Decision Tree Classifier

*Bulding a decision tree classifier for determining the type of wines.

Kare Swetha
Computer Science Department(George Mason University)
Fairfax, Virginia
skare@gmu.edu

***Abstract*—This project is about building a decision tree classifier to predict the types of wines using various features of the wine such as alcohol content, malic acid, ash content, etc. The dataset used for this task is the Wine dataset which contains 30 samples of wine, each with 13 features. The decision tree classifier is built using the scikit-learn library in Python. The hyperparameters of the decision tree, such as the criterion for splitting, maximum depth of the tree, maximum number of features to consider, and the splitting strategy are tuned using a grid search with cross-validation. The best hyperparameters are then used to train the decision tree classifier, which is then used to predict the types of wines for a test dataset. Finally, the accuracy of the predictions is evaluated, and the trained decision tree is visualized using the export text function provided by the scikit-learn library. Our main aim is to be able to identify the type of wine based on the decision tree that has been trained using the samples.**

***Index Terms*—decision tree, cross-validation,hyperparameters.**

I. Introduction

Classification is a supervised machine learning process of recognition,understanding, and grouping of objects and ideas into preset categories. It utilizes input training data for the purpose of predicting the likelihood that the data that follows will fall into one of the predetermined categories.

Basically classification consists of 2 steps one is the learning step and the other is the prediction step, In the learning step, the model is developed based on given training data. In the prediction step, the model is used to predict the response to given data. A decision tree is one of the easiest and most popular classification algorithms used to understand and interpret data. It can be utilized for both classification and regression problems.

The topmost node in a decision tree is known as the root node. It learns how to partition based on attribute value. It partitions the tree in a recursive manner called recursive partitioning. We have used the following approach to build a decision tree 1. Select the best attribute (Attribute Selection Measures-ASM) to split the records. Make that attribute a decision node. Breaks the dataset into smaller subsets.Construct the tree by recursively repeat the process for each child until, either: All the tuples belong to the same attribute value or if There are no more remaining attributes or if There are no more instances.

II. Background

The background of building a decision tree classifier can be traced back to the field of machine learning and data mining. Decision trees are a popular tool used for classification and prediction tasks, where the goal is to build a model that can accurately classify or predict new instances based on past observations. Decision trees are a type of supervised learning algorithm that can be used for both classification and regression tasks.

Decision trees are attractive because they are easy to interpret and can handle both categorical and numerical data. The algorithm works by recursively partitioning the data based on the values of the input features, so that each partition contains instances that are as homogeneous as possible with respect to the target variable. The result is a tree-like structure that can be used to make predictions on new instances by traversing the tree based on their feature values.

Decision trees have been successfully applied in many domains, such as finance, healthcare, and marketing. In the context of the wine dataset, decision trees can be used to classify the types of wines based on their various features. By building an accurate decision tree classifier, we can predict the types of wines based on their characteristics.

III. Proposed Approach

The proposed approach for this project to classify the type of wine based on the given features and data involves using a decision tree classifier. In this approach we load the dataset then we perform a split.

The split is 70% for training and 30% for testing. By splitting our dataset into training and testing data, we can reserve some data to verify our models effectiveness. We do this split before we build our model in order to test the effectiveness against data that our model hasn't yet seen. We can also split our data into training and testing data to prevent overfitting. This can be done using the traintestsplit() function in sklearn.

We created our Decision Tree Classifier model and assigned it to the variable clf We then applied the .fit() method to train the model. In order to do this, we passed in our training data.

We also perform Hyper-parameter tuning, which refers to the process of tuning these values to ensure a higher accuracy score. One way to do this is, simply, to plug in different values and see which hyper-parameters return the highest

score. We have plugged in different combination values into the GridSearchCV method of Scikit-learn where we use a dictionary of different values to try and different combination of parameters are passed to the decision classifier during each run, this helps to increase the accuracy. We assigned a new variable, predictions, which takes the values from applying the .predict() method to our model clf. We make predictions based on our X_test data.

IV. Experiment Results

A gradual improvement has been performed to arrive at the current accuracy.

1. Initially the decision tree classifier method without any parameters has been used DecisionTreeClassifier() and the accuracy was predicted
Predicted classes for the Test Data: [1 1 2 2 1 3 3 3 2]
Accuracy of the Predictions on the Test Data: 0.6666666666666666

2. To improve the accuracy certain parameters were passed to the decision tree DecisionTreeClassifier(criterion='entropy',RandomState=42,splitter='best')
Predicted classes for the Test Data: [1 1 2 2 1 3 3 3 2]
Accuracy of the Predictions on the Test Data: 0.6666666666666666
there was no improvement in the accuracy even after changing the hyperparameters.

