MACHINE LEARNING

ANS 1: – R-squared and Residual Sum of Squares (RSS) are both measures of the goodness of fit of a regression model, but they serve slightly different purposes.

R-squared, also known as the **coefficient of determination**, measures the proportion of variance in the dependent variable that is explained by the independent variables in the model. It ranges from 0 to 1, with a higher value indicating a better fit. R-squared is useful for comparing different models or for determining the proportion of the variability in the dependent variable that is explained by the model.

On the other hand, **Residual Sum of Squares (RSS)** measures the difference between the observed values of the dependent variable and the predicted values by the model. It represents the sum of the squared differences between the actual and predicted values of the dependent variable. The goal is to minimize the residual sum of squares to obtain a better model fit.

In terms of determining the goodness of fit of a model, **R-squared is generally considered a better measure than RSS.** This is because R-squared provides an overall measure of the proportion of variance in the dependent variable that is explained by the model, whereas RSS only measures the magnitude of the residuals. R-squared is a standardized measure and ranges from 0 to 1, making it easy to compare the fit of different models. In contrast, the magnitude of the RSS value depends on the scale of the dependent variable and can't be easily compared across models.

ANS 2:-

1. Total Sum of Squares (TSS): TSS represents the total variation in the dependent variable (response variable) that is explained by the regression

model. It measures the total deviation of the dependent variable from its mean. TSS quantifies the variability in the dependent variable without considering the regression model, serving as a benchmark for evaluating the goodness of fit of the model.

Equation : TSS= $\sum_{i=1}^{n}(y_i-\bar{y})^2$

Where: y_i = observed dependent variable

$\bar{y}$= mean of the dependent variable

2.Explained Sum of Squares (ESS): ESS represents the variation in the dependent variable that is explained by the regression model. It measures the deviation of the predicted values of the dependent variable from its mean. ESS quantifies the improvement in prediction achieved by the regression model compared to simply using the mean of the dependent variable.

Equation: ESS = $\sum(\hat{y}_i-\bar{y})^2$

where $\hat{y}_i$ represent the predicted value of the dependent variable

3.Residual Sum of Squares (RSS):RSS represents the unexplained variation in the dependent variable by the regression model. It measures the deviation of the observed values of the dependent variable from the predicted values (residuals). RSS quantifies the error or discrepancy between the observed and predicted values of the dependent variable.

Equation: RSS = $\sum(y_i-\hat{y}_i)^2$

Where y_i represents the observed value of the dependent variable, and $\hat{y}_i$represents the predicted values of the dependent variable.

Relationship between SSR, SSE, and SST:-

TSS=ESS+RSS.

ANS 3: -

- Regularization is one of the most important concepts of machine learning. It is a technique to prevent the model from overfitting by adding extra information to it. Sometimes the machine learning model performs well with the training data but does not perform well with the test data.
- It means the model is not able to predict the output when deals with unseen data by introducing noise in the output, and hence the model is called overfitted.
- This problem can be deal with the help of a regularization technique. it maintains accuracy as well as a generalization of the model. It mainly regularizes or reduces the coefficient of features toward zero.

ANS 4: -

- The Gini impurity index is a measurement of the impurity or uncertainty in a set of data.
- It's commonly used in decision tree algorithms for classification problems.
- It calculates the probability of misclassifying a randomly chosen element if it were labelled according to the distribution of labels in the set.
- In simpler terms, it measures how often a randomly chosen element would be incorrectly classified if it were randomly labelled according to the distribution of labels in the set.
- A lower Gini impurity indicates a purer node in a decision tree, where all the elements belong to the same class.

ANS 5: -

Yes, unregularized decision trees are prone to overfitting. Overfitting occurs when a model learns the training data too well, capturing noise or random fluctuations that are not representative of the true underlying patterns in the data. Unregularized decision trees have the tendency to grow very deep, complex trees that perfectly fit the training data, but may generalize poorly to unseen data. Some of the reasons for unregularized decision tree are as follows: -

1. High Variance: Unregularized decision trees can capture intricate patterns in the training data, including noise, outliers, or small fluctuations, leading to high variance. As a result, the model might not generalize well to new, unseen data.

