INF552 - Machine Learning for Data Science

Technical Specification and Study Report

[Decision Tree]

Author: Vandit Maheshwari (7701444749), Aviral Upadhyay (8072744512), Rishika Verma(5990934043)

Date: Feb., 7, 2020

Document Control

Author

Position	Name	USC ID
Student	Aviral Upadhyay	8072744512
Student	Vandit Maheshwari	7701444749
Student	Rishika Verma	5990934043

Related documents

Files
Hw1.py
DecisionTree.py
dt_data.txt

Contribution

Name	Participation
Aviral Upadhyay	Implementation (Methods ID3, defaultlib, data retrieval)
Vandit Maheshwari	Implementation (Methods Classify, main), Mathematical Model, Design
Rishika Verma	Implementation (Methods entropy, entropy_of_list, information_gain), Documentation

Table of Contents

1 INTRODUCTION

1.1 About

Decision tree are powerful and popular tools for classification and prediction. Decision tree represent rules, which can be understood by humans and used in knowledge system such as database. A decision tree is a hierarchical model for supervised learning whereby the local region is identified in a sequence of recursive splits in a smaller number of steps. A decision tree is composed of internal decision nodes and terminal leaves. Each decision node m implements a test function fm(x) with discrete outcomes labelling the branches. Given an input, at each node, a test is applied and one of the branches is taken depending on the outcome. This process starts at the root and is repeated recursively until a leaf node is hit, t which point the value written in the leaf constitutes the output.

1.2 Objectives

With the available dataset consisting of records for the previous night-outs with the following attributes, we predict whether we will have a good night-out in Jerusalem for the coming New Year's Eve or not.

- How densely the place is usually **occupied** {High, Moderate, Low}
- How the **prices** are {Expensive, Normal, Cheap}
- Volume of the **music** {Loud, Quiet}
- The **location** {Talpiot, City-Center, Mahane-Yehuda, Ein-Karem, German-Colony}
- Whether you are a frequent customer (**VIP**) {Yes, No}
- Whether this place has your **favorite beer** {Yes, No}
- Whether you **enjoyed** {Yes, No}

2 Functionality

2.1 Description

Based on the decision tree that we have trained using the provided dataset, we need to make a prediction for the given testing data.

2.2 Use Case

For the given data set we need to make a prediction for the below test case where values for each attribute is given as:
occupied = Moderate; price = Cheap; music = Loud; location = City-Center; VIP = No; favorite beer = No

2.3 Data Set

For convenience we made some minor changes in the structure of data but we made sure that data remains as it is. Since we made changes, we also uploaded the new data set along with the program files.
Changes made are as follows:

- In the first line we removed opening and closing parenthesis.
- In all the successive lines we removed the numbering.

