Machine Learning Assignment

Decision Tree Classifier

Abhigyan Mishra

11 th June, 2022

About the Dataset

The dataset used is ‘glass’, which aims to provide information related to how the chemical elements used to manufacture glass determine its type, this provides a great dataset for data exploration & visualization, as an alternative to the commonly used iris dataset.

The dataset contains data for 215 samples belonging to one of 7 different kinds of glass. The data has been collected from a number of glass manufacturing companies.

The dataset has been originally created for R and distributed as an R package, however, exported versions available for use as CSV files have been made available by users.

The dataset includes the following set of features per instance:

- RI: refractive index
- Na: Sodium (unit measurement: weight percent in corresponding oxide, as are attributes 4-10)
- Mg: Magnesium
- Al: Aluminum
- Si: Silicon
- K: Potassium
- Ca: Calcium
- Ba: Barium
- Fe: Iron
- Type of glass: (class attribute)

1. buildingwindowsfloatprocessed
2. buildingwindowsnonfloatprocessed
3. vehiclewindowsfloatprocessed
4. vehiclewindowsnonfloatprocessed (none in this database)
5. containers
6. tableware
7. headlamps

Citation: https://github.com/jbrownlee/Datasets/blob/master/glass.csv

Building a Decision Tree (using Information Gain)

For classification, a decision tree was learnt using the given data partitioned into training and test subsets. The decision tree is learnt using information gain as the measure of goodness of the split, where information gain is defined as:

$$\Delta_{info} = \Phi_{parent}(\chi|m) - \sum_{j=1}^{J} \frac{N_{mj}}{N_m} \phi(\chi_{mj}|m_j)$$

Along with it , other measure of measure have also been utilized :

- Ginni Gain:

$$\Delta = Ginni_{parent}(\chi|m) - \sum_{j=1}^{J} \frac{N_{mj}}{N_m} Ginni(\chi_{mj}|m_j)$$

- Gain Ratio:

$$GR(\chi) = \frac{\delta_{info}}{SplitInf(m)}, whereSplitinf(m) = -\sum_{i=1}^{j} \frac{N_{mj}}{N_m} * log_2(\frac{N_{mj}}{N_m})$$

- Misclassification Error:

$$1 - max_{i \epsilon Z_k}(p(i|m)))$$

Importing & Preparing the Dataset for Training:

The simple variant of the dataset is loaded from the CSV file, and any rows containing N/A values were dropped.

```
glass = pandas.read_csv("glass.csv")

glass = glass.dropna()

instances = glass.drop("Type", axis=1)

responses = glass["species"]
```

The dataset is then divided into training and testing subsets using the sklearn model_selection train_test_split function:

```
X_train, X_test, R_train, R_test = model_selection.train_test_split(
 instances, responses, random_state = 0, train_size = 0.7
)
```

