Assignment 1 Report
Submitted By: Prabha Pandey
(Roll No. : 2018201053)

q-1-1:

Algorithm Implementation:

Step 1) Read csv file.

Step 2) Split data set randomly in 80%:20% ratio for train data set and validate data set respectively.

Step 3) Select only categorical data.
(name: 'Work_accident','left','promotion_last_5years','sales','salary'])

Step 4) To calculate gain, following formula was used:

Entropy=E(Si)=i(V)=-(q log(q) + (1-q) log(1-q))
: For calculating entropy of class label in complete data set (done in calculate_entropy function)

Average_Attribute_Entropy=I(S,A)= Σ(|Si|/|S|).E(Si)
: For calculating the entropy of attribute under consideration (done in entropy_attribute function)

q: probability of employee leaving company.

Gain(S,A)=E(Si)-I(S,A)

max_gain function, uses the above formula to find the attribute having highest gain.

Step 5) Make decision tree by recursively selecting attribute having highest gain as root node.

Structure of a node of decision tree:

label=name of root node
child=dictionary having unique values of attribute as keys and corresponding subtree as value
poscount=number of positives in the node corresponding to class label
negcount=number of negatives in the node corresponding to class label
isLeaf= if the node is leaf or not

For building decision tree, at every step the attribute having max gain is made as root node and for each of it's unique values a subtree is created, for creating the subtree data set is manipulated by dropping attribute which had max gain.

If information gain is zero then data set can not be splitted further.

Step 6) Predict the value of class label when a particular value from

validate data
set is given, for that predict function is used.

For that you need to traverse over the tree and if leaf node(when pure data
is encountered) is hit or the value you are considering for an attribute does
not have a branch then calculate the number of positives and negatives of
the node you are at present and return true if positives > negatives else
false.

Step 7) Calculate accuracy, precision, recall and f1 score based on following
formulae:

tp: True Positive (actual: "True" & predicted: "True")
tn: True Negative (actual: "False" & predicted: "False")
fp: False Positive (actual: "False" & predicted: "True")
fn: False Negative (actual: "True" & predicted: "False")

accuracy=((tp+tn)/(tp+tn+fp+fn))*100
recall=(tp/(tp+fn))*100
precision=(tp/(tp+fp))*100
f1score=(2/((1/recall)+(1/precision)))

Observation:

True Positive: 1
True Negative: 1691
False Positive: 0
False Negative: 556

accuracy: 75.2669039146%
precision: 100.0%
recall: 0.179533213645%
f1 score: 0.358422939068

Result:

In the given situation, we should try to decrease the number of false
negatives, because if an employee is actually leaving and we declare that he is not
leaving then it would be problematic for the company, So we need to have more
value of recall.

But considering only the categorical data isn't giving high recall, hence
making decision tree in this manner isn't a good approach.

Also the accuracy must be 100% for train data set but it comes out to be
76.4738598443% , as only categorical data was considered.

q-1-2:

Algorithm Implementation:

Step 1) Read csv file.

Step 2) Split data set randomly in 80%:20% ratio for train data set and validate data set respectively.

Step 3) functions of calculate_entropy and entropy_attribute are same as above.

Step 4) For numerical data binary split would be performed, for that the value of node giving max gain would be considered and on one side of the branch records having values greater then selected value would go and all the other records would go on other side.

Step 5) For building decision tree, at every step the attribute having max gain is made as root node and for each of it's unique values a subtree is created, for creating the subtree data set is manipulated by dropping attribute which had max gain only for categorical data and not for numerical data.

Step 6) Prediction and calculation of accuracy is done in the same way as above.

Observation:

True Positive: 515
True Negative: 1671
False Positive: 32
False Negative: 27

accuracy: 97.3719376392%
precision: 94.1499085923%
recall: 95.0184501845%
f1 score: 94.5821854913

Following is the data of number of employee leaving company with respect to time they spent in the company:

Time Spent In Company
Number Of People Leaving
2 years
19
3 years
221
4 years
128
5 years
115
6 years
34

Result:

In the last case we considered only the categorical data and here both
categorical and numerical data are considered, which resulted in an improvement
in recall from 0.179533213645% to 95.0184501845%.

