COMP4432 Machine Learning

Assignment #1 (Due Sunday 11:59pm, 27 Feb 2022)

Instructions:
- Answer all questions.
- Interpret the questions logically, show your steps and **write down your assumption(s) when necessary**.
- Please submit your answer to L@PU before the due date.
- Late Submission Policy
  - 3-hour "grace period" is given.
  - 10% off for every 3-hour late
- Plagiarism Policy
  - Both giver and receiver subject to the same penalty below
  - All the students involved not only will receive 0 marks for this assessment, but also will have an additional 50% penalty applied, e.g., 5 marks for a 10-mark assessment.

1. Suppose you are asked by the Octopus Card Company to analyze the following Card Database.

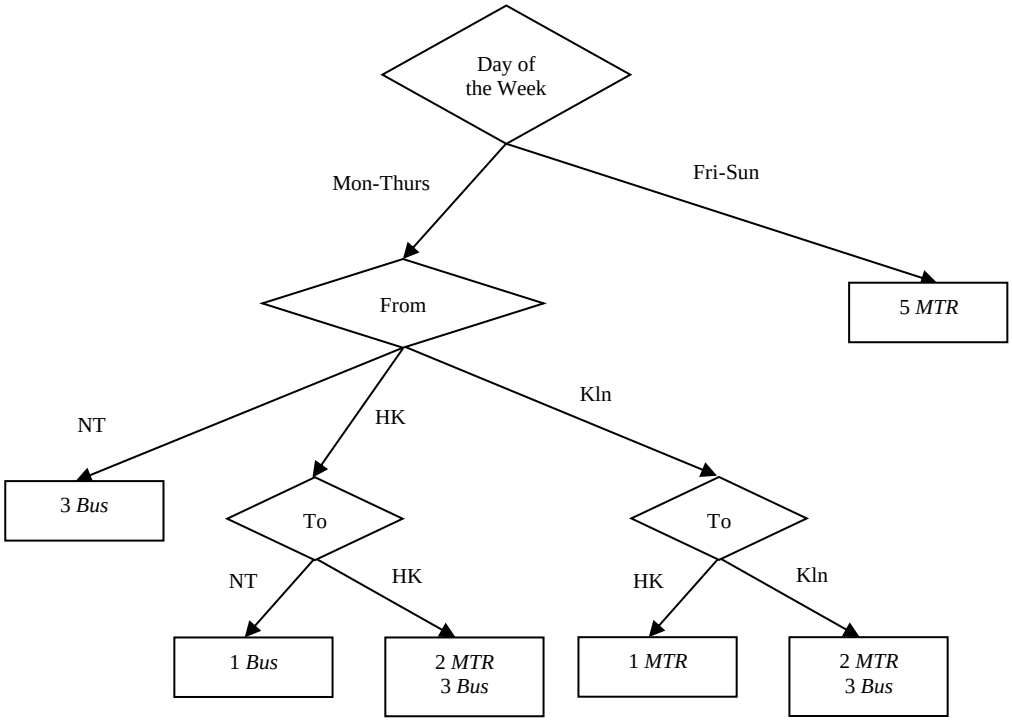
Octopus Card Database for Training

Octopus ID	*Day of the Week*	*From*	*To*	*Transportation Taken*
1	Mon-Thurs	HK	HK	MTR
2	Mon-Thurs	Kln	Kln	Bus
3	Mon-Thurs	NT	Kln	Bus
4	Fri-Sun	HK	HK	MTR
5	Mon-Thurs	Kln	Kln	MTR
6	Mon-Thurs	NT	NT	Bus
7	Mon-Thurs	HK	HK	Bus
8	Fri-Sun	Kln	Kln	MTR
9	Mon-Thurs	Kln	Kln	Bus
10	Mon-Thurs	HK	HK	Bus
11	Fri-Sun	HK	HK	MTR
12	Mon-Thurs	Kln	Kln	Bus
13	Mon-Thurs	HK	HK	Bus
14	Mon-Thurs	Kln	Kln	MTR
15	Fri-Sun	HK	NT	MTR
16	Mon-Thurs	NT	Kln	Bus
17	Mon-Thurs	HK	NT	Bus
18	Mon-Thurs	Kln	HK	MTR
19	Fri-Sun	HK	NT	MTR
20	Mon-Thurs	HK	HK	MTR

