INF 551 Machine Learning for Data Informatics

Homework #1

Group Members

1. Tanmay Diwan
   USCID: 5047-3478-66
2. Kshitija Godse
   USCID: 6833-7845-04

Table of Contents **Page No.**

1. Problem 1-: Decision Trees

1.1 Abstract -

Decision trees are directed graphs that are used to classify or regress data. Each node uses a rule to split the data based on some attribute. As a tree often will contain multiple nodes, this allows multiple rules to be used to classify a data point. A decision tree is used to analyze large amounts of data and results with the most probable outcomes. Each condition is an internal node and each outcome is an external node. Decision trees can be constructed using the ID3 algorithm that splits the data by the feature with the maximum information gain recursively for each branch. It is simply a graphical representation of a sequential decision process.

Entropy- Entropy is a measure of the amount of uncertainty associated with a set of probabilities. Entropy may be used to rank attributes in terms of the reduction in the uncertainty of the class label for a given instance that can be expected if the value of the attribute becomes known for that instance. For a two class case:

$\text{Entropy}(S) = -p_1\log_2(p_1) - p_2\log_2(p_2)$

Cases with more classes follow by summing over the classes.

Information Gain: In order to minimize impurity, we want to choose splits that maximize the information gain, a value that reflects the decrease in entropy that we get by choose a certain split.

1.2 Approach and Procedure-

1.2.1 Data Structure & Functions:

- We started with recursively building the tree on dataset provided. As given in the data set we considered enjoy as a class label.
- ID3 Algorithm we used to construct a decision tree. We did this programming in Python language.
- The main data structure used for creating the each node in the tree is *Class*.
- Basic functions:
    - entropy: This function simply uses the formula to calculate the entropy of each attributes.
    - splitData: This function splits the data on features(attributes).
    - gain: Calculates the information gain.
    - printDecisionTree: To print decision tree
    - predictor: It takes root and test case query as a input & predicts if you have a good night out in Jerusalem.
    - uniquecounts: It returns the total number of ‘YES’ and ‘NO’ for attribute ‘Enjoy’.
    - baseCriteria: For a particular attribute it selects only one value from the all same possible class label values

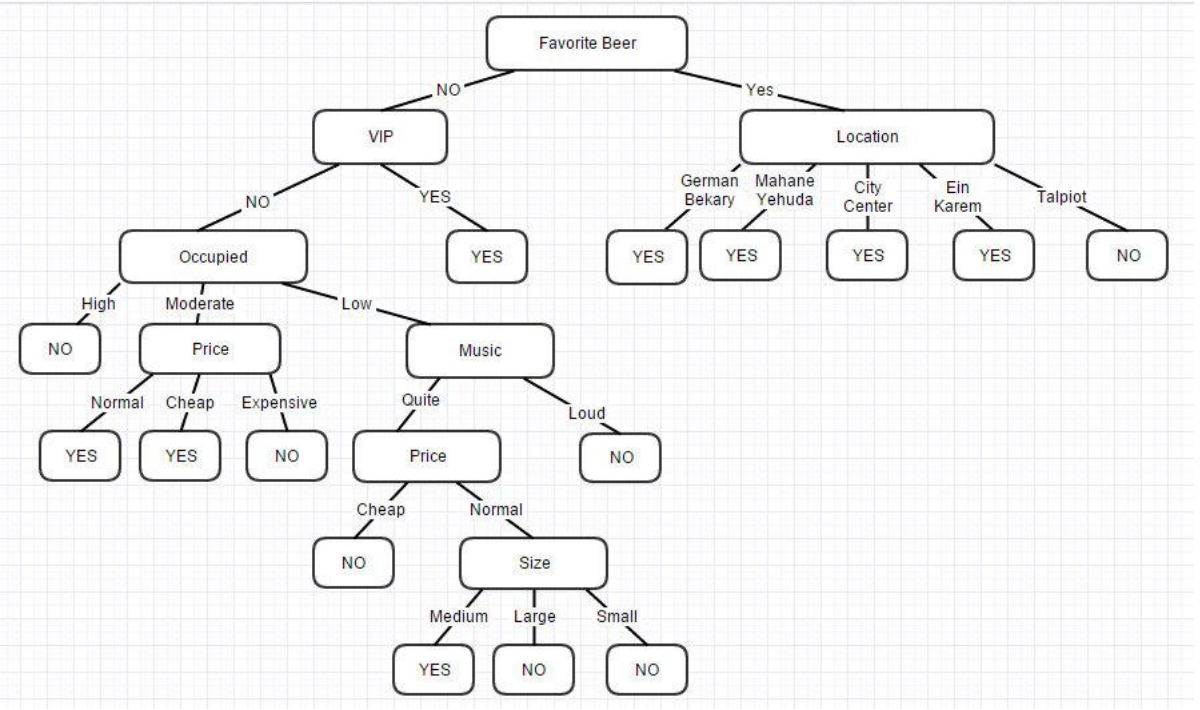
1.2.2 Pre-Processing:

