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?***

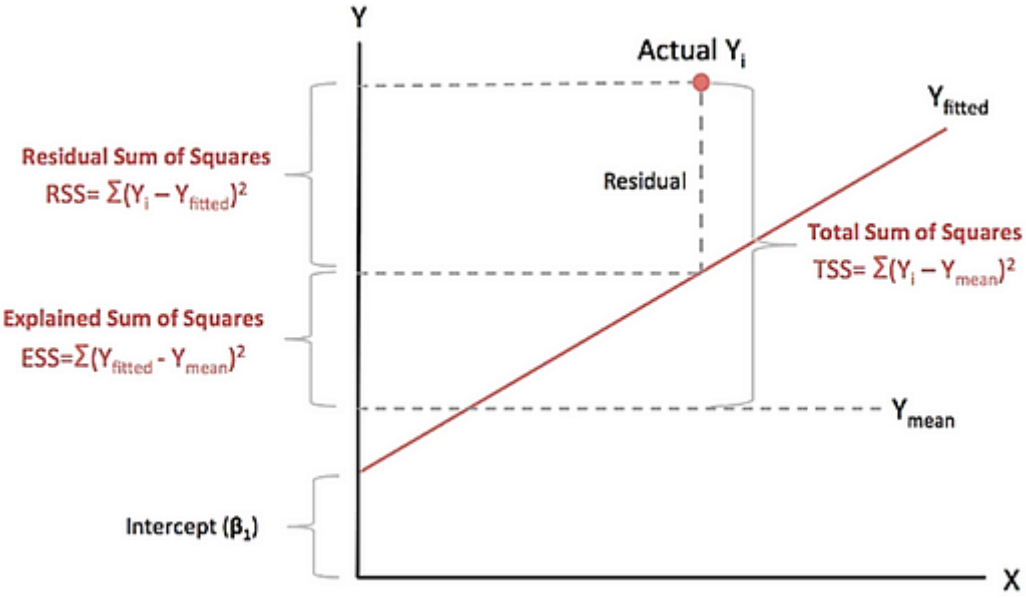
   Answer

   R-squared (R^2) value is generally considered a better measure of the goodness of fit of a model compared to the Residual Sum of Squares (RSS).
   Reasons are:

   1. **Interpretability:** R-squared shows how well the model fits the data. A higher R-squared means the model explains more of the data's variability.
      For example, an R-squared value of 0.80 indicates that 80% of the variability in the dependent variable is accounted for by the independent variables. This makes it easier for researchers and practitioners to understand and communicate the performance of the model.
   2. **Scale Independence:** R-squared is not affected by the scale of the variables and ranges from 0 to 1. Higher values mean a better fit.
      In contrast, RSS is influenced by variable scales.
   3. **Comparability:** R-squared allows easy comparison between models. RSS alone does not give a clear way to compare models directly.
   4. **Model Complexity:** R-squared penalizes adding unnecessary predictors. It will not increase much if extra predictors do not help explain the data. However, RSS decreases when more predictors are added, even if they do not make the model better.

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



1. **Total Sum of Squares (TSS)**

   a) *Definition:*
   - TSS (also known as SST, which stands for "Sum of Squares Total") measures the total variability in the dependent variable (response variable).
   - It quantifies how much the observed data points deviate from the overall mean.
   - Think of TSS as the dispersion of the observed variables around the mean, similar to the variance in descriptive statistics.

   b) *Mathematical Equation:*

   The TSS is calculated as follows

$$TSS = \sum_{i=1}^{N} (Y_i - \bar{Y})^2$$

   Where:
   Y_i *represents observed dependent variable*
   $\bar{Y}$*represents the mean of the dependent variable,*
   N *represents the total number of observations.*

   c) *Interpretation:*
   - A larger TSS indicates greater variability in the data.
   - In regression analysis and ANOVA (analysis of variance), TSS helps assess the goodness of fit of a model and understand the relationship between variables.

2. **Explained Sum of Squares (ESS)**

   a) *Definition:*

- ESS (also known as SSR, which stands for "Sum of Squares due to Regression") quantifies the variability explained by the regression model.
- It measures how well the regression line fits the data by considering the deviation of predicted values from the mean of the response variable.
- ESS represents the sum of squared differences between the predicted data points ($\hat{Y}_i$) and the mean of the response variable (Y_i).

b) Mathematical Equation:

$$ESS = \sum_{i=1}^{N} (\hat{Y}_i - \bar{Y})^2$$

$\hat{Y}_i$ *represents the predicted value of the dependent variable for the ith observation*
$\bar{Y}$*represents the mean of the dependent variable,*
N *represents the total number of observations.*

c) Interpretation:

- A larger ESS indicates that the regression model explains more of the variability in the response variable.
- ESS contributes to the overall goodness of fit of the model.