Octopus Card Database for Testing

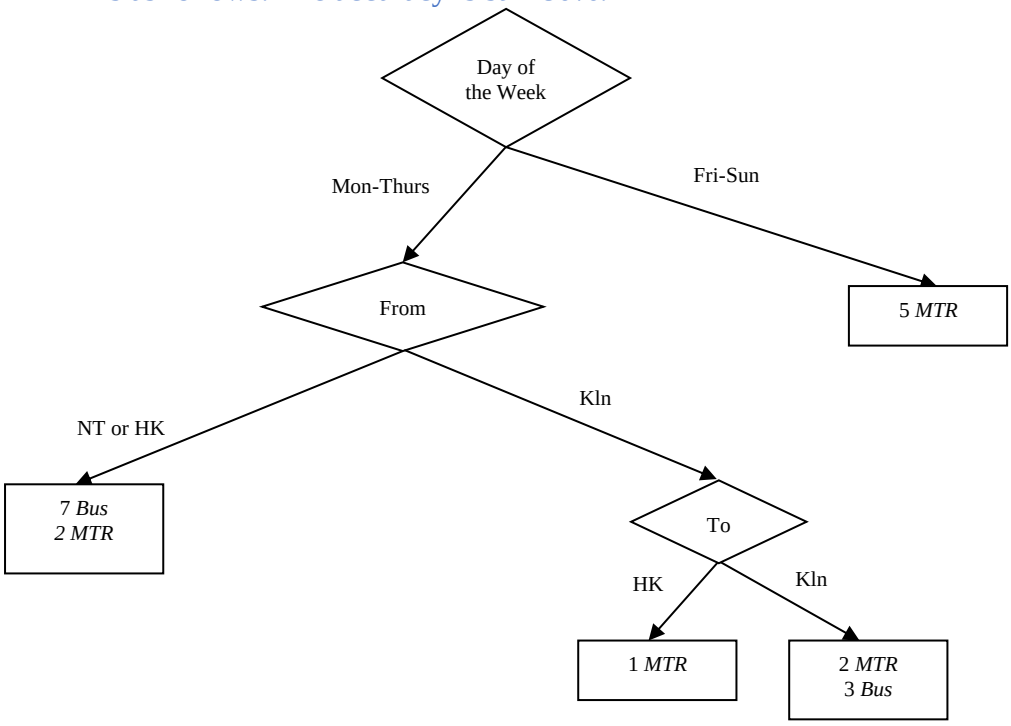
Octopus ID	*Day of the Week*	*From*	*To*	*Transportation Taken*
21	Mon-Thurs	Kln	Kln	Bus
22	Mon-Thurs	HK	HK	MTR
23	Mon-Thurs	Kln	Kln	Bus
24	Mon-Thurs	Kln	Kln	Bus
25	Fri-Sun	Kln	HK	MTR

Your project team member tried to construct a decision tree for the first 20 records above, i.e., the training dataset, and obtained the decision tree.

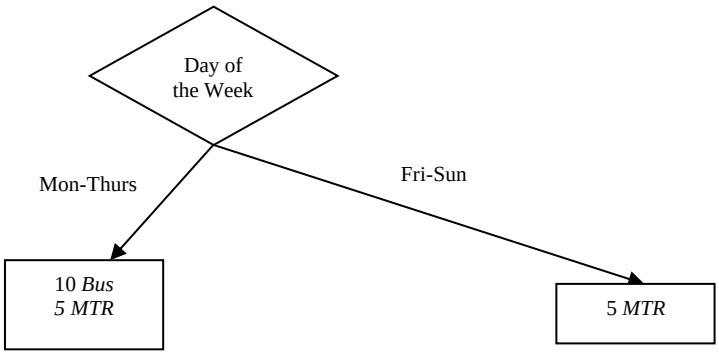


a) A project team member commented that the decision tree above might be overfitted. Prune the tree as much as possible so that the classification rate (accuracy) of the pruned tree for the testing database can be maintained, i.e., as high as that obtained by the given fully grown tree.

