**Programming Assignment 1: Decision Trees - Report**

**Chetan KS –** [**katigana@usc.edu**](mailto:katigana@usc.edu)

**Manoj R –** [**rajalban@usc.edu**](mailto:rajalban@usc.edu)

**Ajitesh J – ajayanth@usc.edu**

**Part 1: Implementation**:

We implemented the decision tree in Python. We have used Pandas to extract data from the data file into Data Frames. Using this data frame, we calculate the entropy change with respect to each column field. Depending on the max information gained, we chose a field to be the root of the decision tree. This process is repeated recursively, until we reach a point were entropy is zero. The label of a path from root to leaf, is decided based on the label of the Enjoy column in the subset at the leaf node. At each step, we build the Decision tree gradually. We are printing the decision tree in the format described in the spec.

**Data Structure**:

* Class Node for defining a Node of the decision tree.
* Dictionary in Python, for storing subset of table, while calculating the entropy

**Challenges**:

1. Terminating a tree/recursion, where there is lack of data available from the training set. We are assuming and assigning ‘Yes’ label during such scenarios, as the overall table has more no. of ‘Yes’, than ‘No’.
2. Grouping of data from the input training data to get subset of data. We used ‘Panda’ package, to extract the data from input .csv file and store it in Data Frames. It is easy to form a subset of the tables, in DataFrames.

**Requested Prediction**: We have added the requested test data in test.csv file. We read this file, and based on the values of this test data, we traverse the decision tree. For the given data, we reach a leaf node with ‘**Yes**’ as the prediction.

**Test.csv**:

Occupied,Price,Music,Location,VIP,Favorite Beer

Moderate,Cheap,Loud,City-Center,No,No

**Output of the program run**:

\*\*\*\* The Decision Tree \*\*\*\*

Occupied

Moderate:Location

Ein-Karem:Yes

Talpiot:Price

Normal:Yes

Cheap:No

Mahane-Yehuda:Yes

German-Colony:VIP

No:No

Yes:Yes

City-Center:Yes

Low:Location

City-Center:Price

Normal:VIP

No:Music

Quiet:No

Cheap:No

Ein-Karem:Price

Normal:No

Cheap:Yes

Mahane-Yehuda:No

Talpiot:No

High:Location

City-Center:Yes

German-Colony:No

Mahane-Yehuda:Yes

Talpiot:No

\*\*\*\* End of decision tree \*\*\*\*

\*\*\*\* The label for test data:Yes\*\*\*\*

**Part2: Software Familiarization**: We found that, scikit-learn package in Python offers a good implementation of Decision Tree algorithm. From this package, we used ‘DecisionTreeClassifier’ function of the ‘Tree’ module, to generate a Decision tree for the train data. The prediction obtained for the test data with this tree matched with the prediction obtained from our decision tree. However, the running time, in training the data from scikit package was quicker (60ms) as against our code, which took 400ms.

Some ideas, to improve the performance:

1. We can store the entropy of subset in some table and re-use it, if we come across the same subset for entropy calculation.
2. A better termination condition than, the one described in challenge 1, can improve the prediction error. For example, using the previous node’s subset, in determining the label.
3. We could avoid over-fitting by bottom-up technique called Pruning.

**Part3: Applications:**

**1.Medical Diagnosis** : A lot of different inductive learning algorithms were developed mostly influenced by ID3 inductive algorithm for generating decision trees from examples of solved problems. Ex : ACLS, CART, AQ...

Inductive learning is well known approach to automatic knowledge acqusition for expert systems. This method generates knowledge usually in the form of decision rules, from the concrete problems already solved by some experts.

This technology is well suited for medical diagnosis in small specialized diagnostic problems. Data about correct diagnoses are often available in the form of medical archives in specialized hospitals and departments. The derived set of decision rules can be used to reveal the basic relations and laws in the problem domain in an explicit and transparent form and, of course can be used for diagnosing new patients.

This learning algorithm was applied in several medical domains such as oncology, liver pathology, prognosis of the survival in hepatitis, urology, thyroid diseases, rheumatology, cardiology..

Typically automatically generated diagnostic rules slightly outperformed the diagnostic accuracy of physicians specialists.

Ref ; Kononenko, I., 1993. Inductive and Bayesian learning in medical diagnosis. Applied Artificial Intelligence an International Journal, 7(4), pp.317-337.

2. **Power Systems : Online steady-state security assessment**.

Inductive inference method using decision trees can be applied to Online steady state security assessment of a power system. For each contingency a number of decision rules in the form of a decision tree (DT) is built offline from a preclassified learning set consisting of operating points of the system. For the real time application of the method the DTs corresponding to the foreseen contingencies are searched online to provide optimal guidelines for preventive control of the system. The algorithm developed is applied to the steady state security assessment of a realistic model of the Hellenic interconnected power system comprising 240 busbars, 270 branches, 57 transformers and 30 equivalent generators.

Ref :

Hatziargyriou, N.D., Contaxis, G.C. and Sideris, N.C., 1994. A decision tree method for on-line steady state security assessment. IEEE Transactions on power systems, 9(2), pp.1052-1061.

**Contributions:** Everyone of us, contributed equally to the project. Individually we concentrated on the below aspect respectively.

1. Chetan – Reading the training data in dictionaries, recursive algorithm in finding Decision tree in coding.
2. Manoj - Defining decision trees and traversing decision tree to get test data prediction in coding, Report.
3. Ajitesh – Entropy calculations, running scikit-learn package, for performance comparison, research on applications of Decision trees.