MACHINE LEARNING

Question 1

R-squared or Residual Sum of Squares (RSS) which one of these two is a better measure ofgoodness of fit model in regression and why?

Answer 1

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

R-squared, denoted as R^2, is a statistical measure that represents the proportion of the variance for the dependent variable that's explained by the independent variables in the model. It is dimensionless and ranges from 0 to 1, where a value closer to 1 indicates a better fit. R^2 is calculated using the formula:

$R^2 = 1 - \frac{RSS}{TSS}$

where RSS is the residual sum of squares, and TSS is the total sum of squares. TSS represents the total variance in the dependent variable.

The RSS, on the other hand, is the sum of the squared differences between the observed actual outcomes and the outcomes predicted by the regression model. It is calculated as:

$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

wherey_i is the actual value and $\hat{y}_i$ is the predicted value from the model for the $i\text{-th}$ observation.

The reason why R^2 is often preferred over RSS as a measure of goodness of fit is due to its standardized nature:

1. Scalability: R^2 is scale-invariant, meaning it does not change if the scale of the data changes, whereas RSS is affected by the scale of the dependent variable. This makes R^2 a better choice when comparing models fitted on different scales.

2. Interpretability: R^2 has an intuitive interpretation as the proportion of variance explained, which is easier to understand than the sum of squared residuals. An R^2 of 0.75 means that 75% of the variance in the dependent variable is explained by the model, which is a straightforward interpretation.

3. Benchmarking: R^2 provides a clear benchmark. An R^2 of 0 indicates that the model explains none of the variability in the response data around its mean, while an R^2 of 1 indicates that the model explains all the variability.

4. Adjustment for model complexity: Adjusted R^2 takes into account the number of predictors in the model, which helps in assessing whether the addition of a new predictor really improves the model or is just adding complexity without significantly improving the fit.

Question 2

What are TSS (Total Sum of Squares), ESS (Explained Sum of Squares) and RSS (Residual Sumof Squares) in regression. Also mention the equation relating these three metrics with each other.

Answer 2

TSS (Total Sum of Squares) in Regression:

The sum of squares total (SST) or the total sum of squares (TSS) is the sum of squared differences between the observed dependent variables and the overall mean. Think of it as the dispersion of the observed variables around the mean, similar to the variance in descriptive statistics.

TSS is the sum of square of difference of each data point from the mean value of all the values of target variable (y). here, the line is with intercept (‘c’ in y = mx + c) equal to Ymean; it means that this line does not include any influence of independent variable.

ESS (Explained Sum of Squares) in Regression:

The sum of squares due to regression (SSR) or explained sum of squares (ESS) is the sum of the differences between the predicted value and the mean of the dependent variable. In other words, it describes how well our line fits the data.

The explained sum of squares (ESS) is the sum of the squares of the deviations of the predicted values from the mean value of a response variable, in a standard regression model. For example, $y_i = a + b_1x1_i + b_2x_{2i} + \ldots\ldots$

RSS (Residual Sumof Squares) in Regression:

The residual sum of squares (RSS) measures the level of variance in the error term, or residuals, of a regression model. The smaller the residual sum of squares, the better your model fits your data; the greater the residual sum of squares, the poorer your model fits your data.

<u>Equation relating these three metrics with each other:</u>

TSS (Total Sum of Squares) = ESS (Explained Sum of Square) + RSS (Residual Sum of Square)

<u>Question 3</u>

What is the need of regularization in machine learning?

<u>Answer 3</u>

The primary goal of regularization is to reduce the model's complexity to make it more generalizable to new data, thus improving its performance on unseen datasets.

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

Below are some detailed explanations about the role of Regularization in Python:

1. Complexity Control:

   Regularization helps to control model complexity by preventing overfitting to train data, resulting in better generalization to new data.

2. Preventing Overfitting:

   One way to prevent overfitting 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.

3. Balancing Bias and Variance:

   Regularization can help balance the trade-off between model bias (underfitting) and model variance (overfitting) in machine learning, which leads to improved performance.