3. Different parameter combinations were tried to ensure the improvement in accuracy.
DecisionTreeClassifier(criterion='entropy',randomstate=42
,splitter='random')
Predicted classes for the Test Data: [2 3 2 2 2 3 3 2 2]
Accuracy of the Predictions on the Test Data: 0.7777777777777778
On using these params there is a slight improvement in the accuracy compared to the previous state. this improvement occured when the splitter was random rather than best.

4. On changing another parameter to the previous param combinations randomstate = 1 the accuracy was improved to 100%.
DecisionTreeClassifier(criterion='entropy',random_state=1
,splitter='random')
Rather than trying out a whole slew of different combinations, Scikit-learn provides a way to automate these tests. This method is the GridSearchCV method, which makes the process significantly faster. We simply need to provide a dictionary of different values to try and Scikit-Learn will handle the process for us.
The dictionary used in the grid search is as follows:
parameters {'criterion': ['entropy'], 'max_depth': [3,4], 'max_features': [None], 'splitter': ['random'], 'random_state':[1]}
Predicted classes for the Test Data: [1 3 2 2 3 3 3 2 2]
Accuracy of the Predictions on the Test Data: 1.0
Furthermore, all the other experiments were performed with these Decision tree params; parameters {'criterion': ['entropy'], 'max_depth': [3,4], 'max_features': [None], 'splitter': ['random'], 'random_state':[1]} 5. Modifications to the test_split method was also done to check the accuracy changes, the random_state was altered, when random_state was 0 in this method train_test_split(X, y, test_size=0.3, random_state=0) and the decesionTreeclassifier having the previously defined parameters a good amount of accuracy was achieved. Predicted classes for the Test Data: [1 3 2 2 3 3 3 2 2]
Accuracy of the Predictions on the Test Data: 1.0

6. When random_state was set to 1 the accuracy achieved was as follows
Predicted classes for the Test Data: [2 3 2 2 3 3 3 1 3]
Accuracy of the Predictions on the Test Data: 0.8888888888888888

7. When random_state was set to 2 the accuracy achieved was as follows
Predicted classes for the Test Data: [1 1 3 1 3 1 2 1 1]
Accuracy of the Predictions on the Test Data: 0.6666666666666666

8. When random_state was set to 6 the accuracy achieved was as follows
Predicted classes for the Test Data: [1 3 3 1 2 2 1 1 3]
Accuracy of the Predictions on the Test Data: 1.0

9. When random_state was set to 7 the accuracy achieved was as follows
Predicted classes for the Test Data: [1 2 1 1 2 1 2 3 3]
Accuracy of the Predictions on the Test Data: 1.0

10. On increasing the test size to 80% the following changes were achieved.
train_test_split(X, y, test_size=0.4, random_state=0)
Predicted classes for the Test Data: [1 3 2 2 3 3 3 3 2 3 1 2]
Accuracy of the Predictions on the Test Data: 0.9166666666666666

11. On reducing the test size train_test_split(X, y, test_size=0.1, random_state=0)
Predicted classes for the Test Data: [1 3 3]
Accuracy of the Predictions on the Test Data: 0.6666666666666666

Finally, the accuracy achieved when the train_test_split(X, y, test_size=0.1, random_state=0) and with these decision tree params; parameters {'criterion': ['entropy'], 'max_depth': [3,4], 'max_features': [None], 'splitter': ['random'], 'random_state':[1]} was 100% these were fixed for the given wine dataset.

V. Conclusion

In this project, a decision tree classifier has been built to predict the type of wine based on several features.The dataset was split into training and testing features. Then the grid search was performed to find the best hyperparameters for the decision tree classifier. Finally, a decision tree classifier using the best hyperparameters was developed and evaluated its performance on the test data. The results showed that the decision tree classifier was able to accurately predict the type of wine with an accuracy of 100%. Many experiments by hyper parameters tuning was performed to arrive to this best result. In conclusion, the decision tree classifier was able to successfully predict the type of wine with high accuracy and provided useful insights into the data.

References

[1] Russell, S. and Norvig, P. (2022) AI: A MODERN APPROACH, 4th Ed. (US edition). UC Berkeley – freely available online.

[2] www.datacamp.com/blog/classification-machine-learning

[3] datagy.io/sklearn-decision-tree-classifier

[4] Poole, D.L., and Mackworth, A.K. (2017) Artificial Intelligence: Foundations of Computational Agents, 2nd Edition. Cambridge University Press