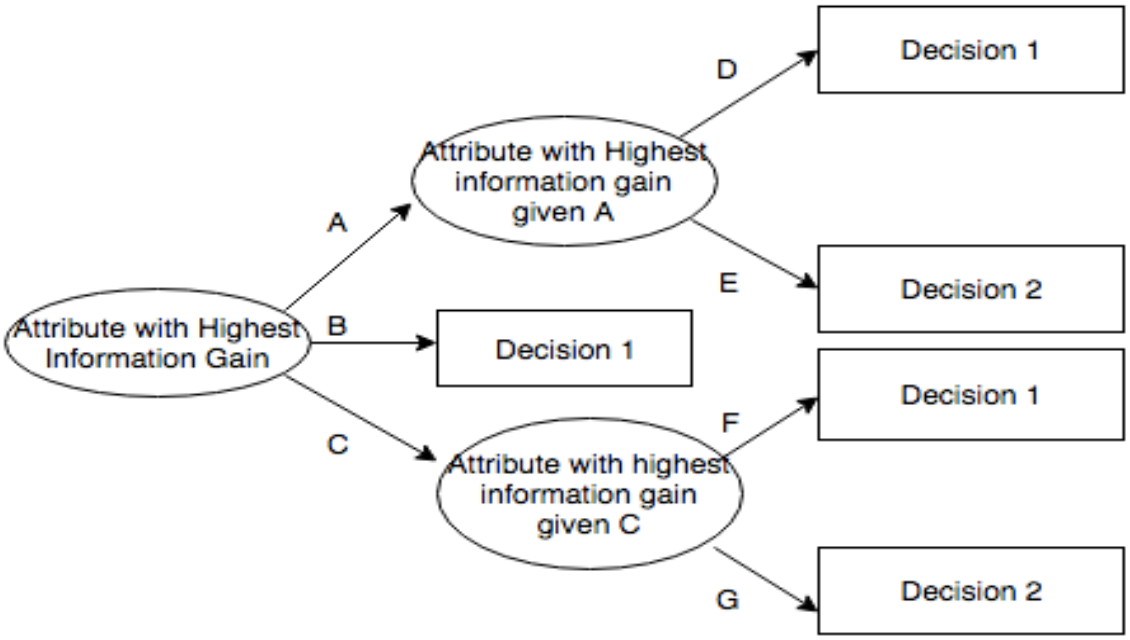
2.4 Test Cases

Prediction for the given test case is done by reading the data through an external text file (testcases.txt) and it also contains some additional testcases listed below
Occupied, Price, Music, Location, VIP, Favorite Beer

- Moderate, Cheap, Loud, City-Center, No, No
- High, Cheap, Loud, City-Center, No, No
- Low, Normal, Quite, City-Center, No, Yes
- Low, Cheap, Quiet, City-Center, No, No

3 Architecture

3.1 Architecture Overview



3.2 Mathematical Model:

- **Entropy :**

Entropy H(S) is a measure of the amount of uncertainty in the data set S.

$$S = -\sum_i P_i \log P_i = -\mathrm{E}_P[\log P],$$

Where, S – The current dataset for which entropy is being calculated. This changes at each step of the ID3 algorithm, either to a subset of the previous set in case of splitting on an attribute or to a sibling partition of the parent in case the recursion terminated previously.

X – The set of classes in S.

P(x) – The proportion of the number of elements in class X to the number of elements in set S.

- **Information Gain :**

Information gain IG(A) is the measure of the difference in entropy from before to after the set S is split on an attribute A. In other words, how much uncertainty in S was reduced after splitting set S on attribute A.

$$IG(S, A) = \mathrm{H}(S) - \sum_{t \in T} p(t)\mathrm{H}(t) = \mathrm{H}(S) - \mathrm{H}(S|A).$$

Where, H(S) – Entropy of set S.

T – the subsets created from splitting set S by attribute A.

P(t) – The proportion of the number of elements in t to the number of elements in set S.

H(t) – Entropy of subset

4 Design

4.1 Data Structure

- Training data set is stored as pandas DataFrame – Two dimensional tabular data.
- Testing data set is stored as pandas DataFrame – Two dimensional tabular data.
- Decision tree is stored in the form of a nested dictionary where the key is attribute name and value is the subtree.
- Entropy and probabilities of individual attributes are stored in a list.
- All information gains are stored in a list.

4.2 Methods

- Hw1.py:
  - main() – This function is used to invoke methods of class decisiontree.
  - defaultlib() –Implementation of decision tree using the Python's sklearn library DecisionTreeClassifier.
    We also perform label encoding using python's preprocessing library to encode target labels with value between 0 and n_classes-1.

- DecisionTree.py:
  - entropy() - Takes a list of probabilities and calculates their entropy.
  - entropy_of_list - Takes a list of items with discrete values and returns the entropy for those items.
  - information_gain() - Takes a DataFrame of attributes and quantifies the entropy of a target attribute after performing a split along the values of another attribute.
  - id3() – Implementation of decision tree learning algorithm through iterative dichotomiser 3.
  - classify() - This algorithm takes an instance and classifies it based on the tree.

4.3 Process

- Import dataset as dataFrame.
- Create the data structure for training dataset and decision tree
- Initialize the decision tree
- Calculate the best information gain and select the best feature to split on.
- Create the child decision tree recursively and select the best feature for each child tree.
- Calculate the entropy and record the label during building tree.
- Return the tree
- Make prediction for the testing data.

