ASSIGNMENT-2

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 indicates the proportion of variance explained by the model and is intuitive, but it factors in the number of variables, potentially leading to over fitting. RSS measures the model fit directly by summing squared residuals, with a lower RSS indicating a better fit. The choice between them depends on data set specifics, and commonly a combined approach alongside other metrics, like adjusted R-squared, provides a clearer measure of model fit. The correct option is d) It depends on the specific characteristics of the data set.

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:-

Interpretability:

a) **R-squared** represents the proportion of the variance in the dependent variable that is explained by the independent variables in the model. It is a standardized metric, ranging from 0 to 1 (or 0% to 100%), which makes it easy to interpret. A higher R-squared value means a better fit.
b) **RSS**, on the other hand, is the total squared difference between the observed and predicted values. While RSS gives the absolute error, it is not standardized and can vary depending on the scale of the data, making it less interpretable on its own.

Comparability:

a) **R-squared** allows for easier comparison between models, especially when datasets have different scales or units. You can compare models based on R-squared values to see which explains more of the variability in the outcome.
b) **RSS** does not allow for such straightforward comparisons, especially across different datasets or models, because it is influenced by the units and magnitude of the data.

Relative Fit:

a) **R-squared** gives a relative measure of how well the model fits the data, as it is based on the total variance explained by the model compared to the total variance present.
b) **RSS** only tells you how far off the predictions are, without giving any indication of how well the model does relative to the overall variance.

In conclusion, **R-squared** is preferred because it provides a clearer and more interpretable measure of the model's goodness of fit and is independent of the scale of the data.

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

   Ans:- Regularization is critical in machine learning to prevent overfitting, improve generalization, control model complexity, and handle high-dimensional data. It helps produce models that perform well on unseen data, making them more robust and reliable in real-world scenarios.

4. What is Gini–impurity index?

   Ans:- The Gini Index is a proportion of impurity or inequality in statistical and monetary settings. In

machine learning, it is utilized as an impurity measure in decision tree algorithms for classification tasks. The Gini Index measures the probability of a haphazardly picked test being misclassified by a decision tree algorithm, and its value goes from 0 (perfectly pure) to 1 (perfectly impure).

Gini Index Formula:-

$$Gini = 1 - \sum_{i=1}^{n} (p_i)^2$$

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

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

A) **High Variance**: Decision trees have the ability to fully fit the training data, splitting it until each leaf node contains only one data point or a very small subset of points. This leads to very specific patterns being captured, including noise in the data.
B) **Complex Trees**: When no regularization is applied (e.g., limits on tree depth or minimum number of samples per leaf), the decision tree can become very complex with many branches, essentially memorizing the training data. While this reduces error on the training set, it often results in poor generalization to new, unseen data.
C) **Sensitivity to Noise**: Unregularized decision trees can model even small, insignificant variations in the data, leading to splits based on noise rather than meaningful patterns, causing overfitting.

To prevent overfitting, regularization techniques like setting a maximum tree depth, minimum samples per leaf, or pruning the tree can be applied.

6. What is an ensemble technique in machine learning?

Ans:- An ensemble technique in machine learning is a method that combines multiple models (often referred to as "weak learners") to produce a more accurate, robust, and reliable prediction than any single model alone. The idea is that by aggregating the predictions of multiple models, the weaknesses of individual models can be minimized, leading to better overall performance.

There are two main types of ensemble methods:

A) **Bagging (Bootstrap Aggregating)**:

- Bagging involves training multiple models independently on different subsets of the training data, typically created using bootstrapping (random sampling with replacement).
- The final prediction is usually made by averaging the predictions (for regression) or majority voting (for classification).
- Example: **Random Forest**, which builds multiple decision trees on bootstrapped datasets and combines their predictions.

B) **Boosting**:

- Boosting sequentially trains models, where each new model focuses on correcting the mistakes made by the previous ones. The models are trained in sequence, with each new model giving higher weight to the previously misclassified data points.

- The final prediction is a weighted sum of all the models.
- Example: **AdaBoost**, **Gradient Boosting**, and **XGBoost**.

C) **Stacking**:

- Stacking involves training multiple different models (e.g., decision trees, support vector machines, etc.) and combining their predictions using a "meta-model" that learns how to best combine them.
- The meta-model is usually trained on the predictions of the base models, learning how to improve the final prediction by weighing the strengths of each base model.

Benefits of Ensemble Techniques:

- **Improved accuracy**: By combining multiple models, ensemble methods can often outperform a single model.
- **Reduced variance**: They help mitigate overfitting by averaging out the predictions from multiple models.
- **Better generalization**: They tend to generalize better to unseen data.