- cleandata: It removes unwanted data from dataset and returns only the attributes with their values. Unnecessary data will be removed.

1.2.3 if-then Rules:

R1:	IF (favorite beer=NO) AND (VIP=NO) AND (Occupied=High) THEN Enjoy=NO
R2**:**	IF (favorite beer=NO) AND (VIP=NO) AND (Occupied=Moderate) AND (Price=Normal) THEN Enjoy=YES
R3:	IF (favorite beer=NO) AND (VIP=NO) AND (Occupied=Moderate) AND (Price=Cheap) THEN Enjoy=YES
R4:	IF (favorite beer=NO) AND (VIP=NO) AND (Occupied=Moderate) AND (Price=Expensive) THEN Enjoy=NO
R5:	IF (favorite beer=NO) AND (VIP=NO) AND (Occupied=Low) AND (Music=Loud) THEN Enjoy=NO
R6:	IF (favorite beer=NO) AND (VIP=NO) AND (Occupied=Low) AND (Music=Quite) AND (Price=Cheap) THEN Enjoy=NO
R7:	IF (favorite beer=NO) AND (VIP=NO) AND (Occupied=Low) AND (Music=Quite) AND (Price=Normal) AND (Size=Medium) THEN Enjoy=YES
R8:	IF (favorite beer=NO) AND (VIP=NO) AND (Occupied=Low) AND (Music=Quite) AND (Price=Normal) AND (Size=Large) THEN Enjoy=NO
R9:	IF (favorite beer=NO) AND (VIP=NO) AND (Occupied=Low) AND (Music=Quite) AND (Price=Normal) AND (Size=Small) THEN Enjoy=NO
R10:	IF (favorite beer=NO) AND (VIP=YES) THEN Enjoy=YES
R11:	IF (favorite beer=YES) AND (Location=German Colony) THEN Enjoy=YES
R12:	IF (favorite beer=YES) AND (Location=Mahane-Yehuda) THEN Enjoy=YES
R13:	IF (favorite beer=YES) AND (Location=City Center) THEN Enjoy=YES
R14:	IF (favorite beer=YES) AND (Location=Ein-Karem) THEN Enjoy=YES
R15:	IF (favorite beer=YES) AND (Location=Talpiot) THEN Enjoy=NO

1.2.4 Decision Tree Diagram:



1.2.5 Pseudo Code:

Psuedo Code for the buildDecisionTree function
The function takes 3 objects as input:
1.)data: Data set or subset as per the call,the list of records.
2.)root:Object of the class Tree,ie Node
3.)Remaining Features:List of indices of the remaining features to split the tree.

If No RemainingFeatures left
then return Node

Extract BestFeature index from set of ReamainingFeatures

If Information gain of Data split on BestFeature is zero
then return Node

Assign the BestFeature Index to the Node attribute splitFeatureIndex

For each dataSubset split on bestFeature do:
Create a child object of class Tree
Assign Node to parent attribute of child_node
Extract the bestFeature value and assign to the attribute splitFeatureValue

Append the child_node to the list attribute children of Node

call buildDecisionTree function with parameters dataSubset,child_node,list of (remainingFeatures - bestFeature)

Return Node

1.3 Code Level Optimization:

- List Comprehensions:
  - Most of the functions use Python's in built method of "List comprehensions" to generate lists.
  - For example line number 70
  - dataSubset.append([row for row in data if row[featureIndex] == aValue])
  - In this particular line of code the append method of list dataSubset appends a list of rows from the list data, satisfying the given if statement.List comprehension helps to generate a list instances of the given object.