3. Residual Sum of Squares (RSS)

a) Definition:

- RSS (also known as SSE, which stands for "Sum of Squares error") quantifies the unexplained variability in the dependent variable (response variable) by a regression model.
- It measures the overall difference between your actual data points and the values predicted by your regression model.
- A "residual" is a measure of the distance from a data point to the regression line.

b) Mathematical Equation:

$$RSS = \sum_{i=1}^{N} (Y_i - \hat{Y}_i)^2 \qquad or \qquad \sum_{i=1}^{N} e_i^2$$

Where:
Y_i *represents observed dependent variable*
$\hat{Y}_i$ *represents the predicted value of the dependent variable for the ith observation*
e_i *is the difference between the actual value of the dependent variable and the predicted value,*
N *represents the total number of observations.*

c) Interpretation:

- A smaller RSS indicates that the regression model fits the data well, as it captures less unexplained variation.
- RSS is used to assess the goodness of fit of the model and evaluate how well it explains the observed data.

The equation that relate the three metrics with each other is

$$\sum_{i=1}^{N} (Y_i - \bar{Y})^2 = \sum_{i=1}^{N} (\hat{Y}_i - \bar{Y})^2 + \sum_{i=1}^{N} (Y_i - \hat{Y}_i)^2$$

OR

$$\sum_{i=1}^{N}(Y_i - \bar{Y})^2 = \sum_{i=1}^{N}(\hat{Y}_i - \bar{Y})^2 + \sum_{i=1}^{N} e_i^2$$

TSS = RSS + ESS

Total Variability = Explained variability + Unexplained variability

Where:

Y_i *represents observed dependent variable*

$\hat{Y}_i$ *represents the predicted value of the dependent variable for the ith observation*

$\bar{Y}$*represents the mean of the dependent variable,*

N *represents the total number of observations.*

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

While training a machine-learning model, the model can easily be over fitted or under fitted.
To avoid this, we use regularization in machine learning to properly fit a model onto our test set.
Regularization techniques help reduce the chance of overfitting and under fitting and help, us get an optimal model.

What Are Overfitting and under fitting?

To train our machine learning model, we give it some data to learn from. The process of plotting a series of data points and drawing the best fit line to understand the relationship between the variables is called **Data Fitting.** our model is the best fit when it can find all necessary patterns in our data and avoid the random data points and unnecessary patterns called Noise.

A. Common Regularization Techniques:
   a) **L2 Regularization (Ridge Regression**): Adds the sum of squared coefficients to the loss function. It penalizes large coefficients.
   b) **L1 Regularization (Lasso Regression):** Adds the sum of absolute coefficients to the loss function. It encourages sparsity (some coefficients become exactly zero).
   c) **Elastic Net:** Combines L1 and L2 regularization.

B. Regularization Using Python in Machine Learning
   A Regularization example in Python using the well-liked scikit-learn module is given below:
   a) Regularization (Lasso Regression)

```
from sklearn.linear_model import Lasso, Ridge, ElasticNet
```

```
#create an instance of lasso

ls = Lasso(alpha = 0.0001)
ls.fit(x_train, y_train)
```

```
print(ls.score(x_train,y_train))
```

```
predls = ls.predict(x_test)
```

```
predls
```

```
print('Mean Sqaure Error', mean_squared_error(y_test, predls))
np.sqrt( mean_squared_error(y_test, predls))
```

```
ls.coef_
```

   b) L2 Regularization (Ridge Regression)

In [ ]:
```
# To minimize the coefficient variance
rd = Ridge(alpha = 0.0001)
rd.fit(x_train, y_train)
print(rd.score(x_train, y_train))

predrd = rd.predict(x_test)
print('Mean Sqaure Error', mean_squared_error(y_test, predrd))
np.sqrt( mean_squared_error(y_test, predrd))
```

In [ ]:
```
rd.coef_
```

   c) Elastic Net Regularization

```
#ElasticNet

eln = ElasticNet(alpha = 0.0001)

eln.fit(x_train, y_train)

predeln = eln.predict(x_test)
print(rd.score(x_train, y_train))

print('Mean Sqaure Error', mean_squared_error(y_test, predeln))
np.sqrt( mean_squared_error(y_test, predeln))
```

```
eln.coef_
```

C. When to Use Which Regularization Technique?

a) L1 Regularization (Lasso):

When to Use:

- Select important features: L1 regularization encourages sparsity by driving some coefficients to exactly zero. It acts as an automatic feature selection method.
- Reduce the number of features: If you have many features and suspect that some are irrelevant, L1 regularization can help eliminate them.

b) L2 Regularization (Ridge):

When to Use:

- Control the magnitude of coefficients: L2 regularization penalizes large coefficients without forcing them to be exactly zero.
- Avoid overfitting: Ridge regression (L2 regularization) helps stabilize coefficient estimates and reduce variance.

c) Elastic Net:

When to Use:

- Balance between feature selection and coefficient shrinkage: It provides a trade-off between L1 and L2 regularization.
- Handle multicollinearity: Elastic Net works well when you have correlated features.