4. 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.

5. Handling Multicollinearity:

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

6. Generalization:

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

Question 4

What is Gini–impurity index?

Answer 4

Gini Impurity is a measurement of the likelihood of an incorrect classification of a new instance of data, if that new instance were randomly classified according to the distribution of class labels from the data set.

The Gini Index is the additional approach to dividing a decision tree. Purity and impurity in a junction are the primary focus of the Entropy and Information Gain framework. The Gini Index, also known as Impurity, calculates the likelihood that somehow a randomly picked instance would be erroneously catalogued.

Question 5

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

Answer 5

Yes. Unregularized decision-trees prone to overfitting.

Overfitting can be one problem that describes if your model no longer generalizes well.

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 point a high granularity that makes then easily overfit. Allowing a decision tree to split to a granular degree, is the behaviour of this model that makes it prone to learning every point extremely well to the point of perfect classification i.e.: overfitting.

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 representative 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 apartment 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.

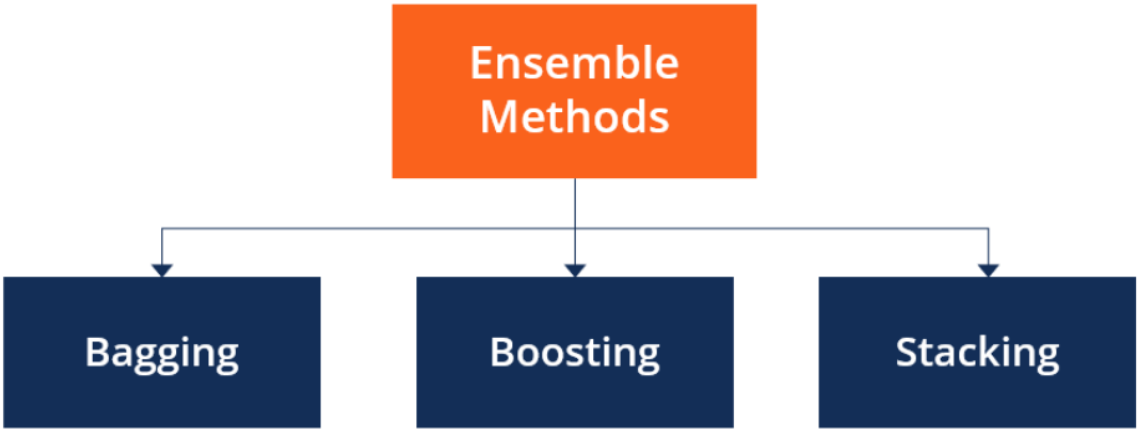
Question 6

What is an ensemble technique in machine learning?

Answer 6

Ensemble methods are techniques that create multiple models and then combine them to produce improved results. Ensemble methods in machine learning usually produce more accurate solutions than a single model would.

Ensemble methods are techniques that aim at improving the accuracy of results in models by combining multiple models instead of using a single model. The combined models increase the accuracy of the results significantly. This has boosted the popularity of ensemble methods in machine learning.



Question 7

What is the difference between Bagging and Boosting techniques?

Answer 7

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.

Bagging Combines multiple models trained on different subsets of data. Boosting Train models sequentially, focusing on the error made by the previous model.

The main objective of bagging technique is to reduce variance by averaging out individual model error and of boosting technique is to Reduces both bias and variance by correcting misclassifications of the previous model.

