MACHINE LEARNING

1. Which of the following in sk-learn library is used for hyper parameter tuning? (**A**)
   A) GridSearchCV() B) RandomizedCV()
   C) K-fold Cross Validation D) All of the above

2. In which of the below ensemble techniques trees are trained in parallel? (**A**)
   A) Random forest B) Adaboost
   C) Gradient Boosting D) All of the above

3. In machine learning, if in the below line of code: (**B**)
   *sklearn.svm.**SVC** (C=1.0, kernel='rbf', degree=3)* we increasing the C hyper parameter, what will happen?
   A) The regularization will increase B) The regularization will decrease
   C) No effect on regularization D) kernel will be changed to linear

4. Check the below line of code and answer the following questions:
   *sklearn.tree.**DecisionTreeClassifier**(*criterion='gini',splitter='best',max_depth=None, min_samples_split=2)* (**A**)
   Which of the following is true regarding max_depth hyper parameter?
   A) It regularizes the decision tree by limiting the maximum depth up to which a tree can be grown.
   B) It denotes the number of children a node can have.
   C) both A & B
   D) None of the above

5. Which of the following is true regarding Random Forests? (**C**)
   A) It's an ensemble of weak learners.
   B) The component trees are trained in series
   C) In case of classification problem, the prediction is made by taking mode of the class labels predicted by the component trees.
   D)None of the above

6. What can be the disadvantage if the learning rate is very high in gradient descent? (**A**)
   A) Gradient Descent algorithm can diverge from the optimal solution.
   B) Gradient Descent algorithm can keep oscillating around the optimal solution and may not settle.
   C) Both of them
   D) None of them

7. As the model complexity increases, what will happen? (**B**)
   A) Bias will increase, Variance decrease B) Bias will decrease, Variance increase
   C)both bias and variance increase D) Both bias and variance decrease.

8. Suppose I have a linear regression model which is performing as follows: (**B**)
   Train accuracy=0.95 and Test accuracy=0.75 Which of the following is true regarding the model?
   A) model is underfitting B) model is overfitting
   C) model is performing good D) None of the above

Q9 to Q15 are subjective answer type questions, Answer them briefly.

9. Suppose we have a dataset which have two classes A and B. The percentage of class A is 40% and percentage of class B is 60%. Calculate the Gini index and entropy of the dataset.

Ans) The proportion of class A (p_A) and class B (p_B) can be calculated as:

p_A = 40 / 100 = 0.4
p_B = 60 / 100 = 0.6

Gini = 1 - (p_A^2 + p_B^2)

Substituting the values, we get:

Gini = 1 - (0.4^2 + 0.6^2) = 0.48

So, the Gini index of the dataset is 0.48.

Entropy = - p_A * log2(p_A) - p_B * log2(p_B)

Substituting the values, we get:

Entropy = - 0.4 * log2(0.4) - 0.6 * log2(0.6) = 0.971

So, the entropy of the dataset is 0.971.

10. What are the advantages of Random Forests over Decision Tree?
Ans) Random Forests have several advantages over Decision Trees:

Reduced overfitting: Random Forests are less prone to overfitting than Decision Trees because they use multiple decision trees and reduce the variance of the model by averaging the predictions of these trees.

Better performance: Random Forests typically have better performance than Decision Trees because they are more accurate and robust. They can handle large datasets and high-dimensional feature spaces.

Feature importance: Random Forests can provide feature importance measures that can be used to identify the most relevant features for the classification task.

Robustness to noise: Random Forests are robust to noise and outliers because they use multiple decision trees and average their predictions, which reduces the impact of individual noisy or outlier data points.

Easy to use: Random Forests are easy to use and require minimal parameter tuning compared to other machine learning algorithms.

Overall, Random Forests are a powerful machine learning algorithm that can be used for a wide range of classification and regression tasks and provide several advantages over Decision Trees.

11. What is the need of scaling all numerical features in a dataset? Name any two techniques used for scaling.

Ans) Scaling numerical features is necessary because many machine learning algorithms use distance-based measures to determine the similarity between data points. If the numerical features are not scaled, features with larger magnitude or range will dominate the smaller features, and the algorithm will be biased towards them.Scaling the numerical features to a similar range avoids this bias and ensures that all features contribute equally to the similarity measure.

Two common techniques for scaling numerical features are:

Min-Max Scaling: In this technique, the numerical features are scaled to a fixed range, usually between 0 and 1. The formula to scale a feature x is: x_scaled = (x - min(x)) / (max(x) - min(x))

Standardization: In this technique, the numerical features are transformed to have zero mean and unit variance. The formula to standardize a feature x is: x_standardized = (x - mean(x)) / std(x)

By scaling the numerical features using either of these techniques, we can ensure that they are on a similar scale and avoid any bias towards features with larger magnitude or range.