- Lambda functions:
  - Python supports the creation of anonymous functions (i.e. functions that are not bound to a name) at runtime, using a construct called "lambda".
  - For example,
  - To calculate log2 ie log to the base 2,as log2 is not supported in python,we calculate it by change of base technique
  - line number 46 log2=lambda x:log(x)/log(2)
  - In this code the log2 variable is assigned a lamba function which takes a variable x as input and gives log2 of that number by using available math library log10 method.

1.4 Challenges:

- The program uses the ID3 algorithm which recursively builds the tree.Extracting the best feature and splitting it accordingly was complex.To extract the index of the best feature by comparing the Information gains of all the features is complicated.
- In the following piece of code bestFeature = max(remainingFeatures, key=lambda index: gain(data, index)),we use the max function with key and lambda expresssion.To modify the object before comparison or to compare based on a particular attribute/index we use the key argument.The key takes each index from the remainingFeatures list and calls gain function with the data subset and the index.Finally we get a list of Information gains where max method gives the index of feature with maximum information gain.

1.5 Predictions:

A prediction for (size = Large; occupied = Moderate; price = Cheap; music = Loud; location = City-Center; VIP = No; favorite beer = No)

The Query posed:

['Size', ' Occupied', ' Price', ' Music', ' Location', ' VIP', ' Favorite Beer']

['Large', ' Moderate', ' Cheap', ' Loud', ' City-Center', ' No', ' No']

Yes you will enjoy the night-out!

1.6 Result:

```
C:\Users\Diwan\Documents\Programs>python DecisionTreeID3.py
            German-Colony,  Yes
        Yes,  Location
            Mahane-Yehuda,  Yes
            City-Center,  Yes
            Ein-Karem,  Yes
            Talpiot,  No
 Favorite Beer
            Yes,  Yes
        No,  VIP
                High,  No
            No,  Occupied
                    Expensive,  No
                Moderate,  Price
                    Normal,  Yes
                    Cheap,  Yes
                    Loud,  No
                Low,  Music
                            Large,  No
                        Normal, Size
                            Small,  No
                            Medium,  Yes
                    Quiet,  Price
                        Cheap,  No

 The Query posed:
['Size', ' Occupied', ' Price', ' Music', ' Location', ' VIP', ' Favorite Beer']
['Large', ' Moderate', ' Cheap', ' Loud', ' City-Center', ' No', ' No']
Yes you will enjoy the night-out!

C:\Users\Diwan\Documents\Programs>
```

2. Problem 2 -: K-Means Clustering

2.1 Abstract-

K-means is a unsupervised learning algorithm. It solves clustering problems. It classifies a given data set through a certain number of clusters. The main idea is to define k centers, one for each cluster and then calculate the distance of these centroids with each remaining data points. For calculating the distance Euclidian Distance and Manhattan Distances are used. From the results obtained, each data point is been assigned to the nearest cluster. Then again this distance is calculated amongst the clusters and by taking the mean value of all the data points new centroid is obtained. Outliers are the main problem in K-means clustering algorithm.

2.2 Approach and Procedure:

2.2.1 Data structure and functions used:

- Initially we randomly selected the centroids using randint() function in Python.
- initializeClusters()- Forms the clusters for selected centroids.
- newCentroidsKmeans()- re-calculates the centers among the clusters.
- kMeans()- It is a recursive function which keep updating the cluster by calculating for new centroids and assigning the data points.
- plotter()- It plots the final cluster in a pictorial way.

2.2.2 L1 Distance and L2 Distance:

- L1 Distance is also called as Manhattan distance. It is the distance between two points measured along axes at right angles. In a plane with p1 at (x1, y1) and p2 at (x2, y2), it is |x1 - x2| + |y1 - y2|.
- L2 Distance is also called as Euclidean distance. It is the straight line distance between two points. In a plane with p1 at (x1, y1) and p2 at (x2, y2), it is $\sqrt{((x1 - x2)\text{^}2 + (y1 - y2)\text{^}2)}$.