2. Memorization of Data: Decision trees have the capability to memorize the training data, especially when allowed to grow without any constraints. This memorization can lead to overfitting, as the model may not learn the underlying patterns and instead merely memorize the training instances.

3. Recursive Partitioning: Decision trees make splits in the data based on features that optimally separate the training instances into pure or nearly pure subsets. Without constraints, the tree can continue to split until each leaf node contains only one training instance, effectively memorizing the training data.

To reduce overfitting, various techniques such as pruning, limiting tree depth, or using ensemble methods like random forests or gradient boosting are commonly employed. These techniques bring regularization to the decision tree model.

ANS 6:-

- Ensemble learning is a machine learning technique that enhances accuracy and resilience in forecasting by merging predictions from multiple models. It aims to mitigate errors or biases that may exist in individual models by leveraging the collective intelligence of the ensemble. The concept behind ensemble learning is to combine the outputs of diverse models to create a more precise prediction.
- This approach not only enhances accuracy but also provides resilience against uncertainties in the data. By effectively merging predictions from multiple models, ensemble learning has proven to be a powerful tool in various domains, offering more robust and reliable forecasts.

ANS 7: -

- Bagging is a method of merging the same type of predictions. Boosting is a method of merging different types of predictions.

- Bagging decreases variance, not bias, and solves over-fitting issues in a model. Boosting decreases bias, not variance.

- In Bagging, each model receives an equal weight. In Boosting, models are weighed based on their performance.

- Models are built independently in Bagging. New models are affected by a previously built model's performance in Boosting.

- In Bagging, training data subsets are drawn randomly with a replacement for the training dataset. In Boosting, every new subset comprises the elements that were misclassified by previous models.

- Bagging is usually applied where the classifier is unstable and has a high variance. Boosting is usually applied where the classifier is stable and simple and has high bias.

Difference Between Bagging and Boosting

Feature	**Bagging**	**Boosting**
Objective	Reduce variance and prevent overfitting	Reduce bias and improve accuracy

Base Learners	Independent models trained in parallel	Sequentially trained weak learners
Weighting	Equal weight for all base learners	Weighted based on performance
Error Correction	Independent errors; no re-weighting	Emphasis on correcting mistakes
Training Speed	Parallel training; faster	Sequential training; slower
Final Model	Average or voting of base models	Weighted sum of base learners
Robustness	Less prone to overfitting	Prone to overfitting if not controlled

ANS 8: -

In random forests, the out-of-bag (OOB) error is an estimation of the model's performance on unseen data. It is calculated using the data points that are not included in the bootstrap sample used to train each individual decision tree in the forest.

Here's how the out-of-bag error is calculated:

1. In random forests, each decision tree is trained on a bootstrap sample of the original dataset. A bootstrap sample is created by randomly selecting data points from the original dataset with replacement, which means some data points may be selected multiple times while others may not be selected at all.

2. Since each bootstrap sample is created randomly and may not contain all data points, there will be some data points left out of each bootstrap sample. These left-out data points constitute the out-of-bag (OOB) data.

3. After training each decision tree, the OOB data points are used to calculate the prediction error of that particular tree. The prediction error is typically measured using a metric such as classification error or mean squared error, depending on the type of problem (classification or regression).

4. Finally, the OOB errors from all individual trees in the random forest are averaged to obtain the overall OOB error for the entire forest. This OOB error provides an unbiased estimate of the model's performance on unseen data without the need for a separate validation set.

The out-of-bag error serves as a useful tool for assessing the performance of a random forest model and tuning its hyperparameters, such as the number of trees in the forest or the maximum depth of each tree. It helps in evaluating the model's ability to generalize to new, unseen data and can guide the selection of optimal hyperparameters to improve the model's performance.

ANS 9: -

Cross-validation is a statistical method used to estimate the skill of machine learning models.

The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is

often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.

Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.

It is a popular method because it is simple to understand and because it generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split.