We can combine the data points that have *From* attribute of NT and HK. The pruned tree is as follows. The accuracy is still 80%.



To be more aggressive, only considering the performance in the test set, we can even prune the tree like the following. The accuracy in the test set is still 80%. However, there is a potential risk of underfitting by doing so.



b) If some newly available database records are available and added to the training database above, describe the situations for
   i) reconstruction of the decision tree is NOT necessary,
   ii) partial reconstruction of the decision tree is necessary,
   iii) full reconstruction of the decision tree is necessary.
   Give appropriate example(s), e.g., added records, to support your arguments.

   i) If the new records will neither create new leaves or change the tree structure, the reconstruction is not necessary. For example, the following record is the same as one of the existing training data so we do not need to reconstruct the current tree.

Fri-Sun	HK	HK	MTR

   ii) If the new records slightly different from the existing rules and only part of the subtree will be changed, then we only need to reconstruct part of the tree. For example, if we add the following record, the left most leaf in the above tree can be further split. In this way, only part of the tree is reconstructed.

Mon-Thurs	NT	HK	MTR

   iii) If the new records will influence the **selection of attribute to split the data**, then the whole tree may need to be reconstructed. In our case, the first attribute used to split the data is "*Day of the Week"*. Suppose that we are using the information gain as our measure, if some new records are added so that the "*Day of the Week"* no longer has the largest information gain, we then need to reconstruct the tree. For example, we have many new data that shows strong relationship between "From" NT and taking "Bus".

   Another case is that we add some unseen data. For example, we have a new transportation method, say taxi. The previous rule does not include Taxi so the whole tree needs to be reconstructed.

Mon-Thurs	NT	HK	Taxi

c) Based on the decision tree above, think and propose a conceptually sound way to classify the following unseen data and show the results (cf. Transportation Taken)

Octopus ID	*Day of the Week*	*From*	*To*	*Transportation Taken*
51	Mon-Thurs	Unknown	HK	?
52	Unknown	NT	HK	?
53	Mon-Thurs	Kln	Unknown	?

   Method:
   In this case, there is an Unknown value in each record, however, we can still predict the Transportation Taken by adopt the Maximum Likelihood Estimation strategy. We will calculate the probability of each possible value of the Unknown data. We divide the features into two parts:

   For *Day of the week*, we will simply use the frequency divided by total number as the probability.

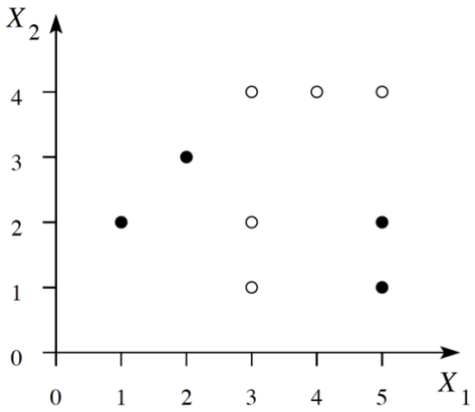
   For *From* and *To*, we will calculate it based on the given value. For example, in ID 51, we will calculate the probability of Unknown given *To* is HK.

For ID 51:
From the training data, we can calculate the distribution of *"From"* feature given that the *To* feature is HK:
P(NT|HK): 0. If it is NT, according to the DT, it will be predicted as bus.
P(HK|HK): 0.875. If it is HK, according to the DT, it will be predicted as bus.
P(Kln|HK): 0.125. If it is Kln, according to the DT, it will be predicted as MTR.
P(bus) = 0.875 > P(MTR) = 0.125
Therefore, it will be predicted as Bus.