Conclusion is considering only the categorical data is not a good idea, taking
numerical data along with it improves performance.

q-1-3

Algorithm Implementation:

Step 1) Read csv file.

Step 2) Split data set randomly in 80%:20% ratio for train data set and validate
data set respectively.

Step 3) functions of calculate_gini, gini_attribute,calculate_misclassificationrate and
misclassification_attribute are same as above except the formula used is:

Gini = 2*q*(1-q)
MisClassification = min(q, 1-q)

Step 4) Again tree is built and prediction is done, based on which accuracy,
prediction, recall and f1 score are calculated.

Observation:

- Entropy:

True Positive: 515
True Negative: 1671
False Positive: 32
False Negative: 27

accuracy: 97.3719376392%
precision: 94.1499085923%
recall: 95.0184501845%
f1 score: 94.5821854913

- Gini:

True Positive: 475
True Negative: 1718
False Positive: 10
False Negative: 43

accuracy: 97.6402493321%
precision: 97.9381443299%
recall: 91.6988416988%
f1 score: 94.7158524427

- Misclassification rate:

True Positive: 486
True Negative: 1703

False Positive: 21
False Negative: 32

accuracy: 97.6360392507%
precision: 95.8579881657%
recall: 93.8223938224%
f1 score: 94.8292682927

Result:

The comparison between Entropy, gini and Misclassification rate can be seen in above observations.
It can also be observed from the formula, that:
Entropy is slower to calculate as logarithmic function is used.
Gini can only be used when the target variable is binary because formula of gini is q*(1-q)

q-1-4

Algorithm Implementation:

Step 1) Read csv file.

Step 2) Split data set randomly in 80%:20% ratio for train data set and validate data
set respectively.

Step 3) Plot for all possible pairs of attributes of data set, then select the one having
fine decision boundaries.

After judging all possible pairs, found the pair of satisfaction_level and last_evaluation to be the apt one.

Observation:

Result:

The graph represents number of people left in orange color and number of people who did not leave in blue color corresponding to satisfaction_level and last_evaluation.

When the satisfaction level is in the range [0.5,0.7] and [0.9,1.0], employee would stay.

And when satisfaction level is [0.7,0.9] and last evaluation is more than 0.8 then they would leave,else for that range of satisfaction level, they would stay.

When satisfaction level is very low and performance is very high (high last evaluation) then employee would surely leave.

Satisfaction level between [0.3,0.5] and last evaluation between [0.45,0.55] would result in employee leaving the company.

#q-1-5

Algorithm Implementation:

Step 1) Read the csv file

Step 2) randomly split the data in 80%:20% ratio for train and validate data set.

Step 3) Use the some above functions for calculate_entropy, entropy_attribute,
find_node(for numerical attribute) and max_gain.

Step 4) Build decision tree as done before.

Step 5) Calculate the maximum depth of decision tree and number of nodes corresponding to all possible depths, and find error corresponding to each
of these values.

Step 6) Plot graph separately for depth as well as number of nodes.

Observation:

Graph plotted was as follows:

Depth:

Number Of Nodes:

Result:

As the number of nodes and depth increases, error decreases.

#q-1-6

- How To Handle Missing Data In Decision Tree?

  There are several ways to handle missing data in case of decision tree, some of them are stated as follows:

1) Ignore the row altogether if row has any missing attribute value in it, and predict for remaining data set.

2) Take the mean, median or mode of each of the columns and if any column has missing value then take any of the three to fill its value and take appropriate decision.

3) If node is encountered where data is missing then go to all the possible branches from that node and find result for each one of them, then declare the best result out of those results as decision.

Explaining with example:

Suppose we get (a1,b1,e1) as input row, i.e. D is missing, so we will start from A go to a1 branch, then will reach to B, then select b1 branch. Now D is encountered, and no value for D is there in the input, so go to both the sides, d1 and d2. For d1, branch e1 would be taken, and we will get p1 and n1 as one of the result. For d2, p2 and n2 would be another result.

Result would be comparison between p1+p2 and n1+n2, i.e. if p1+p2 > n1+n2 then decision would be positive else negative.

Handling Missing data in our code:-

If at any node missing data is encountered, then calculate the number of rows having positive for class label and number of rows having negative for class label.
If positives are more than negatives then declare decision to be positive, else decision would be declared negative.