Worksheet 7
Answers

PART 1 – Machine Learning

1. D)
2. A)
3. B)
4. A)
5. C)
6. A)
7. B)
8. B)
9. To calculate the Gini index and entropy of the dataset with two classes A and B, we need to know the probabilities of each class. Given that the percentage of class A is 40% and the percentage of class B is 60%, we can calculate the probabilities as:
   P(A) = 0.4
   P(B) = 0.6
   Gini index:
   The Gini index is a measure of impurity or inequality in a dataset. It ranges from 0 to 1, where 0 represents a completely pure dataset (i.e., all samples belong to the same class) and 1 represents a completely impure dataset (i.e., the classes are evenly distributed).
   The formula to calculate the Gini index is:
   Gini = 1 - (P(A)^2 + P(B)^2)
   Plugging in the values of P(A) and P(B), we get:
   Gini = 1 - (0.4^2 + 0.6^2) = 0.48
   Therefore, the Gini index of the dataset is 0.48.
   Entropy:
   Entropy is another measure of impurity or randomness in a dataset. It also ranges from 0 to 1, where 0 represents a completely pure dataset and 1 represents a completely random dataset.
   The formula to calculate entropy is:
   Entropy = - P(A) * log2(P(A)) - P(B) * log2(P(B))
   Plugging in the values of P(A) and P(B), we get:
   Entropy = - 0.4 * log2(0.4) - 0.6 * log2(0.6) = 0.971
   Therefore, the entropy of the dataset is 0.971.
10. Random Forests and Decision Trees are both popular machine learning algorithms used for classification and regression tasks. While Decision

Trees are relatively simple and straightforward to interpret, Random Forests offer several advantages over them, including:

Reduced overfitting: Decision Trees are prone to overfitting, which means that they can memorize the training data and perform poorly on new, unseen data. Random Forests reduce overfitting by combining multiple Decision Trees and using them to make predictions, which helps to reduce the variance of the model.

Improved accuracy: Random Forests can achieve higher accuracy than Decision Trees by combining the predictions of multiple trees. This ensemble approach helps to reduce bias and variance, leading to more accurate and reliable predictions.

Robustness to outliers: Decision Trees can be sensitive to outliers and noise in the data, which can lead to poor performance. Random Forests are more robust to outliers because they use multiple trees, which are less likely to be affected by individual data points.

Feature importance: Random Forests can provide information about the relative importance of different features in the data. This can be useful for feature selection and understanding which features are most relevant for making accurate predictions.

Scalability: Random Forests can be trained on large datasets and can handle many features. They can also be parallelized, which allows for faster training on multicore processors or distributed computing systems.

Overall, Random Forests offer several advantages over Decision Trees, including improved accuracy, reduced overfitting, robustness to outliers, feature importance, and scalability. However, they may be slower to train and require more computational resources than Decision Trees.

11. Scaling numerical features in a dataset is important because it helps to ensure that the features are on a similar scale and have similar ranges, which can improve the performance and stability of many machine learning algorithms. Here are a few reasons why scaling is necessary:

    Some machine learning algorithms are sensitive to the scale of the input features. For example, distance-based algorithms like k-nearest neighbors (KNN) and clustering algorithms like k-means can be affected by differences in feature scales.

    Scaling can help to normalize the data and remove any biases due to differences in measurement units or scales. This can improve the interpretability of the model and make it easier to compare the relative importance of different features.

    Two common techniques used for scaling numerical features are:

Min-max scaling (also known as normalization): This method scales the values of the features to a range between 0 and 1. It is calculated as:

X_scaled = (X - X_min) / (X_max - X_min)

where X is the original feature value, X_min is the minimum value of X, and X_max is the maximum value of X.

Standardization: This method scales the values of the features to have zero mean and unit variance. It is calculated as:

X_scaled = (X - mean(X)) / std(X)

where X is the original feature value, mean(X) is the mean of X, and std(X) is the standard deviation of X.

These scaling techniques can be easily applied using libraries such as scikit-learn in Python.

12. Scaling the input features can provide several advantages when using optimization algorithms like gradient descent:

    1. Faster convergence: Scaling can help the optimization algorithm to converge faster because it reduces the oscillations in the cost function surface. When the features have similar scales, the optimization algorithm can take more consistent steps towards the minimum.