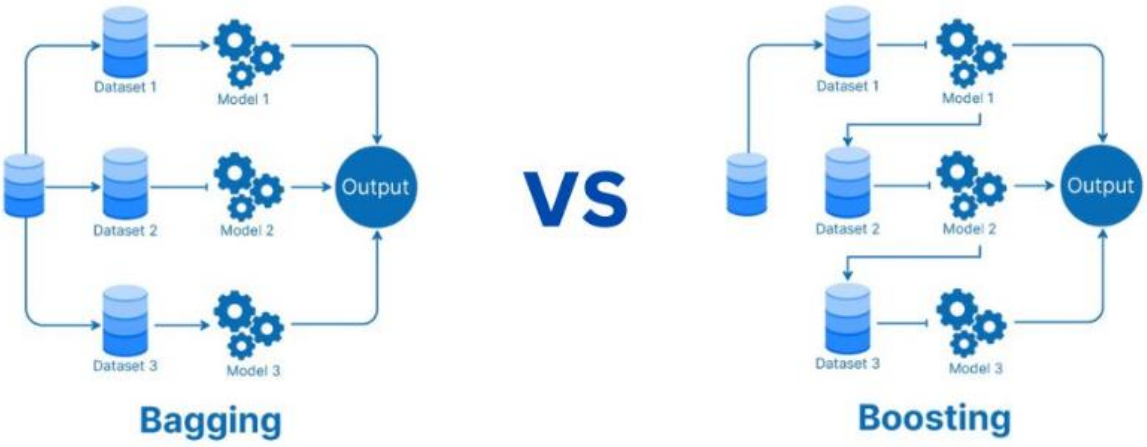
Bagging use Bootstrap to create subsets of the data. Boosting re-weights the data based on the error from the previous model, making the next models focus on misclassified instances.

In bagging each model serves equal weight in the final decision. In boosting models are weighted based on accuracy, i.e., better-accuracy models will have a higher weight.

In bagging each model has an equal error rate. In boosting It gives more weight to instances with higher error, making subsequent model focus on them.

Bagging technique is less prone to overfitting due to average mechanism. Boosting technique is generally not prone to overfitting, but it can be if the number of the model or the iteration is high.

Bagging technique improves accuracy by reducing variance. Boosting technique achieves higher accuracy by reducing both bias and variance.



Question 8

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

Answer 8

The out-of-bag (OOB) error is the average error for each calculated using predictions from the trees that do not contain in their respective bootstrap sample. This allows the RandomForestClassifier to be fit and validated whilst being trained.

Out-of-bag (OOB) error, also called out-of-bag estimate, is a method of measuring the prediction error of random forests, boosted decision trees, and other machine learning models utilizing bootstrap aggregating (bagging).

Question 9

What is K-fold cross-validation?

Answer 9

K-fold cross-validation splits data into k equal parts; each part serves as a test set while the others form the training set, rotating until every part has been tested.

In K-fold cross-validation, the data set is divided into a number of K-folds and used to assess the model's ability as new data become available. K represents the number of groups into which the data sample is divided. For example, if you find the k value to be 5, you can call it 5-fold cross-validation.

Question 10

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

Answer 10

When you'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 these is through multiple experiments, where you pick a set of hyperparameters and run them through your model. This is called hyperparameter tuning.

Hyperparameter optimization finds a tuple of hyperparameters that yields an optimal model which minimizes a predefined loss function on given independent data. The objective function takes a tuple of hyperparameters and returns the associated loss.

Setting the right hyperparameter values is very important because it directly impacts the performance of the model that will result from them being used during model training.

Question 11

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

Answer 11

If the learning rate is too high, the algorithm may overshoot the minimum, and if it too low, the algorithm may take too long to converge. Overfitting: Gradient descent can overfit the training data if the model is too complex or the learning rate to too high.

Too small learning rate will lead to very slow learning or even inability to learn at all, while too large learning rate can lead to exploding or oscillating performance over the training epochs and to a lower final performance.

A too high learning rate will make the learning jump over minima but a too low learning rate will either take too long to converge or get stuck in an undesirable local minimum.

Question 12

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

Answer 12

No. We cannot use Logistic Regression for classification of Non-Linear Data.

Logistic regression is simple and easy to implement, but it also has some drawbacks. One of them is that it assumes a linear relationship between the input features and the output. This means that it cannot capture the complexity and non-linearity of the data.

