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 is generally a better measure of the goodness of fit for a regression model than the residual sum of squares (RSS).

In the context of regression analysis, both R-squared (R^2) and Residual Sum of Squares (RSS) are used as measures of goodness of fit for a model. R^2, the coefficient of determination, indicates the proportion of variance in the dependent variable that can be explained by the regression equation. It is commonly used because it is relatively easy to interpret; the closer to 1.0, the better the model explains the variance in the data.

RSS, on the other hand, is the sum of the squares of the differences between the observed values and the values predicted by the model. Minimizing RSS is the objective of the least-squares criteria. However, while RSS provides an absolute measure, it isn't comparable across models with different sample sizes or numbers of predictors.

Thus, in the context of multiple regression, Adjusted R-squared is often a better measure than both R-squared and RSS because it accounts for the number of predictors in the model, fostering comparison between models with different numbers of independent variables without being biased towards models with more variables.

The standard error of the regression, derived from the square root of the residual mean square is an additional helpful statistic as it offers an average distance that the observed values fall from the regression line.

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 :The Total SS (TSS or SST) tells you how much variation there is in the dependent variable. The formula is:

Total SS = Σ(Yi – mean of Y)2.

which can also be written as

TSS = Σ(Yi – Ȳ)2,

where

Σ = summation

Yi = the actual value of the dependent variable for observation i

Ȳ = the predicted value of the dependent variable for observation i

Σ = the sum of all values.

The higher the TSS, the more variation in the dependent variable. The TSS is a useful measure for comparing models: the model with the lower TSS is the better model, because it is able to explain more of the variation in the dependent variable. The TSS is also a useful measure for determining a model's significance. If the TSS is small, then the model is significant, and it is likely that the model is not due to chance.

The Explained SS tells you how much of the variation in the dependent variable your model explained.

The formula is

$ESS = \Sigma(\bar{Y} – \text{mean of } Y)2$,

where:

$\bar{Y}$ is the predicted value of Y

Y is the observed value of Y

Σ is the sum of all values.

The ESS is a non-negative value, and it can range from zero to the total variation in the dependent variable. The higher the ESS, the better the model is able to explain the variation in the dependent variable. Therefore, the ESS is a useful measure for comparing models. However, the ESS is not the only measure to consider when comparing models. Other measures, such as the residual sum of squares (RSS) and the adjusted R-squared can also be helpful.

4. What is the Residual Sum of Squares?

The residual sum of squares (RSS) helps determine the suitability of a model for your data by measuring the overall difference between your observed data and the values predicted by the estimation model. More specifically, the RSS provides insight into the amount of unexplained variation in the dependent variable. It is calculated as the sum of the squared differences between the actual Y values and the predicted Y values:

$RSS = \Sigma\ e2$

where

e = the difference between the actual and predicted Y value

Σ = the sum of all values.

Manual computation of the formula is time-consuming and prone to errors,. It is much more efficient to rely on software such as R, SAS or Excel, which performs these calculations seamlessly without requiring deep knowledge of the underlying formulas.

A lower RSS indicates a better fit of the model to the data, while a higher value suggests a poor fit. A perfect fit is represented by a residual sum of squares of zero.

One important application is determining the coefficient of determination (R2), which indicates the ratio of the explained sum of squares to the total sum of squares.

3) What is the need of regularization in machine learning?
ANS: Regularization in machine learning serves as a method to forestall a model from over fitting. Over fitting transpires when a model not only discerns the inherent pattern within the training data but also incorporates the noise, potentially leading to subpar performance on fresh, unobserved data.

Role Of Regularization

In Python, Regularization is a technique used to prevent over fitting by adding a penalty term to the loss function, discouraging the model from assigning too much importance to individual features or coefficients.

Let’s explore some more detailed explanations about the role of Regularization in Python:

Complexity Control: Regularization helps control model complexity by preventing over fitting to training data, resulting in better generalization to new data.

Preventing Over fitting: One way to prevent over fitting is to use regularization, which penalizes large coefficients and constrains their magnitudes, thereby preventing a model from becoming overly complex and memorizing the training data instead of learning its underlying patterns.

Balancing Bias and Variance: Regularization can help balance the trade-off between model bias (under fitting) and model variance (over fitting) in machine learning, which leads to improved performance.

Feature Selection: Some regularization methods, such as L1 regularization (Lasso), promote sparse solutions that drive some feature coefficients to zero. This automatically selects important features while excluding less important ones.

Handling Multicollinearity: When features are highly correlated (multicollinearity), regularization can stabilize the model by reducing coefficient sensitivity to small data changes.

Generalization: Regularized models learn underlying patterns of data for better generalization to new data, instead of memorizing specific examples.