    2. Avoidance of local minima: Scaling can help to avoid getting stuck in local minima by making the cost function more symmetric. This is because the gradients of the cost function with respect to each feature are more comparable in magnitude when the features are on the same scale.

    3. More accurate gradients: Scaling can help to prevent numerical instability and overflow/underflow errors when computing the gradients. This is because the gradients are less sensitive to small changes in the input features when they are on the same scale.

    4. Better conditioning: Scaling can improve the condition number of the optimization problem, which can make it easier to find the global minimum. A high condition number means that the cost function surface is steep in some directions and shallow in others, which can make it difficult for the optimization algorithm to find the minimum.

Overall, scaling the input features can make optimization using gradient descent more efficient, stable, and accurate. This is especially important when working with large datasets and complex models, where optimization can be a bottleneck.

13. In case of a highly imbalanced dataset for a classification problem, accuracy is not a good metric to measure the performance of the model. This is because accuracy is a measure of the total number of correct predictions divided by the total number of predictions, and it does not take into account the class distribution of the dataset.

For example, consider a dataset with 95% of the samples belonging to class A and only 5% belonging to class B. If we build a model that predicts all samples as class A, we will get an accuracy of 95%, which looks impressive. However, this model is not useful because it does not predict any samples as class B, which is the class we are interested in. In this case, accuracy is a misleading metric and does not reflect the true performance of the model.

To handle imbalanced datasets, we need to use evaluation metrics that are sensitive to the class distribution, such as precision, recall, F1-score, or Area Under the ROC curve (AUC-ROC). These metrics take into account the number of true positives, false positives, true negatives, and false negatives, and can provide a more accurate assessment of the performance of the model on each class.

In summary, accuracy is not a good metric to measure the performance of the model in case of a highly imbalanced dataset. Instead, we should use evaluation metrics that are sensitive to the class distribution and provide a more accurate assessment of the model's performance.

14. The F-score (also known as F1-score) is a common evaluation metric used in classification problems that calculates the harmonic mean of precision and recall. It is a balanced measure that takes into account both precision and recall and provides a single score that reflects the overall performance of the model.

The mathematical formula for F-score is as follows:
F1-score = 2 * (precision * recall) / (precision + recall)
where:
Precision: The ratio of true positives (TP) to the total number of positive predictions (TP + FP). It measures the accuracy of the positive predictions.
Recall: The ratio of true positives (TP) to the total number of actual positives (TP + FN). It measures the completeness or sensitivity of the positive predictions.

The F1-score ranges between 0 and 1, with higher values indicating better performance. A score of 1 indicates perfect precision and recall, while a score of 0 indicates poor performance.

The F1-score can be a useful metric when we have an imbalanced dataset or when we want to find a balance between precision and recall. However, it can be biased towards either precision or recall depending on the relative importance of each metric for the problem at hand. In such cases, we can use other variations of the F-score, such as F-beta score, which provides a trade-off between precision and recall by using a weighted average of the two metrics with a parameter beta.

15. In machine learning, fit(), transform(), and fit_transform() are methods used in data preprocessing and model training.

    fit(): This method is used to estimate the parameters of a preprocessing step or a model. It learns the properties of the data such as mean, standard deviation, or weights, and stores them as internal attributes. The fit() method does not modify the input data and returns the estimator object itself.

    transform(): This method applies the learned transformation to the input data. It can be used to preprocess new data that has the same properties as the training data. The transform() method can modify the input data and returns the transformed data.

    fit_transform(): This method combines the fit() and transform() methods into a single step. It learns the parameters of the preprocessing step from the training data and applies it to the same data in a single operation. The fit_transform() method modifies the input data and returns the transformed data.

    In summary, fit() is used to estimate the parameters of a model or preprocessing step, transform() is used to apply the learned transformation to new data, and fit_transform() is used to learn the parameters and apply the transformation to the same data in a single step.

    It is important to note that not all estimators have a fit_transform() method, and some estimators may have additional methods or parameters. It is recommended to check the documentation of each estimator to understand its usage and behavior.

PART 2 – Statistics Answers

1. b)
2. d)
3. c)
4. b)
5. b)
6. a)
7. c)
8. b)
9. b)
10. d)
11. a)
12. a)
13. d)
14. a)
15. d)