Machine Learning

ASSIGNMENT 1: Decision Tree

Group 48:

19CS30032 - P Anurag Reddy

19CS10061 - Shashank Suroju

Data:

The given data is about predicting heart disease . The data used for training was provided in .csv format. We have randomly split the data as 80/20 as 80 for training, 20 for testing.

The data has the following attributes:

- Age
- Sex
- Chest pain Type
- Resting blood pressure
- Serum cholesterol
- Fasting blood sugar
- Resting electrocardiographic results
- Maximum heart rate achieved
- Exercise induced angina
- Oldpeak
- Slope of the peak exercise ST segment
- No of major vessels colored by fluoroscopy
- thal

The above attributes have nominal, real, binary and ordered feature values.

Real: 1,4,5,8,10,12 ;Ordered:11; Binary: 2,6,9; Nominal:7,3,13

Functions:

build_tree:

We recursively call build_tree to make new nodes and join them with their children. We call best_split in the build_tree function to find the best split at that node. We make child nodes according to this best split. If further splitting is not possible then, we calculate the classification of the node and make it a leaf node.

print_tree:

Uses Pre-Order traversal to print the decision tree(Best classifier obtained in question 3)

make_prediction:

Recursively goes down the tree from root to a leaf node to find the best classification for the data.

predict:

Calls make_prediction for every value of X which is the validation set

fit:

Calls build tree and stores the root node to root of tree

calculate_leaf_value:

Computes the Classification of leaf node based on target values of the node

get_root:

Returns the root of the tree

get_depth:

Return the depth of the tree

get_num_nodes:

Returns the number of the nodes in the tree

entropy:

Returns the entropy of the node

gini_index:

Returns the gini_index of the node

Information_gain:

Given parent node, children nodes and the mode of impurity measurement calculates the information gain

get_best_split:

Given the dataset, number of samples and number of features at a node splits the dataset into left and right dataset based on an attribute and threshold computed. We have split the node attributes in two different ways based on the fact that they have nominal values or real values.

splits:

Splits the data into left and right subsets based on threshold and feature index

point:

Traverses the complete tree and points the children to their respective parents

prune:

In the prune() function, we are first traversing to the leaf nodes. After that, we are traversing back to the top. While traversing back to the top, we are checking if the accuracy of the prediction is increasing by changing the decision node to a leaf node. If the accuracy increases then we keep it as a leaf node. But if the accuracy does not increase then we change it back to a decision node.

Note:

The classifier giving best accuracy in Q2 has been named as “classifier” in the code.

The classifier giving the best accuracy with optimal height has been named “best_classifier” in the code.

For our instance of random values, they both have the same accuracy. So, please don't get confused among them.

We performed pruning on “classifier” and not on “ best_classifier”.

Results:

Note: We are taking random values using a random seed. So, the values that you will be getting may not be the same as the values we are getting. So, we are attaching appropriate screenshots wherever necessary.

```
gini_Y_pred = gini_classifier.predict(X_test)
print(accuracy_score(Y_test, gini_Y_pred))

enpy_Y_pred = enpy_classifier.predict(X_test)
print(accuracy_score(Y_test, enpy_Y_pred))
```

```
0.7222222222222222
0.6666666666666666
```

The accuracies obtained by using gini index and entropy as impurity measures are above. We can observe that the accuracy of classifier using the gini index is better than the accuracy of the classifier using entropy.

