MACHINE LEARNING

1 What is Gini–impurity index?

The **Gini Index**, also known as **Gini Impurity**, assists the CART algorithm in identifying the most suitable feature for node splitting during the construction of a decision tree classifier[1]

It derives its name from the Italian mathematician Corrado Gini.

Gini Impurity is a method that measures the impurity of a dataset. The more impure the dataset, the higher is the Gini index.

The term Impurity indicates the number of classes present within a subset. The more distinct classes included in a subset, the higher the impurity.

2 WHAT IS K-FOLD CROSS-VALIDATION?

implement, and results in 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 because it is easy to understand, easy to skill estimates that generally have a lower bias than other methods.

In this tutorial, you will discover a gentle introduction to the k-fold cross-validation procedure for estimating the skill of machine learning models.

After completing this tutorial, you will know:

- **That k-fold cross validation is a procedure used to estimate the skill of the model on new data.**
- **There are common tactics that you can use to select the value of k for your dataset.**
- **There are commonly used variations on cross-validation such as stratified and repeated that are available in scikit-learn.**

3 What is bias-variance trade off in machine learning?

it is important to understand prediction errors (bias and variance) when it comes to accuracy in any machine-learning algorithm. There is a tradeoff between a model's ability to minimize bias and variance which is referred to as the best solution for selecting a value of **Regularization** constant. A proper understanding of these errors would help to avoid the overfitting and underfitting of a data set while training the algorithm.

1

4 What is the need of regularization in machine learning?

While developing machine learning models you must have encountered a situation in which the training accuracy of the model is high but the validation accuracy or the testing accuracy is low. This is the case which is popularly known as overfitting in the domain of Machine learning Also, this is the last thing a machine learning practitioner would like to have in his model. In this article, we will learn about a method known as Regularization in Python which helps us to solve the problem of overfitting. But before that let's understand what is the role of regularization in Python and what is underfitting and overfitting.

Role Of Regularization

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.

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

1. **Complexity Control**: Regularization helps control model complexity by preventing overfitting to training 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.

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

The out-of-bag error is the average error for each predicted outcome calculated using predictions from the trees that do not contain that data point in their respective bootstrap sample. This way, the Random Forest model is constantly being validated while being trained. Let us consider the decision tree DTj that has been fitted on a subset of the sample data. For every training observation or sample $zi=(xi,yi)$ not in the sample subset of DTj where xi is the set of features and yi is the target, we use DTj to predict the outcome oi for xi. The error can easily be computed as $|oi-yi|$.

The out-of-bag error is thus the average value of this error across all decision trees.

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

A Machine Learning model is defined as a mathematical model with several parameters that need to be learned from the data. By training a model with existing data, we can fit the model parameters. However, there is another kind of parameter, known as ***Hyperparameters***, that cannot be directly

learned from the regular training process. They are usually fixed before the actual training process begins. These parameters express important properties of the model such as its complexity or how fast it should learn. This article aims to explore various strategies to tune hyperparameters for Machine learning models.
Hyperparameter tuning is the process of selecting the optimal values for a machine learnings model's hyperparameters. Hyperparameters are settings that control the learning process of the model, such as the learning rate, the number of neurons in a neural network, or the kernel size in a support vector machine. The goal of hyperparameter tuning is to find the values that lead to the best performance on a given task.

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

A learning rate that is too large can cause the model to converge too quickly to a suboptimal solution, whereas a learning rate that is too small can cause the process to get stuck the learning rate controls how quickly the model is adapted to the problem. Smaller learning rates require given the smaller changes made to the weights each update, whereas larger learning rates result in rapid changes and require fewer training . the learning rate is a configurable hyperparameter used in the training of neural networks that has a small positive value, often in the range between 0.0 and 1.0. the challenge of training deep learning neural networks involves carefully selecting the learning rate. It may be the most important hyperparameter for the model. The learning rate is a hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated. Choosing the learning rate is challenging as a value too small may result in a long training process that could get stuck, whereas a value too large may result in learning a sub-optimal set of weights too fast or an unstable training process.
The learning rate may be the most important hyperparameter when configuring your neural network. it is vital to know how to investigate the effects of the learning rate on model performance and to build an intuition about the dynamics of the learning rate on model behavior.
In this tutorial, you will discover the effects of the learning rate, learning rate schedules, and adaptive learning rates on model performance.