4. What is Gini–impurity index?
Ans : The Gini impurity index, also known as the Gini index, is a measure of how mixed up or impure a dataset is. It's often used in decision trees to determine how likely it is to misclassify a randomly selected element from a dataset.

Here are some things to know about the Gini impurity index:

Values

The Gini impurity index ranges from 0 to 1, with 0 representing a pure dataset and 1 representing a completely impure dataset.

Calculation

The Gini impurity index is calculated using the formula GiniIndex=1–∑jp2j, where pj is the probability of class j.

Minimum value

The Gini impurity index reaches its minimum value of 0 when all cases in a node fall into a single target category.

Maximum value

The Gini impurity index reaches its maximum value when the probability of the two classes are the same.

Importance

The features with the lowest Gini index are chosen as the optimum split.

The Gini impurity index is named after Italian mathematician Corrado Gini.

5. Are unregularized decision-trees prone to overfitting? If yes, why?
Ans : Overfitting happens when any learning processing overly optimizes training set error at the cost test error. While it's possible for training and testing to perform equality well in cross validation, it could be as the result of the data being very close in characteristics, which may not be a huge problem. In the case of decision tree's they can learn a training set to a point of high granularity that makes them easily overfit. Allowing a decision tree to split to a granular degree, is the behavior of this model that makes it prone to learning every point extremely well — to the point of perfect classification — ie: overfitting.

I recommend the following steps to avoid overfitting:

Use a test set that is not exactly like the training set, or different enough that error rates are going to be easy to see.

What is different enough? Enough to generalize what is being predicted. The problem should dictate this somewhat because if you are predicting on a baseline that is anomalous, you will need to approximate your validation set close enough. If the problem is more general such as spam classification, having enough data will usually have enough entropy to account for enough variance that should exemplify a disparity is cross validation. Additionally, you can use a one-hold-out dataset for extra assurance. You can be more scientific like using "Minimum Description Length principle" which is something related to the size of the error vs the size of the tree but that's getting a bit in the weeds.

Ensure you have enough data.

It's possible that you don't have enough representitive data (more to my first points in this recommendation). There may not be enough examples to describe a specific case. Perhaps you're classifying houses vs apartments and you don't have enough data on apartments within a given range of values such as square feet and bedrooms, the model may not learn that apartments above 2 bedrooms and 2000 square feet in the city can be either house or apartment but is less likely to be an apartement if there are more houses in the dataset than there are in real life, the decision tree will consider the information gain in this range of variables to describe a split on assumptions it can only conclude with the data it observes. I can't think of a better example…

CHECK FOR CLASS IMBALANCE!

Reduce the complexity of the decision tree model

There's a great quote for this: "Everything should be made as simple as possible, but no simpler."

Pruning a tree can be done before or after, but in sklearn, pre-pruning isn't possible, but you can choose to set the minimum amount of data that is required to create a leaf node, which will additionally qualify the conditions on which a split can occur. Additinally, one can set the max-depth to control the complexity, limiting quantity of nodes quite a bit — you could adjust this paramter and check cross-validation each time after you've gridsearched a range that tends to perform well on the classification metric of your choice.

Use decision trees in an ensemble

Since decision trees are greedy, they are also prone to finding locally optimal solutions without considering a broader assumption about data. This is one reason (in my opinion), that makes these ideal for ensembles. Ensemble methods (bagging / random forrests, boosting, etc) that allows for weighting different aspects of decision trees (with bootstrapping, with random variable selection, with weighting of weak learners, with sample weight selection.. these are the main aspects that different ensembles mix and match to generalize better)

Reduce the dimensionality of your data

Additionally, choose better features. Reducing the complexity of a model can also be done if you reduce the complexity of your data. The potential splits (ie: complexity), can be reduced before you even put your data to the model.

6. What is an ensemble technique in machine learning?

ANS : In machine learning, ensemble techniques combine multiple models to improve overall performance. Ensemble techniques are based on the idea that a group of models is more accurate than a single model. Some ensemble techniques include:

Weighted voting

Allows you to increase the importance of some models by counting their predictions multiple times.

Bagging

A technique that involves pulling bootstrapped subsamples from a larger dataset and forming a decision tree on each sample.

Random forests

A technique that's similar to bagging, but instead of splitting at similar features, each tree splits based on a random selection of features.

Pasting

Similar to bagging, but without replacement when sampling the training dataset. This results in less diversity in the sampled datasets, and the data ends up being correlated.

Ensemble techniques can be applied to various machine learning tasks, including classification, regression, clustering, and anomaly detection.

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

ANS: Bagging is a learning approach that aids in enhancing the performance, execution, and precision of machine learning algorithms. Boosting is an approach that iteratively modifies the weight of observation based on the last classification. 2. It is the easiest method of merging predictions that belong to the same type.