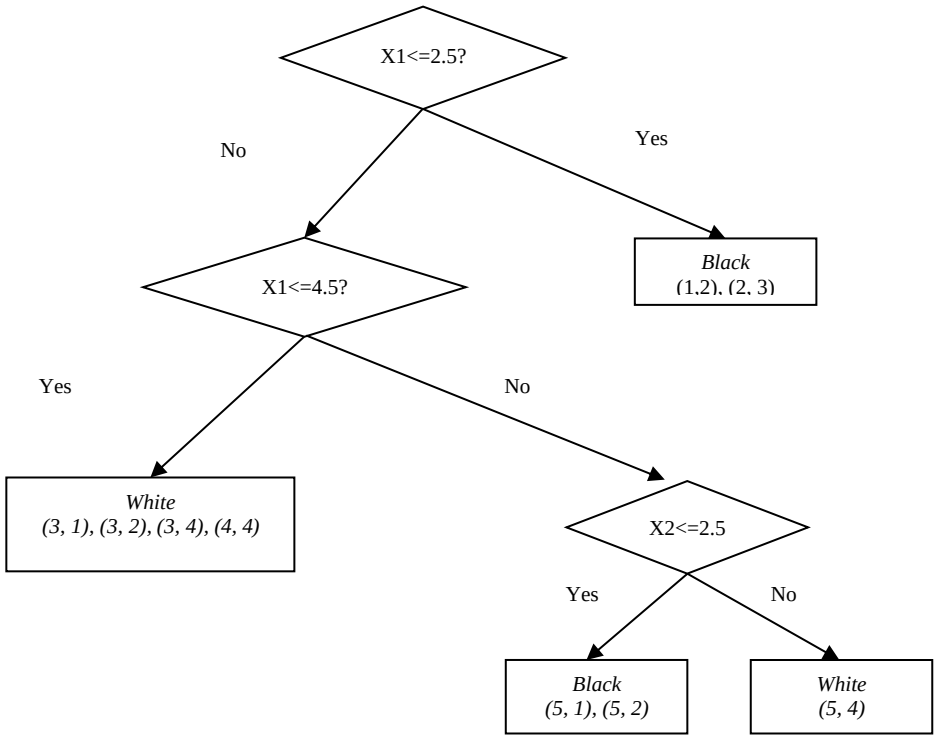
For ID 52:
From the training data, we can calculate the distribution of *"Day of the Week"* feature:
Probability of Mon-Thurs: 0.75. If it is Mon-Thurs, according to the DT, it will be predicted as bus.
Probability of Fri-Sun: 0.25. If it is Fri-Sun, according to the DT, it will be predicted as MTR.
P(bus) = 0.75 > P(MTR) = 0.25
Therefore, it will be predicted as Bus.

For ID 53:
From the training data, we can calculate the distribution of *"To"* feature given that the *From* feature is Kln:
Probability of NT: 0.
Probability of HK: 1/7. If it is HK, according to the DT, it will be predicted as MTR.
Probability of Kln: 6/7. If it is Kln, according to the DT, it will be predicted as Bus.
P(bus) = 6/7 > P(MTR) = 1/7
Therefore, it will be predicted as Bus

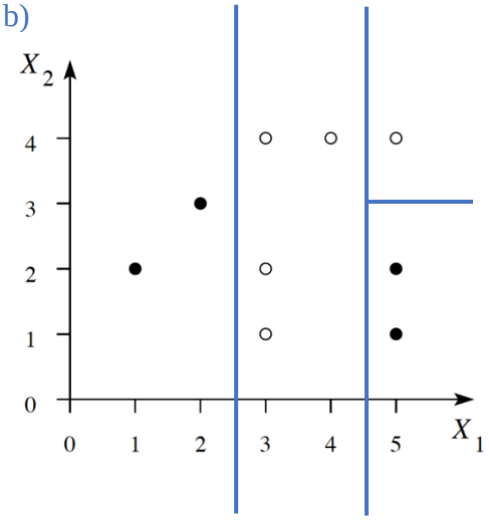
2. For the following dataset with 4 black dots and 5 white dots (data points), with integer aligned coordinate values, construct a decision tree to attain 100% accuracy on the 9 points.
   a) Draw the tree and label all the nodes appropriately. Note that you are NOT required to show the steps.
   b) Draw the corresponding decision boundary of the tree you provided in part (a).
   c) For the root node of your constructed tree in part (a), show the impurity after split (cf. slides 30 and 31 of the decision tree lecture notes)



(a)



b)



c)

Impurity = -2/9 *(1*log(1) + 0*log(0)) - 7/9* (5/7*(log5/7) + 2/7*(log(2/7)))
= 0.6713

3. The model usage time of *k*-nearest neighbor method is sensitive to the dataset size (i.e. amount of training/stored data). When the dataset is large, the model usage time could be prohibitively long. Suggest TWO reasonable ways to reduce the required large amount of similarity computation and comment on their effectiveness.