4. *What is Gini–impurity index?*

   The Gini Impurity, also known as the Gini Index, is a measurement used in building Decision Trees. It helps determine how the features of a dataset should split nodes to form the tree. Here are the key points about Gini Impurity:

   A. *Definition:*
      - The Gini Impurity of a dataset is a number between 0 and 0.5.
      - It indicates the likelihood of new, random data being misclassified if it were given a random class label according to the class distribution in the dataset.

   B. *Use in Decision Trees:*
      - When constructing a decision tree, we want to find the best feature for splitting each node.
      - Gini Impurity measures how poorly each feature divides the data into the correct class (e.g., “default” or “not default”).
      - The feature with the lowest impurity determines the best feature for splitting the current node.

   C. *Use in Decision Trees:*
      - When constructing a decision tree, we want to find the best feature for splitting each node.
      - The feature with the lowest Gini Impurity determines the best feature for splitting the current node.
      - Decision trees aim to minimize the Gini Impurity during the tree-building process.

   D. *Comparison with Entropy:*
      - Gini Impurity and Entropy are both used for feature selection in decision trees.
      - While Gini Impurity is faster to compute, Entropy provides similar results.
      - The choice between them often depends on the specific problem and personal preference.

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

   Yes, unregularized decision trees are prone to overfitting.

   Reasons

   - To prevent overfitting, it's essential to prune decision trees (remove unnecessary branches) or use regularization techniques (such as limiting the depth or controlling the minimum samples per leaf).
   - unregularized decision trees are prone to overfitting due to their flexibility and lack of constraints. Regularization helps control their complexity and improves generalization to unseen data.

6. What is an ensemble technique in machine learning?
   A. Defination:
      - Ensemble techniques in machine learning involve combining multiple individual models to improve predictive performance.
      - These techniques are popular because they can often produce more robust and accurate predictions compared to individual models.
      - Ensemble methods aim at improving predictability in models by combining several models to make one very reliable model.
      - Ensemble methods are ideal for regression and classification, where they reduce bias and variance to boost the accuracy of models.
   B. Categories of Ensemble Methods
      a) a)Sequential Ensemble Techniques:
         - Examples: Adaptive Boosting (AdaBoost)
         - These methods generate base learners sequentially, where each new learner focuses on correcting the errors made by the previous ones.
         - Sequential generation promotes dependence between the base learners, as later models assign higher weights to previously misclassified instances.
      b) Parallel Ensemble Techniques:
         - Examples: Random Forest
         - In contrast to sequential methods, parallel techniques generate base learners simultaneously.
         - Parallel generation encourages independence between the base learners, which helps reduce errors through averaging.

      Additionally, ensemble methods can involve either homogenous or heterogeneous base learners:
      a) Homogeneous Base Learners: All base learners are of the same type and exhibit similar characteristics.
      b) Heterogeneous Base Learners: Base learners are of different types, leading to diverse and complementary contributions to the ensemble's predictive power.
   C. Common types of Ensemble Methods
      a) Bagging(Bootstrap AGGregating)
         - Bagging(Bootstrap AGGregating): It leverages decision trees to reduce variance, ultimately improving predictive performance and mitigating overfitting issues commonly encountered in machine learning models.
         - Bagging is advantageous since weak base learners are combined to form a single strong learner that is more stable than single learners.
         - It also eliminates any variance, thereby reducing the overfitting of models. One limitation of bagging is that it is computationally expensive.
         - Algorithms are
            i. Random Forest Classifier and
            ii. Random Forest regression
      b) **Boosting**
         - This approach organizes weak learners sequentially, allowing each one to learn from its predecessors to build better models.
         - There are various forms of boosting algorithms, such as AdaBoost, gradient boosting, and XGBoost.