8. What is out-of-bag error in random forests?
ANS: The out-of-bag (OOB) error is a way to measure the prediction error of a random forest model. It's a useful tool for machine learning professionals and data scientists because it provides an accurate estimate of model performance without the need for cross-validation.

Here's how the OOB error works:

Bootstrap aggregating

Bagging is a method that uses subsampling with replacement to create training samples for the model to learn from.

OOB error calculation

The OOB error is the average error for each training sample, calculated using predictions from the trees that did not contain that sample in their bootstrap sample.

The OOB error is often used to evaluate the accuracy of a random forest and to select appropriate values for tuning.

9. What is K-fold cross-validation?

ANS: K-fold cross-validation is a technique for evaluating machine learning models by splitting a dataset into subsets, or folds, and using each fold as a training or testing set:

Divide the dataset into k equally-sized subsets

For each iteration, use one subset as the test set and the remaining k-1 subsets as the training set

Repeat the process k times, so that each subset is used as the test set exactly once

Average the k results to get the final result

K-fold cross-validation is useful because it:

Uses the entire dataset: The entire dataset is used for training and validation, and every sample is used for validation exactly once

Reduces bias: It reduces the bias associated with randomly shuffling data into training and test sets

Ensures generalizability: It helps ensure that the model generalizes well to unseen data

When choosing a value for k, it's important to consider the data sample. A poorly chosen value can lead to a misrepresentative idea of the model's skill. Some common ways to choose a value for k include:

Representative

Choose a value of k so that each train/test group is large enough to be statistically representative of the dataset

k=10

This value has been found to generally result in a model skill estimate with low bias and modest variance

k=n

Set k to the size of the dataset, which is called leave-one-out cross-validation

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

ANS: Hyper parameters are model arguments that are set before the learning process starts. They control the model training process and can have a big impact on the model's performance. For example, in a neural network, hyper parameters include the number of hidden layers and the number of nodes in each layer.

Different datasets and models require different hyperparameters, and the best hyperparameters can't be known in advance. Hyperparameter tuning is important because it helps ensure that the model performs well and produces the best results.

11. What issues can occur if we have a large learning rate in Gradient Descent?
ANS: When the learning rate is too large, gradient descent can suffer from divergence. This means that weights increase exponentially, resulting in exploding gradients which can cause problems such as instabilities and overly high loss values

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

ANS: Logistic regression assumes a linear relationship between the input features and the target variable. This means it may not perform that well when the relationship between the feature and outcomes is non-linear (This is where the neural network comes in).

13. Differentiate between Adaboost and Gradient Boosting.

ANS: Gradient Boosting – It is a boosting technique that builds a final model from the sum of several weak learning algorithms that were trained on the same dataset. It operates on the idea of stagewise addition. The first weak learner in the gradient boosting algorithm will not be trained on the dataset; instead, it will simply return the mean of the relevant column. The residual for the first weak learner algorithm's output will then be calculated and used as the output column or target column for the next weak learning algorithm that will be trained. The second weak learner will be trained using the same methodology, and the residuals will be computed and utilized as an output column once more for the third weak learner, and so on until we achieve zero residuals. The dataset for gradient boosting must be in the form of numerical or categorical data, and the loss function used to generate the residuals must be differential at all times.

Adaboost – AdaBoost is a boosting algorithm that also works on the principle of the stagewise addition method where multiple weak learners are used for getting strong learners. The value of the alpha parameter, in this case, will be indirectly proportional to the error of the weak learner, Unlike Gradient Boosting in XGBoost, the alpha parameter calculated is related to the errors of the weak learner, here the value of the alpha parameter will be indirectly proportional to the error of the weak learner.

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

ANS: The bias-variance tradeoff is a fundamental concept in machine learning that refers to the balance between a model's bias and variance. Bias and variance are interdependent, meaning that an increase in one component will usually result in a decrease in the other.

15. Give short description each of Linear, RBF, Polynomial kernels used in SVM.
ANS: he most common SVM kernels are linear, good for straight-line data, polynomial, and useful for curves. Radial basis function (RBF), is great for complex patterns. Also, sigmoid can handle different kinds of data changes.

1. Linear Kernel

The linear kernel is the simplest and is used when the data is linearly separable.

It calculates the dot product between the feature vectors.

2. Polynomial Kernel

The polynomial kernel is effective for non-linear data.

It computes the similarity between two vectors in terms of the polynomial of the original variables.

3. Radial Basis Function (RBF) Kernel

The RBF kernel is a common type of Kernel in SVM for handling non-linear decision boundaries.

It maps the data into an infinite-dimensional space.