1. KD tree: A K-D Tree (also called as K-Dimensional Tree) is a binary search tree where data in each node is a K-Dimensional point in space. It can quickly exclude data of little relevance to the query by building indexes. Because the actual data generally shows a cluster-like clustering pattern, the speed of retrieval can be greatly accelerated by designing an effective **index structure**. The pseudocode is as follows:

The time complexity of building the kd-tree is O(k*n*logn) for k-dimensional data with n instances.

PROCEDURE *createKDTNode*(S_Q, k)
INPUTS:
- A set of training samples S_Q.
- An integer $k \in \{0, 1 \ldots, d\text{-}1\}$, where the $(k+1)^{th}$ input attribute is used to partition S_Q.

OUTPUT: A new node Q of a k-D tree
1. If $|S_Q| < 2^{d+1}$, go to step 7.
2. Set $\boldsymbol{y}_Q$ and $\boldsymbol{z}_Q$ using the set S_Q.
3. Set $T_{Q,L} = T_{Q,R} = \phi$.
4. For each $\boldsymbol{x}_i = (x_{i,1}, x_{i,2} \ldots, x_{i,d}) \in S_Q$
   - If $x_{i,k+1} < \beta$, set $T_{Q,L} = T_{Q,L} + \{\boldsymbol{x}_i\}$, where $\beta = \sum_{x_i \in S_Q} x_{i,k+1} / |S_Q|$
   - Otherwise, set $T_{Q,R} = T_{Q,R} + \{\boldsymbol{x}_i\}$.
5. Invoke *createKDTNode*($T_{Q,L}$, $(k+1)$ mod d) and *createKDTNode*($T_{Q,R}$, $(k+1)$ mod d)
6. Return the node Q and skip step 7.
7. Label Q as a leaf node and create an array of pointers to the training samples in S_Q.

Reference: Bentley J L. Multidimensional binary search trees used for associative searching[J]. Communications of the ACM, 1975, 18(9): 509-517.

2. Locality Sensitivity Hashing (LSH): Unlike the tree-structured approach that gives precise results, LSH gives an approximation because in many fields very high precision is

not required. Especially when there is a huge amount of data, kd-tree sometimes does not work well.

The LSH method is based on the idea that two points that are very close (similar) in the original space have a high probability that their hashes are the same after the mapping of the LSH hash function, while two points that are very far away (not similar) have a small probability that their hashes are equal after the mapping. Meanwhile, we know that in hash table, the search time complexity is O(1), therefore we can improve the KNN algorithm.

The pseudocode is as follows:

Function: basic_ann_by_lsh(X, k, m, block-sz)
begin
  $Y = LSH(X, m)$
  Project Y onto a random direction w, $p = Yw$
  Sort items by p values, and get $\{x_{\pi_1}, x_{\pi_2}, \ldots, x_{\pi_n}\}$
  for $i = 1, \ldots, n/block\text{-}sz$ **do**
    $S_i = \{x_{\pi_{(i-1)\times block-sz+1}}, \cdots, x_{\pi_{i\times block-sz}}\}$ g_i = brute_force_kNN(S_i, k)
  return: $G = \bigcup\{g_i\}$

Algorithm 1: Basic kNN Graph Construction with LSH

Algorithm: Approximate kNN Graph Construction with LSH
Input: X, k, l, m, block-sz
begin
  for $i = 1, \ldots, l$ **do**
    G_i = basic_ann_by_lsh(X, k, m, block-sz)
  Combine $\{G_1, G_2, \ldots, G_l\}$ to get G
  Refine G by one step neighbor propagation
  return: G

Algorithm 2: Main Algorithm

Reference: https://en.wikipedia.org/wiki/Locality-sensitive_hashing

Zhang Y M, Huang K, Geng G, et al. Fast kNN graph construction with locality sensitive hashing[C]//Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, Berlin, Heidelberg, 2013: 660-674.