12. Write down some advantages which scaling provides in optimization using gradient descent algorithm.

Ans) Scaling provides several advantages in optimization using gradient descent algorithm:

Faster convergence: Gradient descent works by iteratively updating the model parameters based on the gradient of the loss function with respect to those parameters. If the features have different scales, the gradient updates for different features will also have different scales, which can cause the optimization process to be slow and converge slowly. Scaling the features to a similar range ensures that the gradient updates are more consistent and can help the algorithm converge faster.

Avoiding numerical overflow/underflow: In gradient descent, it is common to encounter numerical issues such as overflow or underflow if the features are not scaled. This can result in the gradient updates becoming too large or too small, which can make it difficult for the algorithm to converge. Scaling the features can help avoid these numerical issues and ensure that the algorithm is stable.

Better conditioning: If the features have very different scales, the condition number of the Hessian matrix can become large, which can make the optimization problem ill-conditioned. Scaling the features to a similar range can improve the conditioning of the Hessian matrix and make the optimization problem easier to solve.

Improved generalization: Scaling the features can help the algorithm generalize better to new data. If the features are not scaled, the model may learn to rely on features with larger scales, which may not be representative of the underlying relationship between the features and the target variable. Scaling the features can help avoid this issue and ensure that the model learns to rely on all features equally.

Overall, scaling provides several advantages in optimization using gradient descent algorithm and is a common preprocessing step in machine learning.

13. In case of a highly imbalanced dataset for a classification problem, is accuracy a good metric to measure the performance of the model. If not, why?

Ans) In case of a highly imbalanced dataset for a classification problem, accuracy is generally not a good metric to measure the performance of the model. The reason for this is that accuracy does not take into account the class distribution of the dataset, and can be biased towards the majority class.

For example, consider a binary classification problem with 99% of the samples belonging to class A and only 1% belonging to class B. A classifier that always predicts class A will achieve an accuracy of 99%, but it is not useful for the problem at hand because it fails to correctly classify any of the minority class samples. In this case, accuracy is not a good metric to evaluate the performance of the model because it does not reflect the classifier's ability to correctly predict the minority class.

Instead of accuracy, other metrics such as precision, recall, F1-score, or area under the ROC curve (AUC-ROC) are often used to evaluate the performance of models on imbalanced datasets. These

metrics take into account the class distribution and are better suited to evaluate the classifier's ability to correctly predict both the majority and minority classes.

In summary, accuracy is not a good metric to measure the performance of the model in case of a highly imbalanced dataset for a classification problem. Other metrics that consider the class distribution should be used instead.

14. What is “f-score" metric? Write its mathematical formula.
Ans) F-score is a commonly used metric for evaluating the performance of classification models. It is a weighted harmonic mean of precision and recall, which gives equal weight to both metrics. The F-score is defined as the harmonic mean of precision and recall, where precision is the ratio of true positive predictions to the total number of positive predictions and recall is the ratio of true positive predictions to the total number of actual positives in the data.

The mathematical formula for F-score is as follows:

F-score = 2 * ((precision * recall) / (precision + recall))

Where:

Precision = True Positives / (True Positives + False Positives)
Recall = True Positives / (True Positives + False Negatives)
The F-score is a number between 0 and 1, where a score of 1 indicates perfect precision and recall, and a score of 0 indicates poor performance. In practice, the F1-score, which is the F-score computed using a harmonic mean of precision and recall with equal weight, is often used as a single summary statistic for the model's performance.

15. What is the difference between fit(), transform() and fit_transform()?
Ans) In the context of machine learning and data preprocessing, fit(), transform(), and fit_transform() are three commonly used methods for data transformation and model fitting.

fit() is used to compute the necessary parameters or statistics of a preprocessing step or a machine learning model on a training set. For example, when standardizing features, fit() computes the mean and standard deviation of each feature on the training set. These parameters are then used to transform both the training and test sets.

transform() applies the transformation or preprocessing step to the data using the parameters computed by fit(). Continuing with the example of standardizing features, transform() applies the standardization to each feature by subtracting its mean and dividing by its standard deviation. This method is used on the training and test sets separately after the fit() method has been called on the training set.

fit_transform() is a convenience method that combines the fit() and transform() methods into a single step. It first fits the preprocessing step or model to the training set using fit(), and then applies the transformation or preprocessing step to the training set using transform().

FLIP ROBO