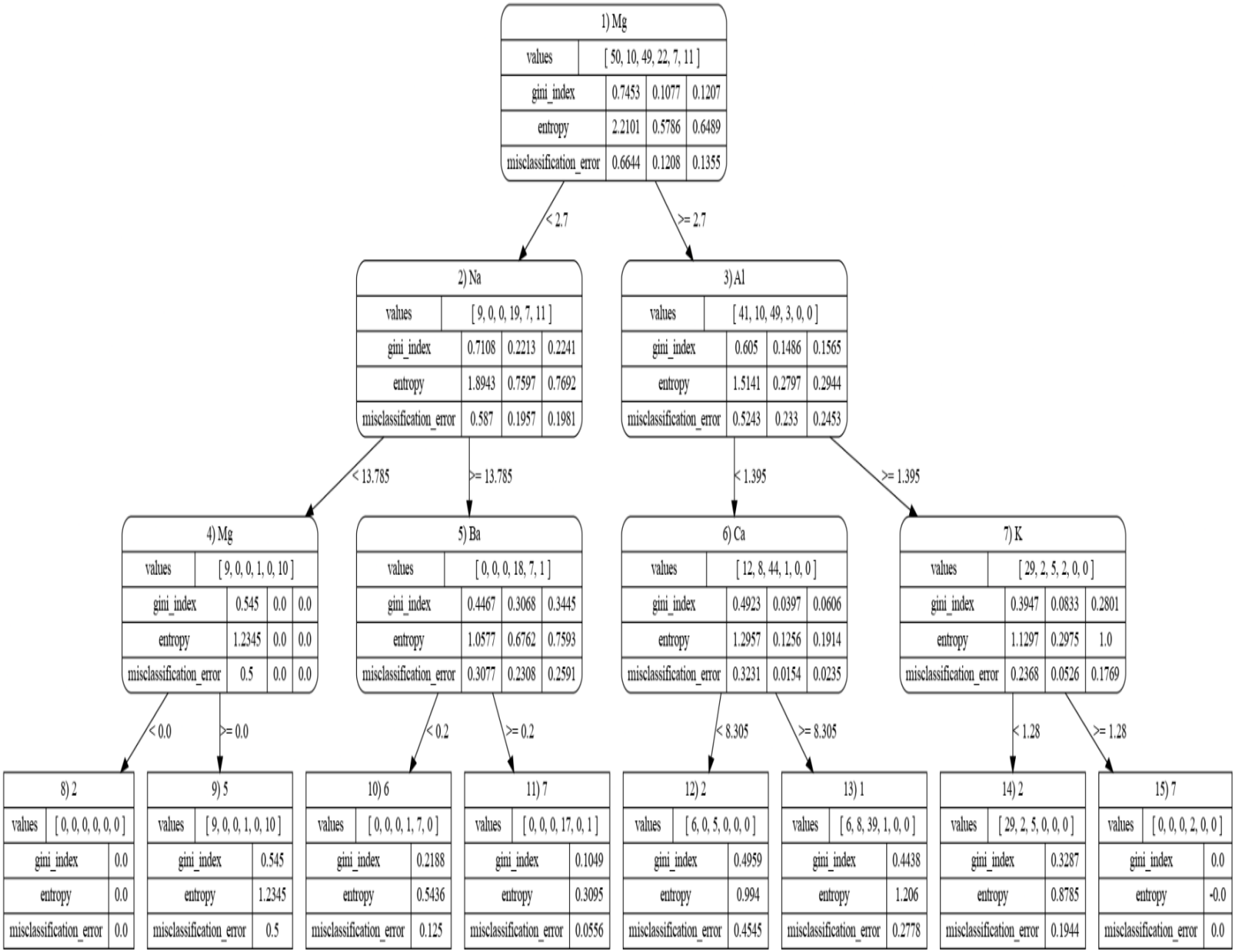
Learning the Decision Tree

The tree was then learnt from the training instances using the previously implemented DecisionTreeClassifier, using information gain as the impurity measure and a maximum depth constraint of 3:

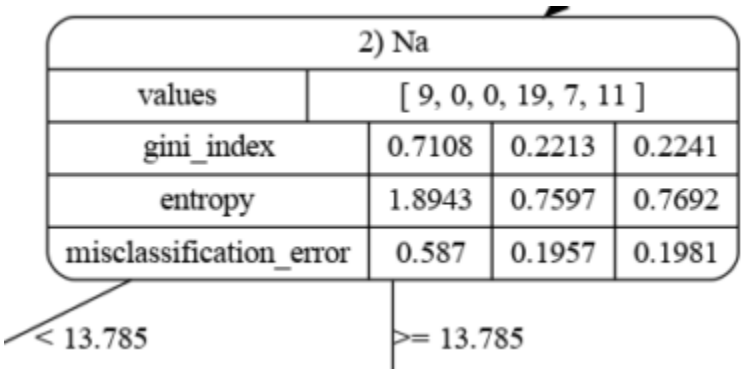
```
tree = DecisionTreeClassifier(
 criterion = 'entropy', # Train via gain in entropy, aka information gain
 use_gain_ratio = False, max_depth = 3
)

tree.fit(X_train, R_train)
```