So, all the subsequent classifiers to be built in the assignment from this point were built using gini index as an impurity measure.

```
print("Average Accuracy: ",sum(gini_accu)/len(gini_accu))
```

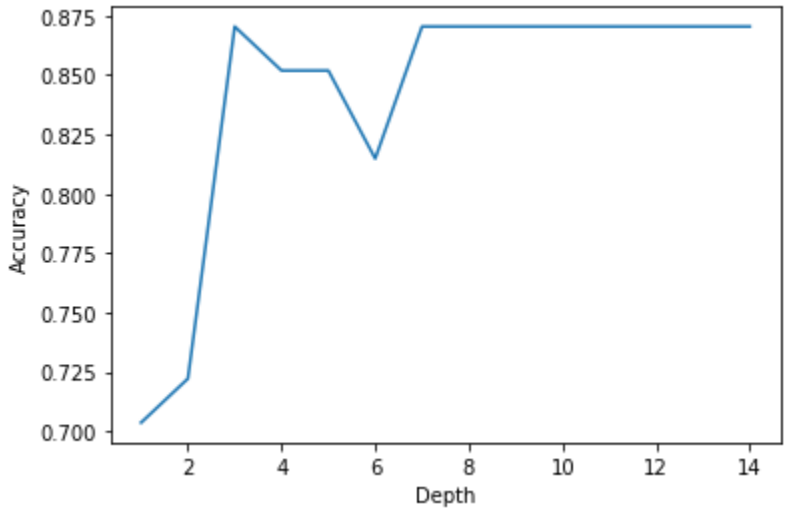
```
Average Accuracy:  0.7518518518518519
```

Accuracy on averaging over 10 random 80/20 splits is **75.185%**

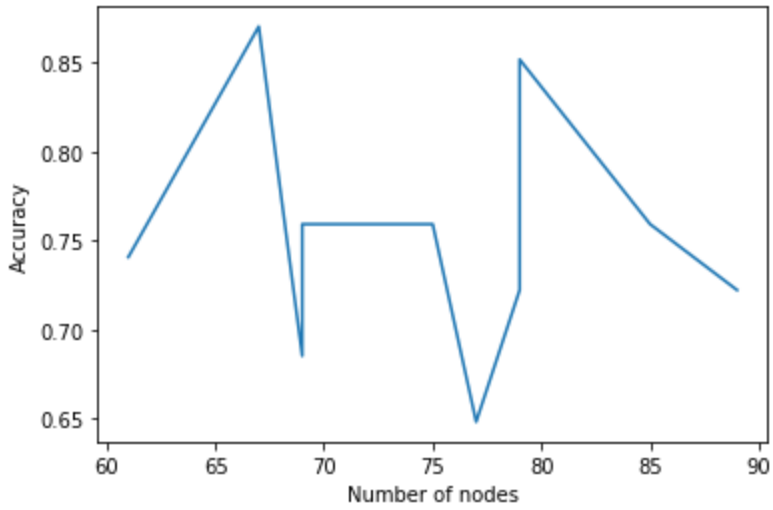
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=.2, random_state=randomlist[maxpos])
best_classifier = DecisionTreeClassifier(max_depth=depth, mode = "gini")
best_classifier.fit(X_train,Y_train)
Y_pred = best_classifier.predict(X_test)
acc = accuracy_score(Y_test, Y_pred)
print(acc)
```

```
0.8703703703703703
```

Tree with best test accuracy: **87.037%**



The accuracy increases with the maximum depth of the classifier. But we can observe that, for depth 3, the classifier performs as well as it does when its depth is more than 10. Therefore the optimal depth is 3 in the given instance.



We can observe no proper relation between the number of nodes and accuracy. But as the number of nodes is increasing the accuracy is decreasing.

Pruning Results:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=.2, random_state=randomlist[maxpos])
best_classifier = DecisionTreeClassifier(max_depth=depth, mode = "gini")
best_classifier.fit(X_train,Y_train)
Y_pred = best_classifier.predict(X_test)
acc = accuracy_score(Y_test, Y_pred)
print(acc)
```

```
0.8703703703703703
```

Before Pruning the accuracy of the Best classifier is **87.037%**.

```
Y_pred = classifier.predict(X_test,root)
print(accuracy_score(Y_test, Y_pred))
```

```
0.9074074074074074
```

` After Pruning the accuracy increased to **90.740%**