ANS 10: -

When we're training machine learning models, each dataset and model needs a different set of hyperparameters, which are a kind of variable. The only way to determine this is through multiple experiments, where you pick a set of hyperparameters and run them through your model. This is called **hyperparameter tuning.** We can summarize by saying that we're training our model sequentially with different sets of hyperparameters. This process can be manual, or we can pick one of several automated hyperparameter tuning methods.

Whichever method we use, we need to track the results of our experiments. We'll have to apply some form of statistical analysis, such as the loss function, to determine which set of hyperparameters gives the best result.

Hyperparameter tuning is an important and computationally intensive process.

Hyperparameter tuning is done to optimize the performance of a machine learning model. The goal is to find the set of hyperparameters that results in the best performance on unseen data, such as improving accuracy, reducing overfitting, or enhancing generalization. Since the optimal values of hyperparameters are

unknown a priori and can significantly affect the model's performance, hyperparameter tuning is a crucial step in the machine learning pipeline.

Hyperparameter tuning is typically performed using techniques such as grid search, random search, Bayesian optimization, or more advanced optimization algorithms. These techniques involve systematically exploring the hyperparameter space and evaluating the model's performance for different hyperparameter configurations. By finding the best combination of hyperparameters, hyperparameter tuning helps to improve the effectiveness and efficiency of machine learning models.

ANS 11: -

Gradient descent is an optimization algorithm used in machine learning to minimize the cost function by iteratively adjusting parameters in the direction of the negative gradient, aiming to find the optimal set of parameters.

The cost function represents the discrepancy between the predicted output of the model and the actual output. The goal of gradient descent is to find the set of parameters that minimizes this discrepancy and improves the model's performance.

Gradient descent can be applied to various machine learning algorithms, including linear regression, logistic regression, neural networks, and support vector machines. It provides a general framework for optimizing models by iteratively refining their parameters based on the cost function.

Having a large learning rate in gradient descent can lead to several issues:

1. **Overshooting the Minimum:** With a large learning rate, the updates to the model parameters can be so significant that they overshoot the minimum of the loss function. This can cause the algorithm to oscillate around the minimum or even diverge altogether, failing to converge to an optimal solution.

2. **Divergence:** A large learning rate can lead to unstable behaviour, causing the optimization process to diverge rather than converge to a solution.

3. **Unstable Gradients:** Large learning rates can result in large gradients, especially in deep neural networks. This can lead to unstable gradients, where the gradients fluctuate widely from one iteration to the next. Unstable gradients can hinder the convergence of the optimization algorithm and make it difficult to train the model effectively.

4. **Poor Generalization:** Overshooting the minimum or divergence caused by a large learning rate can result in poor generalization performance.

5. **Slow Convergence:** Using a large learning rate can sometimes slow down the convergence of the optimization algorithm.

ANS 12: - Logistic Regression is a linear classification algorithm. It assumes a linear relationship between the independent variables and the log-odds of the dependent variable. As a result, Logistic Regression is not well-suited for handling non-linear data directly.

If the decision boundary between classes in the data is non-linear, Logistic Regression may struggle to capture this complex relationship effectively. In such cases, Logistic Regression might underfit the data, leading to poor classification performance.

Logistic Regression is not inherently suitable for handling non-linear data but it can still be used with appropriate techniques and modifications to address non-linearity in the data to some extent.

However, for tasks involving highly non-linear relationships, other machine learning algorithms like kernel methods or neural networks may be more suitable.

ANS 13: -

ADABOOST	**GRADIENT BOOSTING**
During each iteration in Adaboost, the weights of incorrectly classified samples are increased, so that the next weak learner focuses more on these samples.	Gradient Boosting updates the weights by computing the negative gradient of the loss function with respect to the predicted output.