For the training dataset, the following tree was learnt

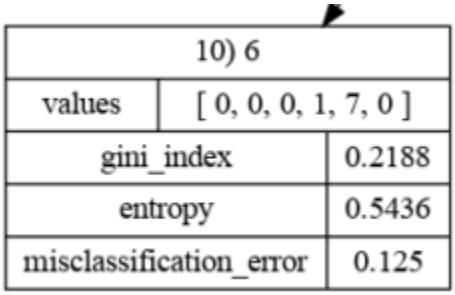

1) Mg
values [50, 10, 49, 22, 7, 11]
gini_index 0.7453 0.1077 0.1207
entropy 2.2101 0.5786 0.6489
misclassification_error 0.6644 0.1208 0.1355
< 2.7
>= 2.7
2) Na
values [9, 0, 0, 19, 7, 11]
gini_index 0.7108 0.2213 0.2241
entropy 1.8943 0.7597 0.7692
misclassification_error 0.587 0.1957 0.1981
3) Al
values [41, 10, 49, 3, 0, 0]
gini_index 0.605 0.1486 0.1565
entropy 1.5141 0.2797 0.2944
misclassification_error 0.5243 0.233 0.2453
< 13.785
>= 13.785
< 1.395
>= 1.395
4) Mg
values [9, 0, 0, 1, 0, 10]
gini_index 0.545 0.0 0.0
entropy 1.2345 0.0 0.0
misclassification_error 0.5 0.0 0.0
5) Ba
values [0, 0, 0, 18, 7, 1]
gini_index 0.4467 0.3068 0.3445
entropy 1.0577 0.6762 0.7593
misclassification_error 0.3077 0.2308 0.2591
6) Ca
values [12, 8, 44, 1, 0, 0]
gini_index 0.4923 0.0397 0.0606
entropy 1.2957 0.1256 0.1914
misclassification_error 0.3231 0.0154 0.0235
7) K
values [29, 2, 5, 2, 0, 0]
gini_index 0.3947 0.0833 0.2801
entropy 1.1297 0.2975 1.0
misclassification_error 0.2368 0.0526 0.1769
< 0.0
>= 0.0
< 0.2
>= 0.2
< 8.305
>= 8.305
< 1.28
>= 1.28
8) 2
values [0, 0, 0, 0, 0, 0]
gini_index 0.0
entropy 0.0
misclassification_error 0.0
9) 5
values [9, 0, 0, 1, 0, 10]
gini_index 0.545
entropy 1.2345
misclassification_error 0.5
10) 6
values [0, 0, 0, 1, 7, 0]
gini_index 0.2188
entropy 0.5436
misclassification_error 0.125
11) 7
values [0, 0, 0, 17, 0, 1]
gini_index 0.1049
entropy 0.3095
misclassification_error 0.0556
12) 2
values [6, 0, 5, 0, 0, 0]
gini_index 0.4959
entropy 0.994
misclassification_error 0.4545
13) 1
values [6, 8, 39, 1, 0, 0]
gini_index 0.4438
entropy 1.206
misclassification_error 0.2778
14) 2
values [29, 2, 5, 0, 0, 0]
gini_index 0.3287
entropy 0.8785
misclassification_error 0.1944
15) 7
values [0, 0, 0, 2, 0, 0]
gini_index 0.0
entropy -0.0
misclassification_error 0.0

Internal nodes (non-leaf nodes) provide the following information:



- Row 1: The index of the node, and the attribute chosen for splitting.
- Row 2: The class-wise count of instances (number of instances per class on this node).
- Rows 3-5: Values of the measure of impurity. The first column contains the value of the impurity measure at the node, the second column contains the gain in respect to the impurity measure, and the third column represents the gain ratio in respect to the impurity measure.

