MACHINE LEARNING ASSIGNMENT – 7

1. A

2. A

3. A

4. A

5. A, B, C

6. C

7. B

8. B

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?**

The Gini index is a measure of impurity in a dataset, where a value of 0 represents a perfectly pure dataset (i.e., all elements belong to the same class) and a value of 1 represents a completely impure dataset (i.e., elements belong to different classes with equal probability).

To calculate the Gini index for a dataset with two classes A and B, we can use the following formula:

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

where p_A is the proportion of elements in class A and p_B is the proportion of elements in class B.

Given that the percentage of class A is 40% and percentage of class B is 60%, we can calculate the Gini index as follows:

Gini index = 1 - (0.4^2 + 0.6^2) = 1 - (0.16 + 0.36) = 1 - 0.52 = 0.48

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

Entropy = -0.4 * log2(0.4) - 0.6 * log2(0.6) = 0.97

The entropy is 0.97 which means that the data is impure and hard to classify.

10. **What are the advantages of Random Forests over Decision Tree?**

Random Forests are an ensemble method that combines multiple decision trees to improve the overall performance of the model. The main advantages of Random Forests over a single decision tree are:

Improved accuracy: Random Forests average out the errors of multiple decision trees, resulting in a more accurate model.

Reduced overfitting: Decision trees can easily overfit to the training data, especially when the tree becomes deep. Random Forests reduce overfitting by averaging the predictions of multiple decision trees, which are trained on different subsets of the data.

Better handling of missing values and outliers: Random Forests can handle missing values and outliers better than decision trees, as the randomness in the subsets of data and the averaging of predictions can reduce the impact of these issues.

Feature importance: Random Forests provide a way to measure the importance of each feature in the dataset, which can be used for feature selection and engineering.

Handling non-linearity: Random Forests can handle non-linearity in the data better than decision trees, as it can model the interactions between features.

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

Scaling numerical features in a dataset is important because many machine learning algorithms, such as linear and logistic regression, k-nearest neighbors, and support vector machines, are sensitive to the scale of the input features. The features with a larger scale can dominate the objective function and affect the learning process. Scaling the features can help to avoid this problem by transforming the data so that it has the same scale.

There are two commonly used techniques for scaling numerical features:

Min-Max Scaling: Min-Max Scaling scales the data to a fixed range, usually between 0 and 1. It is sensitive to outliers and the extreme values. It is calculated by subtracting the minimum value of the feature and dividing by the range.

Standardization: Standardization scales the data by subtracting the mean and dividing by the standard deviation. The result is that the data has a mean of 0 and a standard deviation of 1. It is not sensitive to outliers but it assumes that the data is normally distributed.

These scaling techniques are important for the algorithms because most of the algorithms are based on Euclidean distance, which is sensitive to the scale of the feature values. Scaling will ensure that the feature values are in the same scale and will not dominate the objective function.

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

Faster convergence: Scaling the features can help to ensure that the gradients have similar scales, which can result in faster convergence of the optimization algorithm.

Improved optimization: Scaling the features can help to ensure that the optimization algorithm is working on similar scales, which can result in a better optimization of the objective function.

Avoiding oscillations: Scaling the features can help to avoid oscillations in the optimization process by making sure the gradients have similar scales, which can result in faster convergence.

Better generalization: Scaling the features can help to improve the generalization of the model, by making sure the optimization algorithm is working on similar scales and not overfitting.

Better handling of missing values and outliers: Scaling the features can help to handle missing values and outliers better, by transforming the data so that it has the same scale, which can reduce the impact of these issues.

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?**

In case of a highly imbalanced dataset, accuracy is not a good metric to measure the performance of the model because it can be misleading. When the classes are imbalanced, the model can achieve a high accuracy by simply predicting the majority class for all instances, without actually learning the underlying patterns in the data.

For example, suppose we have a binary classification problem with a highly imbalanced dataset where 99% of the instances belong to class A and only 1% of the instances belong to class B. A model that always predicts class A will achieve an accuracy of 99%, but it will not be able to correctly classify any instance of class B.

Instead, other metrics such as precision, recall, F1-score, and area under the Receiver Operating Characteristic (ROC) curve (AUC-ROC) are better suited for imbalanced datasets. These metrics take into account both the true positive and false positive rates, which are important for identifying the performance of the model when the classes are imbalanced.

In addition, It's also common to use different evaluation metrics in imbalanced datasets such as precision, recall, F1-score, Receiver Operating Characteristic (ROC) curve, and Area Under the ROC curve (AUC-ROC) which are better suited to evaluate the performance of the model.

14. **What is “f-score" metric? Write its mathematical formula?**

The F1-score, also known as the F-score or F-measure, is a metric that combines precision and recall to provide a single measure of a model's performance. The F1-score is the harmonic mean of precision and recall, with a value of 1 representing a perfect score and a value of 0 representing a model that fails to predict any positive class instances.

The mathematical formula for the F1-score is:

F1-score = 2 * (precision * recall) / (precision + recall)

Where:

precision = true positives / (true positives + false positives)

recall = true positives / (true positives + false negatives)

The F1-score is particularly useful when the classes are imbalanced, as it balances the importance of precision and recall.

It's a balance between precision and recall, where precision is the number of true positive divided by the number of true positive plus false positive, and recall is the number of true positive divided by the number of true positive plus false negatives. It's also called harmonic mean as it's the harmonic mean of precision and recall.

15**. What is the difference between fit(), transform() and fit_transform()?**

fit(): The fit() method is used to calculate the internal parameters of a model or transformer. For example, in the case of a scaler, the fit() method is used to calculate the mean and standard deviation of a dataset, which will be used later to scale the data.

transform(): The transform() method is used to apply a transformation to a dataset using the internal parameters calculated during the fit() step. For example, after fitting a scaler to a dataset, the transform() method can be used to scale the dataset using the calculated mean and standard deviation.

fit_transform(): The fit_transform() method combines the fit() and transform() methods into a single step. This method is useful when you want to fit a model to a dataset and then immediately transform the dataset using the fitted model. For example, you can use the fit_transform() method to fit a scaler to a dataset and then scale the dataset in one step.