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

Aspect	Bagging	Boosting
History	Bagging was introduced by Leo Breiman in 1996. Though Random Forest coined in 2001.	Boosting's foundation was laid by Robert Schapire in 1990. Key developments occurred with the introduction of AdaBoost in 1996.
Objective	Reduce variance by combining diverse models.	Reduce bias by iteratively improving weak learners.
Training	Models trained independently on random subsets with replacement.	In boosting technique, the data for the training is resampled and combined in an adaptive manner such that the weights in the resampling are increased for those data points which got misclassified more often.
Weighting	Equal weighting of models during aggregation (voting or averaging).	Weighted is assigned to the models based on their performance.
Ensemble Size	Typically larger ensembles for better performance.	Smaller ensembles due to emphasis on model improvement.
Overfitting	Less prone to overfitting due to averaging/voting of diverse models.	If boosting is applied too aggressively or the number of iterations is too high, it can lead to overfitting. The model may become too complex and memorize the training data, including noise.

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

A. Definition:

- The OOB error is the average error for each data point calculated using predictions from the trees that do not contain that data point in their respective bootstrap sample.
- In other words, it estimates how well the random forest model generalizes to unseen data without the need for a separate validation set.

B. How It Works:

During the construction of each decision tree in the random forest:

- A bootstrap sample (randomly selected subset) of the training data is used to build the tree.
- The remaining data points (not included in the bootstrap sample) are used for validation.
- The OOB error is computed by evaluating the predictions of the tree on these out-of-bag data points.

C. Advantages:

- The OOB error provides an unbiased estimate of the model's performance without the need for cross-validation.
- It helps assess the effectiveness of the random forest while training.

In summary, the OOB error allows random forests to be validated during training, making it a useful metric for evaluating model performance.

9. What is K-fold cross-validation?

K-fold cross-validation is a statistical method used to estimate the skill of machine learning models. It is commonly used in applied machine learning to compare and select a model for a given predictive modeling problem.

Here are the key points about k-fold cross-validation:

A. Definition:
   - K-fold cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.
   - It helps estimate how well a model will perform on unseen data by repeatedly splitting the dataset into training and validation subsets.

B. Procedure:
   - The dataset is divided into k subsets (folds) of approximately equal size.
   - The model is trained on k-1 folds and validated on the remaining fold.
   - This process is repeated k times, with each fold serving as the validation set once.
   - The average performance across all folds provides an estimate of the model's generalization performance.

C. Advantages:
   - Provides a more reliable estimate of model performance than a single train/test split.
   - Helps detect overfitting and assess model stability.
   - Utilizes the entire dataset for both training and validation.

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

A. What are Hyperparameters?

- In machine learning, hyperparameters are external configuration variables that control the learning process of a model.
- Hyperparameter tuning is a crucial process in machine learning that involves adjusting the settings or hyperparameters of a model to optimize its performance.
- Unlike model parameters (such as weights and biases), which are learned from the data, hyperparameters are manually set before training a model.
- Examples of hyperparameters include the learning rate, the number of layers in a neural network, and the kernel size in a support vector machine.

B. Why Hyperparameter Tuning?

- Model Performance Optimization: Hyperparameter tuning allows data scientists to tweak model performance for optimal results.
- Essential for Success: Choosing appropriate hyperparameter values is crucial for the success of a machine learning model.
- Impact on Model Behavior: For example, consider the learning rate as a hyperparameter. If it's too high, the model may converge too quickly with suboptimal results. If it's too low, training takes too long, and results may not converge.

C. How It Works:

- Hyperparameter tuning involves adjusting the hyperparameters to find the optimal combination that leads to the best model performance.
- This process is also known as hyperparameter optimization.
- Data scientists experiment with different hyperparameter values through multiple training runs to evaluate their impact on model performance.

D. Methods for Hyperparameter Tuning:

- Manual Tuning: Data scientists manually select hyperparameter values based on intuition and domain knowledge.
- Automated Methods:
  - Grid Search: Exhaustively searches through a predefined set of hyperparameter values.
  - Random Search: Randomly samples hyperparameter values from a distribution.
  - Bayesian Optimization: Uses probabilistic models to guide the search process.
  - Genetic Algorithms: Inspired by natural selection, evolves a population of hyperparameter sets.

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

A. When using Gradient Descent with a large learning rate, several issues can arise:

a) Overshooting the Minimum:
   - A large learning rate (step size) can cause the algorithm to take huge steps in the direction of the gradient.
   - If the learning rate is too large, it might overshoot the minimum point of the loss function.
   - As a result, the algorithm may diverge instead of converging to the optimal solution.

b) Oscillations and Instability:
   - Large learning rates can lead to oscillations around the minimum.
   - The algorithm may keep bouncing back and forth, unable to settle down.
   - This instability prevents the model from finding the true minimum.

c) Slow Convergence:
   - Surprisingly, a very large learning rate can slow down convergence.
   - The algorithm might keep overshooting and oscillating, taking longer to reach the minimum.

d) Overflow or Underflow:
   - Numerical issues can occur due to overflow (very large values) or underflow (very small values).
   - For example, if the gradients are large, the parameter updates can become too extreme.