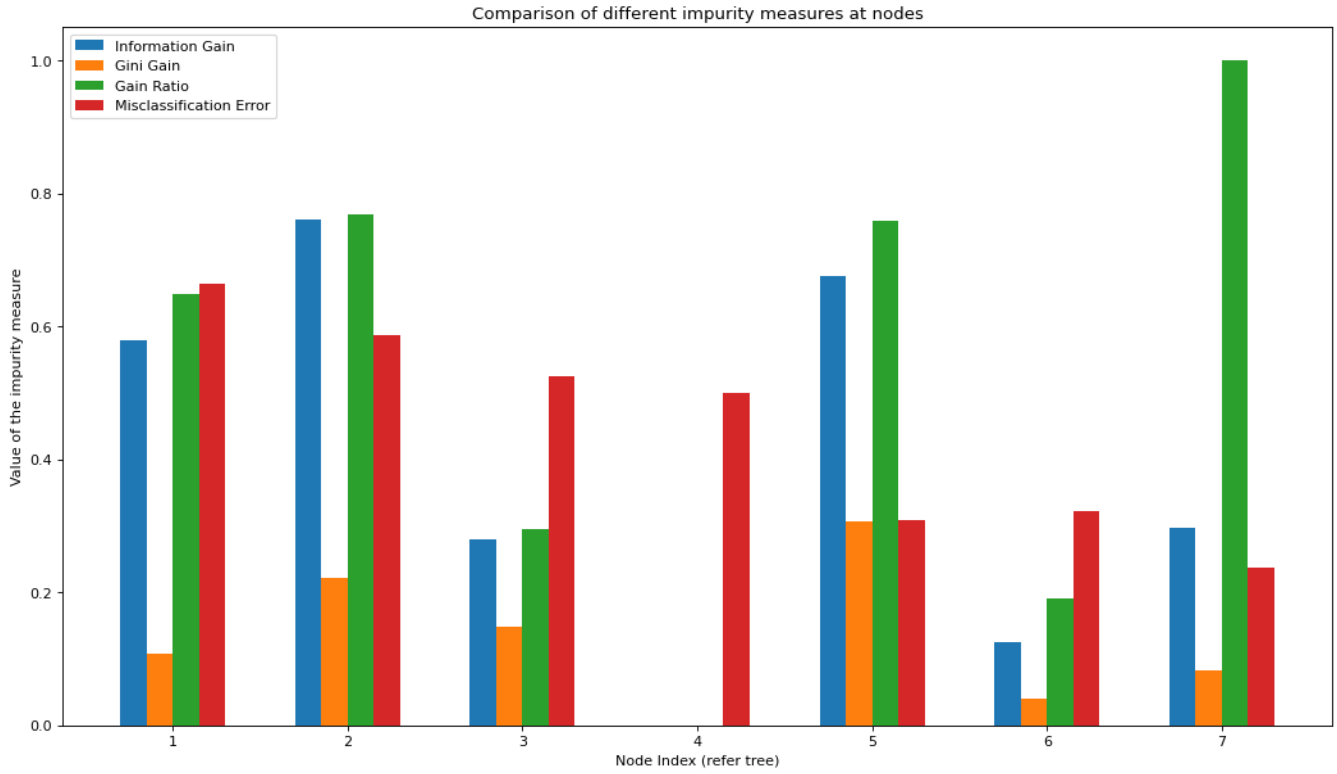
Leaf nodes provide the following information:



- Row 1: The index of the node, and the value of the majority class (Adelie).
- Row 2: The class-wise count of instances (number of instances per class on this node).
- Rows 3-5: Values of the measure of impurity. Only the value of the impurity measure is reported.

Node-wise Comparison of Impurity Measures

A bar plot comparing the values of the different impurity measures at the nodes of the first 2 levels was generated, using matplotlib:



From the bar plot, the following observations can be noted:

- ★ The values of information gain and misclassification error follow a similar trend across the nodes. Both metrics show a decrease in value on the first level (a pure partition using a binary split is not possible, however a binary split on the continuous attribute had the highest gain) followed by an increase owing to purer splits on the nodes of the second level (due to a different attribute being chosen for the split).
- ★ The values of gini gain follow a similar trend as information gain for the subsequent levels (only difference being the scale of values), however for the root node the value is different (the left child of the root shows an increase in gini gain compared to a decrease in information gain).

★ Gain ratio follows a differing trend across the nodes, owing to the scaling induced by the values of split entropy (representing bits/information required to represent branching) which penalizes equi-partitional multi-way splits (multi way splits with equal number of instances being distributed across the branches tend to overfit). However, for the multiway splits induced on nodes 3 and 5, the partitions are highly unbalanced across the child nodes, leading to a lower value of split entropy and thereby a higher value of gain ratio

The tree's performance is summarized via the following classification report:

	precision	recall	f1-score	support
1	0.50	0.71	0.59	21
2	0.60	0.46	0.52	26
3	0.00	0.00	0.00	7
5	0.50	1.00	0.67	2
6	0.50	1.00	0.67	2
7	1.00	1.00	1.00	7
accuracy			0.58	65
macro avg	0.52	0.70	0.57	65
weighted avg	0.54	0.58	0.55	65

