Implementation of Decision Tree Classifier Algorithm

Project Link:
https://colab.research.google.com/drive/1wKaMkfcd86XmkP6qjiit4BTV7Hi8crL8?usp=sharing

1. Introduction

Decision tree is a type of supervised algorithm that starts with a root node and traverse the tree based on a greedy approach in which all the possible traversing nodes are checked and the best one is selected. The best attribute is usually used as the root node and then the data is split into two parts, one for the training and testing the accuracy of the algorithm. These steps are repeated until the entire tree has been traversed. The core principle of decision trees is to identify those features which have the most "information" on the objective feature and then divide the dataset over the values of these features so that their target value is as pure as possible for the resulting sub datasets.

Each year hundreds of animal species is discovered and scientists look for various features in these animals to distinguish them from others in order to classify them in different animal categories. There are seven categories that these newly discovered animal species can fit in and to do that scientists look for factors such as how many legs does it has or does it have hairs, etc. This report introduces a decision tree algorithm that can classify these new discovered species into their respective category.

2. Decision Tree Algorithm

The decision tree algorithm consists of total four modules, that are calculation of entropy, calculation of information gain, the decision tree algorithm function and finally functions for prediction and testing. Details on important modules is as follow.

Entropy

Entropy is the measure of how pure the dataset is, and it is used to determine the information gain, which in turn is used to decide on the best possible split while traversing the tree. The formula for entropy is as follow.

Entropy = $-\sum(P(x)*\log_2(P(x))$, where P(x) is the number of occurrences divided by the total number of samples.

This is converted into python code as shown below in the lines 47 to 53.

```
47. #Function to Calculate the Entropy
48.
49. def calculate_entropy(target_column):
50.     histogram = np.bincount(target_column)
51.     ps = histogram / len(target_column)
52.     result= -np.sum([p * np.log2(p) for p in ps if p > 0])
53.     return result
```

Information Gain

Information gain is the measure of how much information is gained when the node is split into child nodes. The idea is to split the nodes on the basis of most information gain. The formula for information gain is as follow.

Information Gain = Entropy(Parent) – [Weighted Average]*Entropy(Children)

This is translated into python code as follow in the lines 59 to 73.

```
59. def Information_Gain(data,split_attribute_name,target_name="Animal_Type"):
60.
61.     #Calculation of the Total Entropy
62.     total_entropy = calculate_entropy(data[target_name])
63.
64.     #Calculation of Split Counts and Values for the Attribure
65.     values,counts= np.unique(data[split_attribute_name],return_counts=True)
66.
67.     #Calculation of Weighted Entropy
68.     weighted_entropyropy = np.sum([(counts[i]/np.sum(counts))*calculate_entropy(data.where(data[split_attribute_name]==values[i]).dropna()
[target_name]) for i in range(len(values))])
69.
70.     #Calculation of Information Gain
71.     Information_Gain = total_entropy - weighted_entropyropy
72.
73.     return Information_Gain
```

Prediction & Testing

The prediction module uses the dictionary values as the queries in conjunction with the tree, on which the prediction task is performed. This is a recursive task in which the entire tree is traversed until the leaf node is reached.

The prediction function works by checking if the query values contain any feature values and check for its root node against query features. If a match is found, the tree is traversed along the root node and the last value, that is, the value of the leaf node is the prediction in this case.

Otherwise, if any other node is matched with the query value, then the features from the query is matched with that node value, and then the tree is traversed in such a manner that that query[key] == query feature value. This is continued in a recursive manner.

This recursive process is shown in the code below.

```
122. def predict_data(query,tree,default = 1):
123. 
124.     #Match each feature in query
125.     for key in list(query.keys()):
126.         if key in list(tree.keys()):
127. 
128.             try:
129.                 result = tree[key][query[key]]
130.             except:
131.                 return default
132. 
133.             result = tree[key][query[key]]
134. 
135.             if isinstance(result,dict):
136.                 return predict_data(query,result)
137. 
138.             else:
139.                 return result
```

In the testing module, we simple remove the targeted feature values from the data and convert them into a dictionary format. This is then used as the query values for the prediction function to work on. The results are stored in a variable called predicted. Finally, the accuracy of the test is calculated with the formula. This is shown in code below.

```
143. def test_data(data,tree):
144. 
145.     #Converting the targeted column in dictionary format
146.     queries = data.iloc[:,:-1].to_dict(orient = "records")
147. 
148.     # To store the predicted data
149.     predicted = pd.DataFrame(columns=["predicted"])
150. 
151.     #To calculate the accuracy of the prediction
152.     for i in range(len(data)):
153.         predicted.loc[i,"predicted"] = predict_data(queries[i],tree,1.0)
154.     print('The Accuracy of Algorithm is : ',(np.sum(predicted["predicted"] == data["Animal_Type"])/len(data))*100,'%')
```