2.3 Code Level Optimization:

- Dictionaries: Python dictionaries are very powerful built-in data structures which are a sort of hash-map. It contains a series of key -> value mappings where the "key" is of any type that is hash able. The "value" may be of any type and value types need not be homogeneous. The clusters formed in the program are represented by dictionaries where key is the cluster number and value contains the list of data points associated with that cluster.

- Matplotlib.pyplot: We use a standard library for plotting the points on a X-Y plane.

2.4 Result:

```
[Anaconda2] C:\Users\Diwan\Documents\Programs>python km1.py
Which distance to use for calculation 1.)L1/Manhattan 2.)L2/Eucidean 2
[[0.2075693146603773, 0.12365060556603773], [9.923003253480768,
0.15096781351923
```

078], [4.692988296511112, 5.315995387888889]]
Centroid 1[-0.08667900622000003, -0.005460671820000012]
[[0.510039315, -0.033454921], [0.535879992, -0.544984943], [-0.030554412, 0.7333
89872], [0.127854799, -0.650548946], [0.715793713, 0.853664924], [-0.581730577,
0.633250846], [0.025365753, -0.196067087], [0.31512883, -0.114040657],
[0.173119
979, 0.672802088], [-0.735966061, 0.615308491], [0.746495179, 0.025851566], [-0.
386343423, -0.628925622], [1.284563957, -0.219195524], [-0.048681172,
0.29153848
6], [-1.55638947, -0.04485903], [0.395953933, 0.494127144], [-1.326373584, 0.644
738349], [0.877523801, -0.183924509], [0.250195385, -0.874326245],
[0.781958321,
0.45034738], [0.58644216, -0.576533163], [-0.406803883, 0.161830401], [-0.92433
6618, -0.918433377], [-0.849464839, -0.451029601], [-0.60671378, -0.144416903],
[-0.665808152, 0.342018893], [-1.006567109, -0.776052519], [-0.209310532, 0.6038
22258], [0.457496263, 0.649517422], [-0.146233036, 0.470677195], [0.011270991,
0
.753474974], [-1.2159853, -0.388557274], [-0.326760934, -0.59826327], [-0.656723
353, -0.7606785], [0.491927392, -0.245524001], [-1.541393849, -0.188592587], [0.
446950982, 0.238815419], [0.462508086, -0.242278427], [0.006692734,
0.864144208]
, [-0.4594557, -0.066167696], [-0.82299408, -0.853014691], [-0.669972989, 0.3455
17373], [-0.781926012, -0.255114051], [-0.969241215, -0.488606904],
[1.157738921
, 0.048978901], [0.800929342, 0.512101627], [0.605962008, 0.576236041], [0.45949
1212, -0.38556238], [-0.552806475, 0.834823601], [0.917303196, -1.260858222]]

Centroid 2[9.923003253480768, 0.15096781351923078]
[[6.544734089, 1.503317083], [8.425299789, 2.645545801], [12.16058412, -
0.362125
835], [10.22058808, 0.009016604], [9.273663197, -1.252746611], [10.58896571, 1.2
71951549], [10.46418918, 0.204367166], [8.39829614, 1.587436854], [10.17837225,
0.248014021], [10.23183027, -0.399234152], [11.30847308, 0.992310253],
[7.877767
016, 0.084727778], [10.02904103, 0.639421298], [10.66429482, 0.754367268], [10.6
3529107, -0.47353238], [10.78690808, 0.909634764], [10.25152739, -0.986960442],
[10.16240361, 0.521375024], [10.22228819, -0.717850129], [8.106908717,
1.1273924
87], [11.43037929, -0.766911821], [10.07089281, 0.938592588], [8.005566045, -1.3
67474706], [10.61661672, 0.745495925], [10.65859351, 1.163325858],
[10.63190368,
1.746369341], [8.48653224, 0.156627329], [10.51752782, 0.17169319],
[9.48934315
7, -0.190821782], [11.36968167, 1.283354542], [9.157291133, 0.912298291], [9.453