Formulating a Hypothesis:

From the noted observations, sufficient information is not available (owing to the selected dataset) to determine in general whether a different impurity measure may perform better than information gain. However, gain ratio is known to produce better splits (especially at nodes where large numbers of partitions are induced, which might be favored by information gain, as gain increases with sparser but multiple partitions), and further the values of gain ratio are high for only a few nodes. So, a tree with splits induced using gain ratio may generalize better (as some attributes with relatively lower gain but much smaller split information value may have been ignored by information gain in favor of splits with slightly higher gain but a considerably larger split information value).

Thus, a hypothesis may be stated as follows: **Gain Ratio may induce better splits across nodes for the selected dataset (preferring binary instead of multiway splits), leading to a higher generalization.**

Verifying the Hypothesis:

To verify the hypothesis, the performance of trees generated upon different measures can be evaluated. If a tree created using an alternate impurity measure always performs better in terms of accuracy, precision or recall then the hypothesis may be assumed to hold.

For verification, trees over all the different impurity measures were induced, and their metrics were evaluated using a classification report.

The results were as follows :

Metrics for tree trained using splits based on gain on gini_index :

	precision	recall	f1-score	support
1	0.53	0.81	0.64	21
2	0.70	0.54	0.61	26
3	0.00	0.00	0.00	7
5	0.50	1.00	0.67	2
6	0.00	0.00	0.00	2
7	0.78	1.00	0.88	7
accuracy			0.62	65
macro avg	0.42	0.56	0.47	65
weighted avg	0.55	0.62	0.57	65

Metrics for tree trained using splits based on gain on entropy :

	precision	recall	f1-score	support
1	0.50	0.71	0.59	21
2	0.60	0.46	0.52	26
3	0.00	0.00	0.00	7
5	0.50	1.00	0.67	2
6	0.50	1.00	0.67	2
7	1.00	1.00	1.00	7
accuracy			0.58	65
macro avg	0.52	0.70	0.57	65
weighted avg	0.54	0.58	0.55	65

Metrics for tree trained using splits based on gain on misclassification_error :

	precision	recall	f1-score	support
1	0.48	0.71	0.58	21
2	0.59	0.50	0.54	26
3	0.00	0.00	0.00	7
5	0.50	1.00	0.67	2
6	0.00	0.00	0.00	2
7	0.86	0.86	0.86	7
accuracy			0.55	65
macro avg	0.41	0.51	0.44	65
weighted avg	0.50	0.55	0.52	65

Metrics for tree trained using splits based on gain ratio on gini_index :

	precision	recall	f1-score	support
1	0.53	0.81	0.64	21
2	0.70	0.54	0.61	26
3	0.00	0.00	0.00	7
5	0.50	1.00	0.67	2
6	0.00	0.00	0.00	2
7	0.78	1.00	0.88	7
accuracy			0.62	65
macro avg	0.42	0.56	0.47	65
weighted avg	0.55	0.62	0.57	65

Metrics for tree trained using splits based on gain ratio on entropy :

	precision	recall	f1-score	support
1	0.50	0.71	0.59	21
2	0.60	0.46	0.52	26
3	0.00	0.00	0.00	7
5	0.50	1.00	0.67	2
6	0.50	1.00	0.67	2
7	1.00	1.00	1.00	7
accuracy			0.58	65
macro avg	0.52	0.70	0.57	65
weighted avg	0.54	0.58	0.55	65

Inference

The trees induced by the various metrics under comparison are mostly similar, with the exception of the tree induced when using misclassification error reduction (gain on misclassification _error) as all of the trees have the extremely similar accuracy, recall and precision values. The hypothesis does not seem to hold for the selected dataset, as the performance of the tree induced using gain ratio is the same as that of the tree induced using information gain and other impurity metrics. Thus, for the selected dataset, gain ratio does not provide an advantage or better performance.

Overall, the choice of impurity measure has little effect on the performance of the tree and the for differing observations the choices are specific to the dataset and the types of attributes present. In general, a tree induced using gain ratio may perform reasonably well as other attributes as it tends to avoid sparser and multiway splits.