AdaBoost uses simple decision trees with one split known as the decision stumps of weak learners.	Gradient Boosting can use a wide range of base learners, such as decision trees, and linear models.
AdaBoost is more susceptible to noise and outliers in the data, as it assigns high weights to misclassified samples.	Gradient Boosting is generally more robust as it updates the weights based on the gradients, which re less sensitive to outliers.
Algorithm: training process start with the decision tree stump(usually). At every step, the weights of the training samples which are misclassified are increased for the next iteration. The next tree is built sequentially on the same training data but using the newly weighted training sample s this process is repeated until a desired performance is achieved.	Algorithm: GBM uses gradient descent to iteratively fit new weak learners to the residuals of the previous ones, minimising a loss function. There are several loss functions to choose from. Mean squared error being most common for regression and cross entropy for classification. GBM uses decision tree as the weak learners.
The final model is formed by combining the predictions from individuals' trees through a weighted sum.	The final model is an equal-weighted sum of all the individual trees.

ANS 14: -

The bias-variance trade-off is a fundamental concept in machine learning that describes the balance between two sources of error, bias, and variance, when building predictive models. It helps us understand the trade-off between model complexity and generalization performance.

1. Bias: - Bias refers to the error introduced by the assumptions made by the learning algorithm when simplifying the underlying patterns in the data. High bias can lead to underfitting, where the model fails to capture the true relationship between the features and the target variable.

2.Variance: - Variance refers to the error introduced by the model's sensitivity to small fluctuations in the training data. High variance can lead to overfitting, where the model performs well on the training data but generalizes poorly to unseen data.

The bias-variance trade-off arises because decreasing bias often increases variance and vice versa. Achieving optimal model performance involves finding the right balance between bias and variance.

Here's how the trade-off works:

High Bias, Low Variance Models: - Models with high bias and low variance are typically simple and make strong assumptions about the data. They tend to generalize well but may underfit the training data. Examples include linear regression with few features or shallow decision trees.

Low Bias, High Variance Models: - Models with low bias and high variance are often complex and flexible, capturing intricate patterns in the data. They may perform well on the training data but generalize poorly to new data. Examples include deep neural networks, high-degree polynomial regression, or decision trees with high depth.

Optimal Trade-off: - The goal is to find a model that strikes the right balance between bias and variance to minimize the total error, known as the expected prediction error.

Methods for managing the bias-variance trade-off include:

Model Selection: Choosing the appropriate complexity of the model based on the problem requirements and available data.

Regularization: Applying techniques like L1 or L2 regularization to penalize overly complex models and reduce variance.

Cross-Validation: Evaluating model performance using techniques like k-fold cross-validation to estimate the generalization error and avoid overfitting.

Ensemble Methods: Combining multiple models (e.g., bagging, boosting) to reduce variance and improve predictive performance.

the bias-variance trade-off is a crucial concept in machine learning that helps practitioners understand the relationship between model complexity, generalization performance, and the inherent trade-off between bias and variance. Finding the optimal balance is essential for building models that generalize well to new, unseen data.

ANS 15: -

Kernel Function is a method used to take data as input and transform it into the required form of processing data. “Kernel” is used due to a set of mathematical functions used in Support Vector Machine providing the window to manipulate the data. So, Kernel Function generally transforms the training set of data so that a non-linear decision surface can transform to a linear equation in a higher number of dimension spaces. Basically, it returns the inner product between two points in a standard feature dimension.

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

1.Linear Kernel: - The linear kernel is the simplest kernel function used in SVM. It calculates the dot product between two feature vectors in the original input space. The decision boundary generated by the linear kernel is a straight line in two dimensions (a hyperplane in higher dimensions). It is suitable for linearly separable data or when the decision boundary is expected to be linear.

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 transforms the input space into a higher-dimensional space using a non-linear mapping. The RBF kernel computes the similarity between two samples based on the Euclidean distance between them in the transformed space.

3.Polynomial Kernel: - The polynomial kernel computes the similarity between two samples as the polynomial of the dot product of the original feature vectors. It is defined by a degree parameter, which determines the degree of the polynomial. The decision boundary generated by the polynomial kernel can be linear or non-linear depending on the degree of the polynomial. Higher-degree polynomials can capture more complex decision boundaries but may also lead to overfitting, especially with high-dimensional data.

The choice of kernel depends on the characteristics of the data and the complexity of the decision boundary required for the problem at hand.