3. Model Evaluation

The model is evaluated on the data taken from the UCI Machine Learning Repository. The dataset is called Zoo Data Set. The dataset is multivariate with categorical and integer attributes. There are total 101 rows and 18 columns. The attributes include animal features such as if it has hairs, is venomous, etc. The last column is the animal type attribute, which is an integer type with values ranging from 1 to 7. Each number represents a set of animal class. Every other attribute has a Boolean type value of either 0 or 1, representing yes and no. For example, the attribute hair with value 0 means the animal does not have hairs, else if the value is 1 then the animal does have hairs.

Data Preparation

Fortunately, the data has no missing values and was loaded with the code shown below.

```
#Loading The Dataset

df = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/zoo/zoo.data',
                 names=['animal_name','hair','feathers','eggs','milk','airbone','aquatic','predator','toothed',
                        'backbone','breathes','venomous','fins','legs','tail','domestic','catsize','Animal_Type',])

```

After loading the data, the first attribute, named 'animal_name' was dropped using the drop() function. This was done because it served no purpose in the classification model and would only decrease the prediction accuracy.

Experiment Design and Evaluation

The data is split into training data and testing data. The training data is fed to the decision tree algorithm for the algorithm to run it. It also serves as the original data for the algorithm with the feature being selected as the last column attribute, i.e., 'animal_type'.

The testing data is fed to the test module along with the entire tree that the decision tree algorithm made.

Evaluation Results

In the first run the data was split using the iloc() pandas module as 80% rows for training and 20% rows for testing as shown in the code below.

```
[24] #Function to split the data into training and testing sets

     def train_test_split(df):
         training_data = df.iloc[:80].reset_index(drop=True)
         testing_data = df.iloc[80:].reset_index(drop=True)
         return training_data,testing_data

     training_data = train_test_split(df)[0]
     testing_data = train_test_split(df)[1]
```

This splitting achieved an accuracy score of about 85% as shown below.

```
[30] tree = decision_tree_algorithm(training_data,training_data,training_data.columns[:-1])
     test_data(testing_data,tree)

     The Accuracy of Algorithm is :  85.71428571428571 %
```

The second run had data split as 90% rows as training and 30% rows as testing. The result was an accuracy score of about 97%, a 12% increase from the first run.

```
[51] tree = decision_tree_algorithm(training_data,training_data,training_data.columns[:-1])
     test_data(testing_data,tree)

     The Accuracy of Algorithm is :  97.1830985915493 %
```

4. Conclusion

The decision tree algorithm was implemented on the zoo dataset and using the best features of the data available the tree was constructed. The algorithm works by choosing the best possible parent child node relationship by calculating the information gain when traversing the tree. This helps the algorithm to choose the best child node and when the leaf node is encountered, that is, there is no possible child nodes, the algorithm stops. The data is then split into two parts, one for the training purposes and the other for testing.

The training data is ran through the decision tree algorithm and the output is a tree which is then ran though the test module along with the training data. The final accuracy of the algorithm was 97%.

References

Beach, J. 2020, “PlanetB | Syntax Highlight Code in Word Documents,” *Planetb.ca*, viewed 11 October 2020, <http://www.planetb.ca/syntax-highlight-word>.

BlueSurfer 2013, “Fastest way to compute entropy in Python”, *Stack Overflow*, viewed 11 October 2020, <https://stackoverflow.com/questions/15450192/fastest-way-to-compute-entropy-in-python>.

Pangas 2020, “pandas documentation — pandas 1.1.3 documentation”, *Pydata.org*, viewed 11 October 2020, <https://pandas.pydata.org/pandas-docs/stable/index.html>.

PythonCourse 2016, “Machine Learning with Python: Decision Trees in Python”, *Python-course.eu*, viewed 11 October 2020, <https://www.python-course.eu/Decision_Trees.php>.

Rpalsaxena 2017, “How Decision Tree Algorithm works,” *Dataaspirant*, viewed 11 October 2020, <https://dataaspirant.com/how-decision-tree-algorithm-works/>.

UCI 2015, “UCI Machine Learning Repository: Zoo Data Set”, *Uci.edu*, viewed 11 October 2020, <http://archive.ics.uci.edu/ml/datasets/zoo>.