725245, 0.024082023], [8.613609933, 0.012923723], [9.089636524, 1.130473664], [9
.62386144, 0.071194103], [10.87284469, -0.872054667], [9.67196997, -0.767002838]
, [9.647099157, 1.218331242], [11.47927287, 0.08871278], [10.58651356, 0.155673799], [10.91537566, -0.161546934], [9.212519802, -1.195793489], [8.505171787, 0.454283639], [11.41916539, -0.109842662], [8.860945078, -0.147791862], [10.5774327
2, -0.238018987], [12.14314995, -1.011735108], [10.61106931, -2.581639716], [10.19259728, -2.288667349], [9.76625099, -0.183011689], [9.225220637, 1.193757348],
[8.562983285, -0.040977873]]

Centroid 3[4.719158277729167, 5.1259647529375005]
[[4.791128395, 4.370299589], [4.346058487, 4.018067016], [8.231033082, 6.184286192], [3.888145888, 5.267364444], [5.025547369, 6.794483892], [3.807810649, 6.987931983], [5.910493777, 4.093302008], [5.478176272, 5.516863136], [6.62794025, 3.722270626], [2.343750659, 4.57456785], [6.068372741, 5.4309343], [4.23229562, 5.206003122], [5.195116303, 2.36092694], [2.488663641, 6.355903079], [5.28271118, 5.648397944], [5.288476, 5.89259064], [4.602393734, 2.647978123], [5.666789848, 5.701026649], [4.804948111, 5.157872987], [3.685961459, 4.082833651], [3.999606696, 5.384602232], [4.790463114, 7.188303292], [3.796399358, 5.737118242], [5.820319289, 3.416805651], [6.68456776, 6.001567197], [4.863746388, 3.705479154], [7.339464614, 5.503620224], [4.413579111, 4.893457373], [4.278927519, 2.989392822],
[3.570462507, 5.05118644], [4.187363007, 5.006079291], [5.049608903, 5.681134719], [4.379925229, 6.509004434], [1.708022242, 4.50515623], [6.236542245, 5.405212129], [4.109295861, 3.899938059], [4.442377563, 5.751869679], [2.913106881, 5.253241651], [3.342571332, 5.414482622], [5.537613951, 1.817610623], [4.612527375,
6.265716757], [7.064695701, 5.108753941], [4.029541008, 7.011044352], [6.293670058, 4.343930914], [3.877198382, 5.839108313], [3.592349817, 9.653682134], [2.785143946, 5.398666109], [5.034694009, 3.296239386]]

