some cases, a large learning rate might not cause overshooting, but it can make the learning process very jittery and take much longer to converge on the optimal solution.
- **Local Optima:** With a large step size, the model might get stuck in a shallow valley (local minimum) instead of reaching the global minimum (the deepest valley).

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

**Answer:-** Logistic regression is best suited for linearly separable data in classification tasks. Here's why:

- **Limited by Linearity:** It models the relationship between features and class probability using a straight line.

- **Non-linear Data Struggles:** Real-world data often has curves and bends, which a straight line can't capture effectively.

However, it can still be useful for non-linear data with some tricks:

- **Feature Engineering:** We can create new features that capture the non-linearity (e.g., polynomial terms, feature interactions).
- **Regularization:** Techniques like L1/L2 can prevent overfitting to non-linear quirks in the data.

For truly non-linear data, consider algorithms like decision trees, support vector machines, or neural networks.

**13.Differentiate between Adaboost and Gradient Boosting.**

**Answer:-** AdaBoost and Gradient Boosting are both ensemble techniques that excel at improving model performance by combining multiple weak learners, but they differ in their approach:

**AdaBoost (Adaptive Boosting):**

- Focuses on reducing training error.
- Trains weak learners sequentially.
- Adjusts the weights of data points in subsequent stages.
- Emphasizes instances that were misclassified by previous learners.
- Final prediction is a weighted average of individual weak learner predictions.

**Gradient Boosting:**

- Focuses on reducing loss function (e.g., squared error for regression).
- Trains weak learners sequentially.
- Each learner targets the residual errors from the previous learner.
- Aims to progressively improve upon the model's fit.

- Final prediction is a weighted sum of individual weak learner predictions.

In simpler terms:

- AdaBoost: Makes existing weak learners pay more attention to previously misclassified examples.
- Gradient Boosting: Teams up weak learners to correct the errors of each other in a step-by-step manner.

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

**Answer:-** The bias-variance trade-off in machine learning is a fundamental concept that deals with the balance between two sources of error in a model:

- Bias: How well the model captures the underlying trend in the data (generalization). A high bias model underfits the data, meaning it's too simple and misses important patterns, leading to consistently wrong predictions.
- Variance: How sensitive the model is to the specific training data (sensitivity to noise). A high variance model overfits the data, meaning it memorizes even noise and irrelevant details, leading to poor performance on unseen data.

The ideal scenario is to find a sweet spot between these two errors. A model with low bias and low variance would be both accurate on the training data and generalize well to unseen data.

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

**Answer:-** Here's a brief description of each kernel function used in Support Vector Machines (SVM):

1. Linear Kernel:

- **Simplest kernel:** Represents the dot product of the input features.

- **Works well for:** Linearly separable data (data that can be perfectly divided by a straight line).
- **Computationally efficient:** Faster to train compared to other kernels due to its simplicity.

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

- **Widely used:** Captures non-linear relationships between features by transforming them into a higher-dimensional space.
- **Flexible:** Can handle complex data patterns.
- **Hyperparameter tuning:** Requires tuning the "gamma" parameter which controls the influence of each data point.

**3. Polynomial Kernel:**

- **Explicitly maps data to a higher-dimensional polynomial space.**
- **Can model complex non-linear relationships.**
- **Drawbacks:** Can be computationally expensive for high-degree polynomials. Possibility of overfitting if the chosen degree is too high.

In essence, the choice of kernel function depends on the complexity of your data and the desired balance between accuracy and computational efficiency. Linear kernel is good for simple, linearly separable data, while RBF and Polynomial kernels are more versatile for handling non-linear data, with trade-offs in terms of complexity and hyperparameter tuning.