B. Solution:

   - Tune the Learning Rate: Experiment with different learning rates to find an optimal value.
   - Use Adaptive Learning Rates: Techniques like Adam, Adagrad, or RMSProp adaptively adjust the learning rate during training.
   - Monitor Loss and Validation Performance: Observe the loss function and validation performance during training to detect issues.

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

Yes, in theory, Logistic Regression can be extended to handle non-linear data.

Explanation:

Logistic regression is primarily linear; it can be extended to handle non-linear data using polynomial features or other techniques.

While logistic regression is fundamentally a linear model, it can be adapted to handle non-linear data to some extent. This can be achieved by incorporating polynomial features or employing other techniques to transform the input variables, allowing logistic regression to capture non-linear relationships in the data. However, it's important to note that logistic regression may not be as effective for highly complex non-linear patterns compared to more flexible models like decision trees or neural networks.

13. Differentiate between Adaboost and Gradient Boosting

Aspect	**AdaBoost**	**Gradient Boosting**
Type	Ensemble method that combines multiple weak learners (usually decision trees).	Ensemble method that also combines weak learners (usually decision trees) into a strong learner.
Weighted Voting	Each weak learner is assigned a weight based on its performance.	Sequential training of trees, focusing on residuals (errors) from the previous tree.
Training Approach	Sequentially trains weak learners, adjusting weights based on misclassified samples.	Sequentially builds trees, correcting errors of the previous ones (focuses on residuals).
Weak Learners	Typically uses shallow decision trees (stumps).	Can use deeper trees, making it more expressive.
Regularization	Implicitly regularized by sample weights.	Requires explicit regularization (e.g., learning rate, depth).
Prone to Overfitting	Less prone to overfitting due to focus on misclassified samples.	Prone to overfitting without proper regularization.
Performance	May have slightly lower performance compared to Gradient Boosting.	Often performs better than AdaBoost due to regularization and deeper trees.

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

The bias-variance tradeoff is a fundamental concept in machine learning that affects the performance of predictive models.

A. Definition:
   The bias-variance tradeoff refers to the delicate balance between two types of errors that a model can make:
   - Bias Error: The error due to overly simplistic assumptions in the learning algorithm. High bias leads to underfitting.
   - Variance Error: The error due to too much complexity in the learning algorithm. High variance leads to overfitting.

B. Understanding Bias and Variance:
   Bias:
   - High bias occurs when the model is too simple to capture the underlying patterns in the data.
   - It results in poor performance on both training and test data.
   - Bias is the difference between the model's predictions and the true values.

   Variance:
   - High variance occurs when the model is too complex and fits the training data too closely.
   - It leads to excellent performance on training data but poor generalization to unseen data.
   - Variance is the variability of model predictions for different training sets.

C. Tradeoff:
   - As we increase model complexity (e.g., by adding more features or increasing model capacity), bias decreases but variance increases.
   - The goal is to find the right balance:
     a) Low Bias: The model should fit the training data well.
     b) Low Variance: The model should generalize well to unseen data.

D. Practical Implications:
   - Underfitting: High bias, low variance (too simple model).
   - Overfitting: Low bias, high variance (too complex model).
   - Optimal Tradeoff: Choose a model complexity that minimizes the total error (bias + variance).

E. Regularization:
   - Regularization techniques (e.g., L1/L2 regularization) help control model complexity and find an optimal bias-variance tradeoff.

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

A. Linear Kernel:
   - The linear kernel is the simplest and most straightforward.
   - It defines a linear decision boundary in the feature space.
   - Suitable for problems where classes are linearly separable.
   - Often used when the data is approximately linearly separable.

B. Polynomial Kernel:
   - The polynomial kernel introduces non-linearity by mapping the data into a higher-dimensional space.
   - It creates decision boundaries that are polynomial curves (e.g., circles, ellipses).
   - Controlled by the degree parameter, which determines the order of the polynomial.

C. Radial Basis Function (RBF) Kernel:
   - The RBF kernel is also known as the Gaussian kernel.
   - It transforms the data into an infinite-dimensional space using Gaussian radial basis functions.
   - Creates decision boundaries that are smooth and flexible.
   - Suitable for complex, non-linear problems.