4.4 Optimizations and Challenges

4.4.1 Optimization

- As what we design, TRAINING_SET and TESTING_SET can be obtained from input file if resource is available.
- The data for which prediction is to be made, can be stored into a separate text file such that for every entry in that data the code dose not need to be altered or hard coded.

4.4.2 Challenges

- Space complexity will be very high if data set is large. We sacrifice the space so that we can get gradually high speed as we use recursion.
- If the split of a dataset was found empty we return a default value.
- For a homogenous split of a dataset we return the key for that particular attribute directly.
- Data structure of decision tree can be designed better because we waste much space to store the node and relative values such as branches, cases, names of node, etc.

5 Usage Instruction

According to training set data and lower index of training data attribute has higher priority assumption, the decision tree cannot provide the result of prediction for given training and testing data. For instance, in the provided dataset the data:
occupied = Low; price = Normal; music = Quite; location = City-Center; VIP = No; favorite beer = No

This occurs twice but each time with a different value of Enjoy attribute. Hence, for our training module the probability for both the outcome is equal which is why it resulted in ‘none’, but to handle this while classifying we provide the default value as ‘Yes’ (the output with more probability i.e 13/22 here).

Also, for the above instance if value for the attribute favorite beer is changed to yes, the model predicts the value for Enjoy as ‘none’ because this particular set of values was never there for the model to learn and hence the arbitrary result. This type of situation is also handled by the modification specified previously.

Hence, the training data is insufficient to support the complete testing requirements.
There are additional testing data prepared in the file testcases.txt, please execute the program in python version 3.7 environment for more details.

5.1 Script File and Dataset File

- Hw1.py
- DecisionTree.py
- dt-data.txt

5.2 Execution

- Language and version: Python 3.7
- Execute Hw1.py

5.3 Print Tree

```
{'Occupied': {'High': {' Location': {' City-Center': dict_keys([' Yes']),
                                     ' German-Colony': dict_keys([' No']),
                                     ' Mahane-Yehuda': dict_keys([' Yes']),
                                     ' Talpiot': dict_keys([' No'])}},
              'Low': {' Location': {' City-Center': {' Price': {' Cheap': dict_keys([' 
No']),
                                                                ' Normal': {' Music': {' 
Quiet': {' VIP': {' No': {' Favorite.....
```

The above format describes how the tree is being printed. The tree consists of nested dictionaries hence the first key represents the attribute which is selected as the root node of the tree and the value of that key is the subtree associated with it. Each subsequent value is again a key for the next set of values or subtree.

The above tree successfully depicts as to how the ID3 algorithm works. The attribute considered as root node is Occupied, which then splits into all the possible values i.e. ‘High’, ‘Low’ & ‘Moderate’. The next step would be to split the data further on the basis of next attribute for that value of Occupied and so on.
The last value following the chain of attributes results in the prediction for a given set of values for the attributes. If in case, the model cannot predict value for a certain set of information due to any number of reasons such as missing data or no such type of training data then it can be handled by setting a default value(Value with maximum probability) for such cases as described in section 5.

5.4 Predicted Result

Below is the printed decision tree and the predicted value for the given use case.

```
{'Occupied': {'High': {' Location': {' City-Center': dict_keys([' Yes']),
                                     ' German-Colony': dict_keys([' No']),
                                     ' Mahane-Yehuda': dict_keys([' Yes']),
                                     ' Talpiot': dict_keys([' No'])}},
              'Low': {' Location': {' City-Center': {' Price': {' Cheap': dict_keys([' No']),
                                                                ' Normal': {' Music': {' Quiet': {'
VIP': {' No': {' Favorite Beer': {' No': (' '

'No',

1)}}}}}}}},
                                    ' Ein-Karem': {' Price': {' Cheap': dict_keys([' Yes']),
                                                              ' Normal': dict_keys([' No'])}},
                                    ' Mahane-Yehuda': dict_keys([' No']),
                                    ' Talpiot': dict_keys([' No'])}},
              'Moderate': {' Location': {' City-Center': dict_keys([' Yes']),
                                         ' Ein-Karem': dict_keys([' Yes']),
```

```
                                    ' German-Colony': {' VIP': {' No': dict_keys([' No']),
                                                               ' Yes': dict_keys([' Yes'])}},
                                    ' Mahane-Yehuda': dict_keys([' Yes']),
                                    ' Talpiot': {' Price': {' Cheap': dict_keys([' No']),
                                                            ' Normal': dict_keys([' Yes'])}}}}}}
```

```
   Occupied   Price  Music      Location  VIP  Favorite Beer precdiction
0  Moderate   Cheap   Loud   City-Center   No             No      ( Yes)
```