Ensemble techniques are widely used in competitive machine learning and real-world applications due to their effectiveness.

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

Ans:- Bagging and Boosting are both ensemble techniques in machine learning, but they differ in how they create and combine multiple models to improve predictive performance. Here's a breakdown of their key differences:

S.No.	Partiular	Bagging	Boosting
1.	**Purpose and Approach**	**Goal**: Reduce variance and prevent overfitting. **Approach**: Multiple models are trained independently on different bootstrapped subsets of the training data, and the final prediction is made by averaging (for regression) or voting (for classification). **Key Idea**: Parallel training of multiple models on random subsets to produce stable, less overfitted results	**Goal:** Reduce bias by improving weak models. **Approach:** Models are trained sequentially, where each new model tries to correct the errors made by the previous one. Misclassified instances are given more weight in the next iteration. **Key Idea:** Build a strong model by focusing more on difficult cases through iterative improvement.
2.	**Training Process**	● The base models (usually decision trees) are trained in **parallel** on bootstrapped datasets (random sampling with replacement). ● Each model is treated equally, and predictions are averaged (for regression) or majority-voted (for classification) in the end.	● The base models are trained in a **sequential** manner, where each new model is built to correct the errors of the previous model. ● The final model gives different weights to each base model's prediction, and the emphasis is on the models that reduce errors in hard-to-classify examples.

3.	**Handling of Data**	Each base model gets a **randomly sampled subset** of data with replacement (bootstrapping). This means some data points may be repeated in a subset while others may be excluded.	Each new model is trained on the **entire dataset**, but higher weight is assigned to data points that were misclassified by the previous model, making it focus on harder cases.
4.	**Model Complexity**	It helps in reducing **variance**, making models less sensitive to fluctuations in the training data. It works well when the base models tend to overfit (e.g., decision trees).	It reduces **bias**, making models stronger by focusing on hard-to-predict examples. Boosting typically results in a more complex final model than bagging because it systematically improves each iteration.
5.	**Final Prediction**	Predictions from multiple models are **combined equally** by averaging (for regression) or voting (for classification). Each model contributes the same weight to the final prediction.	Predictions from models are **combined with different weights**, usually based on their accuracy, meaning better-performing models have a larger influence on the final prediction.
6.	**Common Algorithms**	**Random Forest** is the most well-known bagging algorithm, where multiple decision trees are trained and their results are averaged or voted.	**AdaBoost**, **Gradient Boosting**, **XGBoost**, and **LightGBM** are popular boosting algorithms, with each iteration improving on the previous one.

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

Ans:- Out-of-bag (OOB) error in Random Forests is a method for evaluating the performance of the model without the need for separate test data or cross-validation. It provides an internal estimate of the model's accuracy based on the data samples that were not used during the construction of each individual tree in the forest.

9. What is K-fold cross-validation?

Ans:- **K-fold cross-validation** is a resampling technique used to evaluate the performance of a machine learning model. It helps assess how well the model generalizes to unseen data by reducing the risk of overfitting. The technique involves splitting the dataset into multiple subsets (or "folds") and using them for both training and validation in a systematic way.

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

Ans:- Hyperparameter tuning in machine learning refers to the process of selecting the optimal values for the hyperparameters of a model. Hyperparameters are parameters that are not learned directly from the training data, but instead, they are set before the learning process begins. These parameters control how the model is trained and how it makes predictions.

Why is Hyperparameter Tuning Done?:

- Optimize Model Performance: The performance of a machine learning model (accuracy, precision, recall, etc.) is highly dependent on the choice of hyperparameters. Tuning them ensures that the model performs as well as possible on the given data.

- Prevent Overfitting or Underfitting: Proper hyperparameter tuning helps prevent overfitting (where the model is too complex and memorizes the training data) or underfitting (where the model is too simple to capture the patterns in the data).
- Model Efficiency: It also affects the efficiency of the training process. For example, selecting an appropriate learning rate can speed up convergence without overshooting the optimal solution.

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

Ans:- A large learning rate in gradient descent can cause several issues that affect the performance of the model and the convergence of the optimization process. Here are the key problems that can occur:

1. Overshooting the Minimum:

- A large learning rate means that the model takes **large steps** in the direction of the gradient during each iteration. Instead of moving smoothly toward the optimal solution (the minimum of the loss function), the algorithm may overshoot the minimum repeatedly. This results in the model bouncing around the minimum without converging.
- **Effect**: The model may never settle at the optimal solution, causing poor convergence or no convergence at all.

2. Failure to Converge:

- With a large learning rate, the updates to the model's parameters may be so large that the gradient descent algorithm fails to converge to a solution. The model may either diverge (increase the error) or oscillate without making meaningful progress toward reducing the loss function.
- **Effect**: Gradient descent may fail to minimize the loss function, leading to either a high error or the algorithm becoming stuck without improving performance.