Logistic regression has traditionally been used to come up with a hyperplane that separates the feature space into classes. But if we suspect that the decision boundary is nonlinear we may get better results by attempting some nonlinear functional forms for the logit function.

By employing the logistic function, which maps any real-valued number into a range between 0 and 1, logistic regression models can effectively handle non-linear relationships.

Question 13

Differentiate between Adaboost and Gradient Boosting.

Answer 13

Gradient boosting and AdaBoost are both ensemble learning techniques that combine multiple weak learners to create a strong learner. While they share some similarities, they differ in the way they update the weights of the training instances and how they build the ensemble model.

Following are the key differences between Gradient Boosting and AdaBoost:

- ***Update Strategy:***
  - AdaBoost (Adaptive Boosting): In AdaBoost, the algorithm assigns weights to the training instances and focuses on the misclassified instances in each iteration. It increases the weight of misclassified instances so that subsequent weak learners focus more on these instances.
  - Gradient Boosting: In gradient boosting, the algorithm fits the new weak learner to the residual errors made by the existing ensemble. It uses the gradient of the loss function with respect to the predictions of the ensemble to update the model in the direction that minimizes the loss.
- ***Loss Function:***
  - AdaBoost: AdaBoost typically uses the exponential loss function, which puts more emphasis on the misclassified instances.
  - Gradient Boosting: Gradient boosting can work with various loss functions, such as squared error loss (for regression problems) or log

loss (for classification problems). The choice of loss function can be tailored to the specific problem being addressed.

- ***Weak Learners:***

  - AdaBoost: AdaBoost focuses on building a sequence of weak learners, where each weak learner tries to correct the mistakes of the previous ones.

  - Gradient Boosting: Gradient boosting typically uses decision trees as weak learners, building trees sequentially where each new tree is trained to predict the residual errors of the existing ensemble.

- ***Learning Rate:***

  - AdaBoost: AdaBoost uses a learning rate parameter that controls the contribution of each weak learner to the final prediction.

  - Gradient Boosting: Gradient boosting also uses a learning rate, but it is typically lower than in AdaBoost. The learning rate in gradient boosting scales the contribution of each tree, helping to prevent overfitting.

In summary, while both AdaBoost and gradient boosting are ensemble learning methods that aim to improve model performance by combining weak learners, they differ in their update strategies, loss functions, weak learners used, and how they build the ensemble model. Gradient boosting, with its focus on minimizing the loss function using gradients, is often more flexible and can be tailored to different types of problems.

Question 14

What is bias-variance trade off in machine learning?

Answer 14

In statistics and machine learning, the bias-variance trade off describes the relationship between a model's complexity, the accuracy of its predictions, and how well it can make predictions on previously unseen data that were not used to train the model.

In machine learning, as you try to minimize one component of the error (e.g., bias), the other component (e.g., variance) tends to increase, and vice versa. Finding the right balance of bias and variance is key to creating an effective and accurate model. This is called the bias-variance trade off.

The equation $\varepsilon = [g(X) - f(X')] + \delta$ or error = bias + variance represents the bias variance tradeoff for some model f. Holding the error fixed, decreasing the bias means then the model variance increases.

A model with high bias is too simple and cannot capture the genuine relationship between the input and output variables. On the other hand, a model with high variance is too complex and captures the random noise in the data, resulting in poor generalisation of new data.

Question 15

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

Answer 15

RBF in SVM

In machine learning, the radial basis function kernel, or RBF kernel, is a popular kernel function used in various kernelized learning algorithms. In particular, it is commonly used in support vector machine classification.

Linear in SVM

Decision Boundary: Form: The linear kernel produces a decision boundary that is a hyperplane in the feature space. This hyperplane separates data points from different classes in a linear fashion. Assumption: It assumes that the relationship between the features and the target variable is linear.

Polynomial in SVM

In machine learning, the polynomial kernel is a kernel function commonly used with support vector machines (SVMs) and other kernelized models, that represents the similarity of vectors (training samples) in a feature space over polynomials of the original variables, allowing learning of non-linear models.

DS2406