5.5 Additional Testing Data

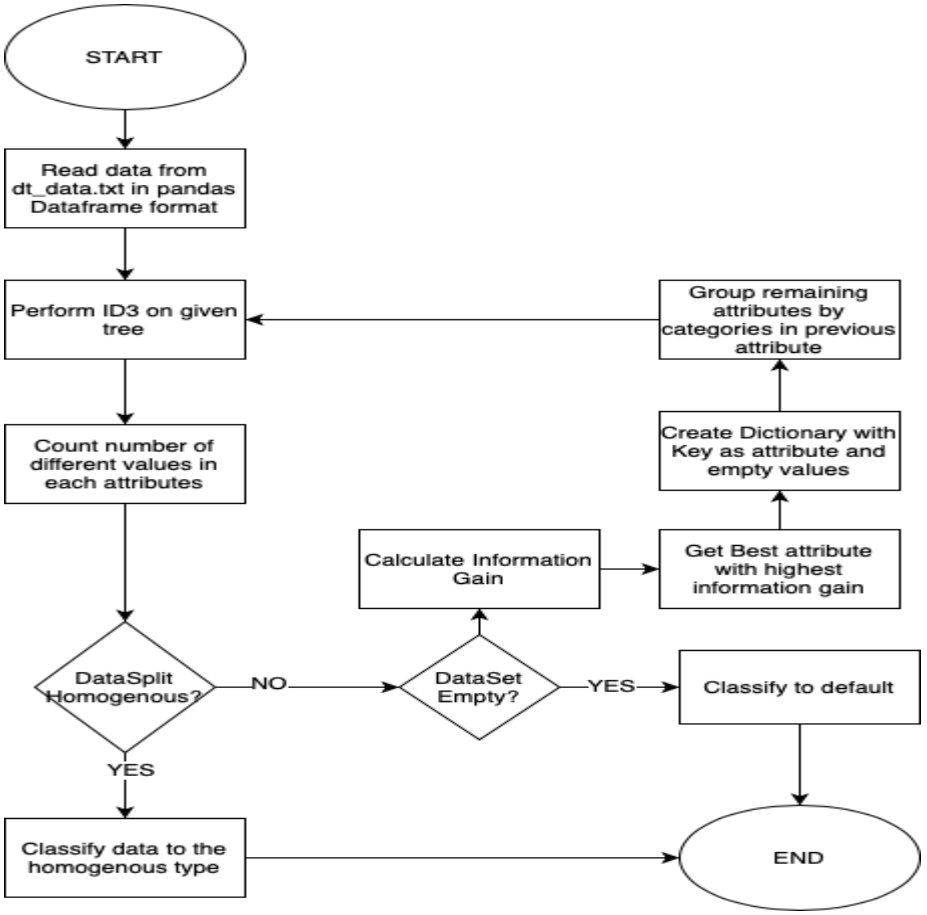
Additional test cases are listed in section 2.4 and their predictions are as follows.

```
Occupied    Price   Music      Location  VIP  Favorite Beer prediction
0  Moderate    Cheap    Loud   City-Center   No             No      ( Yes)
1      High    Cheap    Loud   City-Center   No             No      ( Yes)
2       Low   Normal   Quite   City-Center   No            Yes         Yes
3       Low    Cheap   Quiet   City-Center   No             No       ( No)
```

6 Software Familiarization

6.1 Overview

Decision Tress are a non-parametric supervised learning method for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. The best advantage of implementing decision tree is that they perform relatively well even if its assumptions are somewhat violated by the true model from which the data were generated.



As the above figure depicts, the first step towards building a decision tree is to import data from an external file in form of pandas DataFrame and create a decision tree by executing ID3 algorithm, which checks if the datasplit is homogenous or empty and returns the class, if not it calculates the total entropy of the label and information gain(using probabilities and entropy) for every attribute.