3. Instability:

- Large steps in gradient descent can cause instability in the training process. This instability manifests as erratic behavior in the loss function, where the loss may increase and decrease dramatically from one iteration to the next.
- **Effect**: The training process becomes unpredictable, and the model's performance may degrade instead of improving, making it harder to monitor and debug.

4. Suboptimal Solution:

- A large learning rate might lead to missing the global minimum and instead settling in a suboptimal region of the loss function. This happens because the large steps prevent the model from exploring the finer details of the loss landscape, potentially missing smaller valleys that represent better solutions.
- **Effect**: The model might converge to a higher loss (local minimum) rather than the global minimum, leading to suboptimal performance.

5. Divergence:

- In extreme cases, if the learning rate is too large, the model's parameters may move far away from the optimal values with each update, causing the loss function to increase rapidly.
- **Effect**: The model can diverge entirely, meaning the loss increases instead of decreasing, making the training process completely ineffective.

Visual Representation:

In a plot of the loss function:

- With a **small learning rate**, the gradient descent algorithm moves slowly but steadily toward the minimum.
- With a **large learning rate**, the steps are so large that the algorithm can overshoot the minimum, oscillate around it, or diverge altogether.

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

Ans:- Logistic Regression, in its basic form, **cannot** directly handle non-linear data for classification because it is a **linear model**. Here's a detailed explanation of why that is the case and what can be done to extend its capabilities:

Why Logistic Regression Can't Handle Non-Linear Data:

Linear Decision Boundary:

A) Logistic regression is designed to find a linear relationship between the input features and the log-odds of the target class. In other words, it fits a **linear decision boundary** to separate the classes.
B) For binary classification, the decision boundary is a straight line (in 2D), a plane (in 3D), or a hyperplane (in higher dimensions) that separates one class from the other.
C) **Problem**: If the data is not linearly separable (i.e., the relationship between the input features and the output class is non-linear), logistic regression will not be able to accurately capture the pattern in the data, leading to poor classification performance.

13. Differentiate between Adaboost and Gradient Boosting.

Ans:- **AdaBoost** (Adaptive Boosting) and **Gradient Boosting** are both popular ensemble learning techniques that combine multiple weak learners (usually decision trees) to create a stronger, more accurate model. However, they differ significantly in the way they build and update their models.

Here's a Comparison:

S.No.	Aspect	AdaBoost	Gradient Boosting
1.	**Error Focus**	Focuses on **misclassified examples**	Focuses on **residual errors**
2.	Loss Function	Implicit (exponential loss)	Explicit (minimizes a specific loss)
3.	Data Weighting	Reweights misclassified examples	No reweighting, fits residuals instead
4.	Weak Learners	Typically uses decision stumps (shallow trees)	Typically uses deeper trees
5.	Regularization	Limited control over regularization	Offers various regularization options
6.	Speed	Faster, but less effective on complex data	Slower, but more powerful and flexible
7.	Sensitivity to Noise	More sensitive to noisy data and outliers	Less sensitive with regularization

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

Ans:- On the other hand, in certain cases, it struggles to grasp the intricacies of the data and thus fails to provide an accurate prediction. Striking a balance between accuracy and the ability to make predictions beyond the training data in an ML model is called the bias-variance.

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

Ans:- In Support Vector Machines (SVM), kernels are functions that enable the algorithm to operate in a higher-dimensional space without explicitly transforming the data. Different kernels allow SVM to capture various types of relationships in the data. Here's a short description of three common kernels:

1. Linear Kernel:

Definition: The linear kernel computes the inner product of two input vectors in the original feature space.

Use Case: Best suited for linearly separable data, where a straight line (or hyperplane) can effectively separate the classes.

Advantages: Simple and computationally efficient; it works well when the data is not too complex and has a clear linear boundary.

2. Polynomial Kernel:

Definition: The polynomial kernel computes the inner product of the input vectors raised to a specified power (degree).

Use Case: Useful for data where the relationship between classes is polynomial in nature, allowing for curved decision boundaries.

Advantages: Can model more complex relationships compared to the linear kernel, providing flexibility in capturing interactions between features.

3. RBF (Radial Basis Function) Kernel:

Definition: The RBF kernel computes the similarity between two input vectors based on their Euclidean distance, using a Gaussian function.

Use Case: Effective for non-linear data and can handle cases where the classes are not linearly separable in the original feature space.

Advantages: Highly flexible and can create complex decision boundaries; it can adapt to the distribution of the data without requiring prior knowledge about the relationships.