WORKSHEET 7 MACHINE LEARNING

Q1 ANS: D

Q2 ANS: A

Q3 ANS: B

Q4 ANS: A

Q5 ANS: C

Q6 ANS: A

Q7 ANS: B

Q8 ANS: B

Q9 ANS:Pprobabilities of each class:

Probability of class A = 0.4

Probability of class B = 0.6

The Gini index is defined as:

Gini index = 1 - (p of A)^2 - (p of B)^2

where p of A and p of B are the probabilities of class A and B, respectively.

After substituting the value ,Gini index = 1 - (0.4)^2 - (0.6)^2 = 0.48

Entropy = - p of A log2 p of A - p of B log2 p of B

Substituting the values, we get:

Entropy = - 0.4 log2 0.4 - 0.6 log2 0.6 = 0.971, Therefore, the Gini index of the dataset is 0.48 and the entropy is 0.971.

Q10 ANS Advantages of Random Forests over Decision Trees:

Random Forests are less prone to over fitting compared to Decision Trees, as they are built by combining multiple decision trees through bagging or boosting. This results in a more robust and stable model that is less sensitive to noise and outliers in the data.Random Forests can handle both categorical and continuous variables, while Decision Trees are primarily designed for categorical variables.Random Forests provide a feature importance measure that can help in feature selection and understanding the most important features in the data. Random Forests are computationally efficient and can handle large datasets with high dimensionality.Random Forests can be easily parallelized, which makes them suitable for distributed computing and big data applications.

Q11 ANS Scaling all numerical features in a dataset is important to ensure that all features contribute equally to the model training process. When features have different scales, the model may give more weight to features with larger scales, which can result in biased predictions. Scaling also helps to improve the convergence rate of many optimization algorithms, such as gradient descent.
Two common techniques used for scaling numerical features in a dataset are:
1: Standardization: This involves transforming the features so that they have zero mean and unit variance. This can be achieved by subtracting the mean and dividing by the standard deviation of each feature. Standardization is useful when the distribution of the feature values is normal or approximately normal.
2: Min-Max Scaling: This involves transforming the features so that they are in the range [0,1]. This can be achieved by subtracting the minimum value of each feature and dividing by the range (i.e., the difference between the maximum and minimum values). Min-Max scaling is useful when the distribution of the feature values is not normal or when the algorithm used for modeling requires the features to be in a certain range.

Q12 ANS Scaling provides several advantages in optimization using gradient descent algorithm, including: 1: Faster convergence: Scaling can help the gradient descent algorithm converge faster to the minimum of the cost function. This is because the step size taken by the algorithm is proportional to the size of the gradient, and when features are not scaled, some gradients can be much larger than others, causing the algorithm to take much larger steps in some directions than others. This can result in the algorithm taking a longer time to Converge to the minimum of the cost function.
2: Improved numerical stability: Scaling can also help to improve numerical stability in gradient descent. When features have large differences in scale, the Hessian matrix of the

cost function can become ill-conditioned, which can cause numerical instability and make the algorithm sensitive to small perturbations. Scaling the features can help to improve the condition number of the Hessian matrix, making the algorithm more stable.
3: Better generalization: Scaling can also improve the generalization performance of the model. When features are not scaled, the model may give more weight to features with larger scales, which can result in over fitting. Scaling the features can help to prevent this by ensuring that all features contribute equally to the model training process.

Q13 ANS: In case of a highly imbalanced dataset for a classification problem, accuracy may not be a good metric to measure the performance of the model. This is because accuracy can be misleading in such cases, as it measures the proportion of correctly classified instances out of the total number of instances in the dataset.
For example, if we have a dataset with 95% of instances belonging to class A and only 5% belonging to class B, a classifier that always predicts class A will achieve an accuracy of 95%. However, such a classifier would be of little use in practice, as it would fail to correctly identify instances of class B.In such cases, it is often better to use other evaluation metrics such as precision, recall, F1-score, and ROC-AUC curve. These metrics take into account the true positives, false positives, true negatives, and false negatives of the classification results and provide a more informative measure of the performance of the classifier, especially for the minority class.
Therefore, accuracy may not be a good metric to measure the performance of the model in case of highly imbalanced datasets, and other evaluation metrics should be used instead.

Q14 ANS: The F-score, also known as the F1-score, is a performance metric used in binary classification problems that balances the trade-off between precision and recall. It is the harmonic mean of precision and recall and is defined as follows:
F1-score = 2 * (precision * recall) / (precision + recall)
where precision is the ratio of true positives to the total number of positive predictions, and recall is the ratio of true positives to the total number of actual positive instances.The F1-score ranges from 0 to 1, where a score of 1 indicates perfect precision and recall, and a score of 0 indicates poor performance.

Q15 ANS: In machine learning, fit(), transform() and fit_transform() are common methods used in data pre-processing and modeling pipelines. The main differences between these methods are as follows:

- fit(): This method is used to fit the parameters of the model or the data transformer to the training data. In other words, it learns the patterns and structure of the training data to be used for later prediction or transformation. Once the model or transformer is fit to the training data, it can be used to predict or transform new data.
- transform(): This method is used to apply the learned transformation to new data, i.e., to make predictions or transform the input features based on the learned patterns from the training data. This method is typically used after the fit() method to apply the learned patterns to new data.
- fit_transform(): This method combines the fit() and transform() methods into a single step, i.e., it fits the parameters of the transformer to the training data and applies the transformation to the same data in one step. This method is used when we want to perform both fitting and transformation on the same dataset in a single step.