Next, it selects the attribute with highest information gain and creates a dictionary with attribute as key and empty values. It then splits the data and performs the above steps recursively to find the value which is a subtree.

After the model is trained we can use that model to predict values for a given test data.

Requirements: Pandas, Numpy python library for implementation of ID3 algorithm. Additionally sklearn for default library implementation.

6.2 Library Function

6.2.1 DecisionTreeClassifier

From Python's library, scikit-learn Scikit-learn is a free software machine learning library for the Python programming language. It features classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. DecisionTreeClassifier is a class under sklearn capable of performing multi-class classification on a given dataset. It takes as two input arrays: an array X, sparse or dense holding the training samples, and an array Y of integer values, holding the class labels for the training samples.

6.2.2 Advantages

- Simple to understand and to interpret.
- Requires little data preparation such as use of Label Encoder or One Hot Encoder.
- The cost of using the tree is logarithmic in the number of data points used to train the tree.
- Able to handle multi-output problems.
- Since, we can validate a model using statistical test it makes it possible to account for the reliability of the model.

6.2.3 Disadvantages

- Decision tree learners can result into creation of over complex trees that do not generalize the data well. This phenomenon is called Overfitting. Mechanisms such as pruning or setting the maximum depth of the tree are required to avoid this problem.
- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated.
- Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree.

6.3 Improvements over current techniques

- We can simplify our data structure and functions with the usage of nested list.
- We can implement pruning which sets the minimum number of samples required at a leaf node.
- To avoid overfitting, we can also set the maximum depth of the tree .
- Adaptive boosting or AdaBoost algorithm is another appreciated way to improve performance. The approach here is that sequential and subsequent classifiers are closely related, quite opposite of random forests.

- A good way to ensure that our model works is handling of missing values instead of just excluding those cases.

7 Applications

7.1 Customer Relationship Management

A frequent used approach to manage customers relationships is to investigate how individuals access online services. Such an investigation is mainly performed by collecting and analyzing individuals' usage data and then providing recommendations based on the extracted information. According to a study, decision trees were applied to investigate the relationships between the customers' needs and preferences and the success of online shopping. In that study, the frequency of using online shopping is used as a label to classify users into two categories: (a) users who rarely used online shopping and (b) users who frequently used online shopping.
In terms of the former, the model suggests that the time customers need to spend in a transaction and how urgent customers need to purchase a product are the most important factors which need to be considered. With respect to the latter, the created model indicates that price and the degree of human resources involved (e.g. the requirements of contacts with the employees of the company in having services) are the most important factors. The created decision trees also suggest that the success of an online shopping highly depends on the frequency of customers' purchases and the price of the products. Findings discovered by decision trees are useful for understanding their customers' needs and preferences.

7.2 Fault Diagnosis

Another widely used application in the engineering domain is the detection of faults, especially in the identification of a faulty bearing in rotary machineries. This is probably because a bearing is one of the most important components that directly influences the operation of a rotary machine. To detect the existence of a faulty bearing, engineers tend to measure the vibration and acoustic emission (AE) signals emanated from the rotary machine.
However, the measurement involves a number of variables, some of which may be less relevant to the investigation. Decision trees are a possible tool to remove such irrelevant variables as they can be used for the purposes of feature selection. Through feature selection, three attributes were chosen to discriminate the faulty conditions of a bearing, i.e., the minimum value of the vibration signal, the standard deviation of the vibration signal, and kurtosis. The chosen attributes, subsequently, were used for creating another decision tree model. Evaluations from this model show that more than 95% of the testing dataset has been correctly classified. Such a highly accurate rate suggests that the removal of insignificant attributes within a dataset is another contribution of decision trees.

8 Future Applications

8.1 Reliability of Findings

Although decision tree is a powerful tool for data analyses, it seems that some data are misclassified in the decision tree models. A possible way to address this issue is to exploit the extracted knowledge by human-computer collaboration. In other words, experts from different domains use their domain knowledge to filter findings from the created model. By doing so, the irrelevant findings can manually be removed. However, the drawback of employing such a method is the necessity of large investment as it involves the cost and time of experts from different domains.