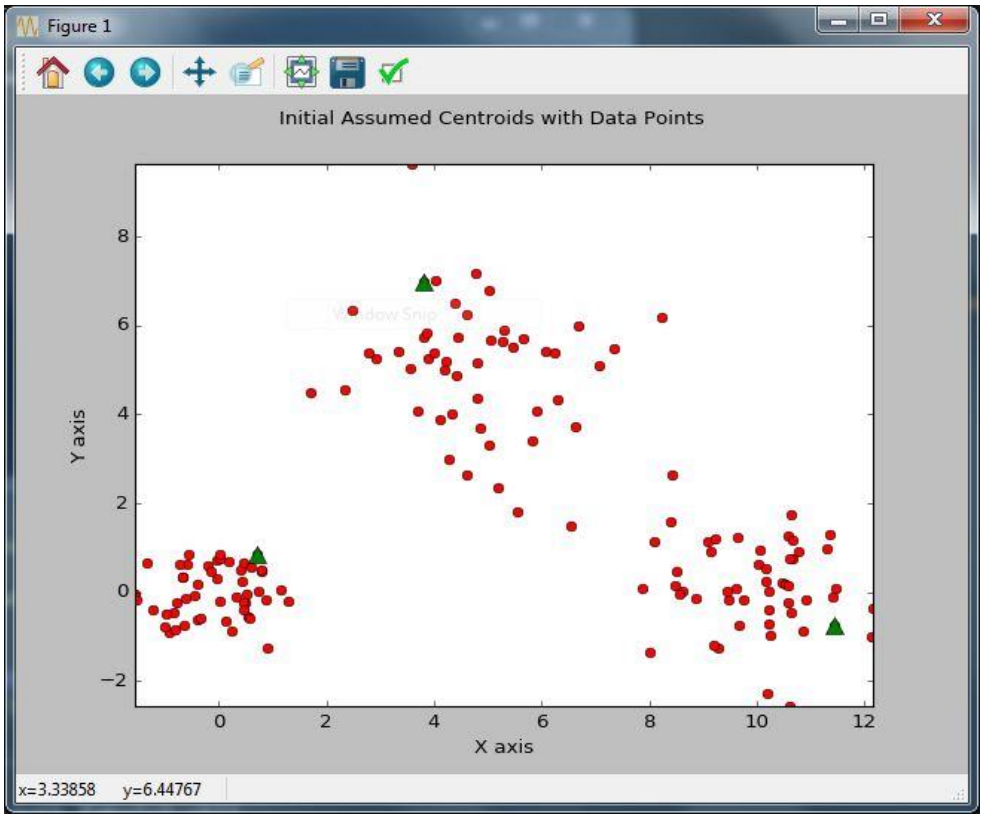


Figure 1
Initial Assumed Centroids with Data Points
Y axis
X axis
x=3.33858 y=6.44767

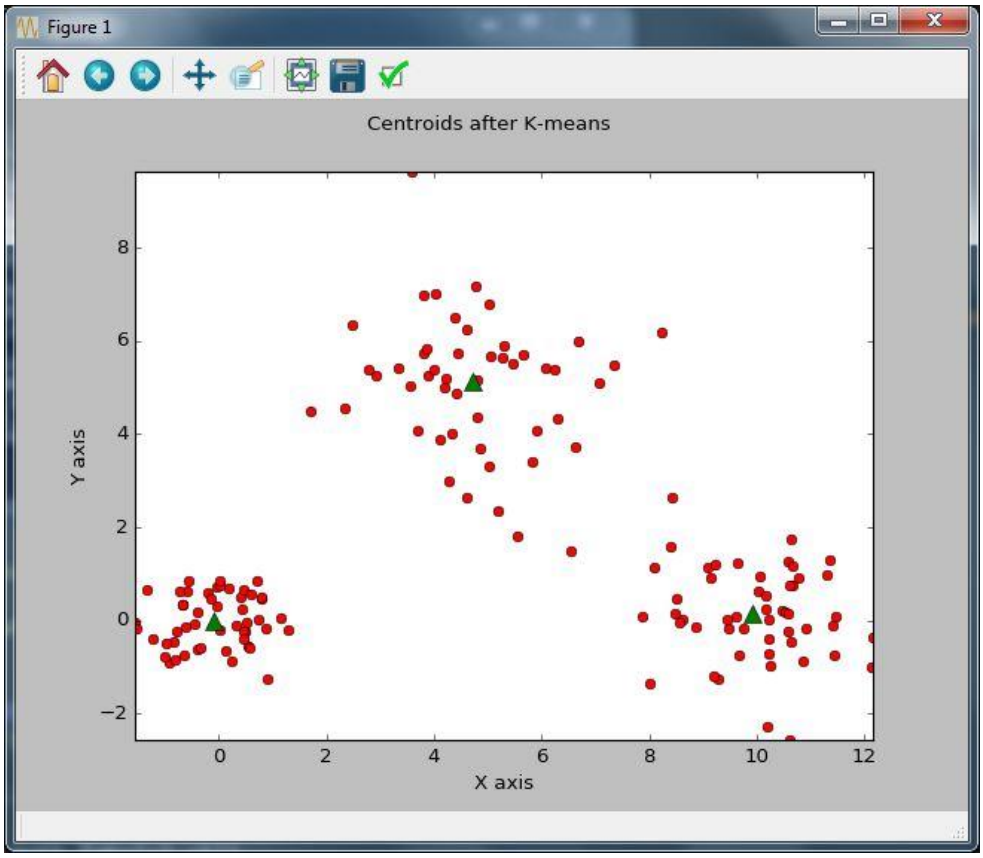


Figure 1
Centroids after K-means
Y axis
X axis