8 Differentiate between Adaboost and Gradient Boosting

- **Boosting** is a general method that iteratively trains models to focus more on instances that previous models misclassified. Many boosting algorithms exist like Gradient Boosting, XGBoost, LightGBM, CatBoost, etc.

- **AdaBoost** was the first successful boosting technique developed specifically for binary classification problems. It assigns weights to training instances, focusing subsequent models more on misclassified instances from previous iterations.

In both cases, models are added sequentially until a stopping condition is met. The key advantage is reducing bias and variance to improve predictive performance.

The most significant difference is that gradient boosting minimizes a loss function like MSE or log loss while AdaBoost focuses on instances with high error by adjusting their sample weights adaptively.

Gradient boosting models apply shrinkage to avoid overfitting which AdaBoost does not do. Gradient boosting also performs subsampling of the training instances while AdaBoost uses all instances to train every weak learner.

Overall gradient boosting is more robust to outliers and noise since it equally considers all training instances when optimizing the loss function. AdaBoost is faster but more impacted by dirty data since it fixates on hard examples.

Gradient boosting and AdaBoost are both powerful ensemble machine learning algorithms based on boosting techniques. When comparing their performance, gradient boosting generally achieves higher accuracy.

9 What is the need of regularization in machine learning?

The need to regularize a model will tend to be less and less as you increase the number of samples that you want to train the model with or your reduce the model's complexity. However, the number of examples needed to train a model without (or with a very very small regularization effect) increases [super]exponentially with the number of parameters and possibly some other factors inherit in a model.

Since in most machine learning problems, we do not have the required number of training samples **or** the model complexity is large we have to use regularization in order to avoid, or lessen the possibility, of over-fitting. Intuitively, the way regularization works is it introduces a penalty term to $\text{argmin}\sum L\ (\text{desired},\text{predictionFunction}(Wx))$ where L is a loss function that computes how much the model's prediction deviates from the desired targets. So the new loss function becomes $\text{argmin}\sum L(\text{desired},\text{predictionFunction}(Wx))\ +\ \text{lambda}*\text{reg}(w)$ where reg is a type of regularization (e.g. $\text{squared}\ L2$) and lambda is a coefficient that controls the regularization effect. Then, naturally, while minimizing the cost function the weight vectors are restricted to have a small squared length (e.g. $\text{squared}\ L2\ \text{norm}$) and shrink towards zero. This is because the larger the squared length of weight vectors, the higher the loss is. the weight vectors also need to compensate for lowering the model's loss while the optimization is running.
Now imagine if you remove the regularization term (lambda = 0). Then the model parameters are free to have any values and so do the squared length of weight vectors can grow no matter you have a linear or non-linear model. This adds another dimension to the complexity of the model (in addition to the number of parameters) and the optimization procedure may find weight vectors that can exactly match the training data points. However, when exposed to unseen (validation or test) data sets the model will not be able to generalize well since it has over-fitted to the training data.

10 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

---- TSS (Total Sum of Squares)
he sum of squares (SS) is a statistic that measures the variability of a dataset's observations around the mean. It's the cumulative total of each data point's squared difference from the mean.

----- ESS (Sum of Squares)
explained sum of square (ESS) or Regression sum of squares or Model sum of squares is a statistical quantity used in modeling of a process. ESS gives an estimate of how well a model explains the observed data for the process. It tells how much of the variation between observed data and predicted data is being explained by the model proposed.

------ RSS (Residual Sum of Squares)

Residual Sum of Squares (RSS) is a statistical method that helps identify the level of discrepancy in a dataset not predicted by a regression model. Thus, it measures the variance in the value of the observed data when compared to its predicted value as per the regression model. Hence, RSS indicates whether the regression model fits the actual dataset well or not.

11 What is the difference between Bagging and Boosting techniques?

= As we know, Ensemble learning helps improve machine learning results by combining several models. This approach allows the production of better predictive performance compared to a single model. Basic idea is to learn a set of classifiers (experts) and to allow them to vote. **Bagging** and **Boosting** are two types of **Ensemble Learning**. These two decrease the variance of a single estimate as they combine several estimates from different models. the result may be a model with higher stability. Let's understand these two terms in a glimpse.

1. **Bagging**: It is a homogeneous weak learners' model that learns from each other independently in parallel and combines them for determining the model average.
2. **Boosting**: It is also a homogeneous weak learners' model but works differently from Bagging. In this model, learners learn sequentially and adaptively to improve model predictions of a learning algorithm.