Assignment 3
Data Mining

1) 1.1 Supermarket dataset top 4 association rules with Apriori (default parameters)

1. biscuits=t frozen foods=t fruit=t total=high 788 ==> bread and cake=t 723 <conf:(0.92)> lift:(1.27) lev:(0.03) [155] conv:(3.35)
2. baking needs=t biscuits=t fruit=t total=high 760 ==> bread and cake=t 696 <conf:(0.92)> lift:(1.27) lev:(0.03) [149] conv:(3.28)
3. baking needs=t frozen foods=t fruit=t total=high 770 ==> bread and cake=t 705 <conf:(0.92)> lift:(1.27) lev:(0.03) [150] conv:(3.27)
4. biscuits=t fruit=t vegetables=t total=high 815 ==> bread and cake=t 746 <conf:(0.92)> lift:(1.27) lev:(0.03) [159] conv:(3.26)

Customer who buy biscuits, frozen foods, fruit also tend to buy bread and cake.
Customer who buy baking needs, biscuits, fruit also tend to buy bread and cake.
Customer who buy baking needs, frozen foods, fruits also tend to buy bread and cake.
Customer who buy Biscuits, fruit, vegetables also tend to buy bread and cake.

Main parameters weka allows us to change are:

- upperBoundMinSupport -- Upper bound for minimum support. This value must be chosen correctly for the algorithm as the algorithm starts from the upper bound and goes down till the lowerBoundMinSupport. The value for this parameter should be high as we need to consider itemsets with higher support as they more repetitive in the transaction list indicating more chances of occurrence. The delta value(0.05) is subtracted from the upperBoundMinSupport in each iteration.
- LowerBoundMinSupport -- Lower bound for minimum support. This is the least value of support the algorithm will go up to. If the value of this parameter is too high, then the algorithm might not find any association rules. This parameter value depends based on your dataset and how many rules who want to find.
- delta -- Iteratively decrease upperBoundMinSupport by this factor. Reduces support until min support is reached or required number of rules has been generated.

Support is expressed as a proportion of no of instances. Starts at upperBoundMinSupport decreases by delta at each interval (default 5%) stops when numRules reached or at lowerBoundMinSupport (default 10%)

- minMetric -- Minimum metric score. (Default 90%) Sets the minimum metricType. Consider only rules with scores higher than this value. If the value is too high not many rules will be found and vice-versa. We use confidence for this parameter.
- metricType -- Set the type of metric by which to rank rules. We can choose the metric based on which our algorithm must find the rules for us. In our case we are using confidence. The 4 available options are Confidence, Lift, Leverage, Conviction. We choose confidence in our algorithm.
- numRules -- Number of rules to find. This parameter gives the number of rules apriori algorithm must find. This parameter also acts as the stopping criteria for the algorithm as it stops if it meets the lowerBoundMinSupport or the numRules.
- outputItemSets -- If enabled the itemsets are printed on the screen.

Low support high confidence (Default):

1. biscuits=t frozen foods=t fruit=t total=high 788 ==> bread and cake=t 723 <conf:(0.92)>
2. baking needs=t biscuits=t fruit=t total=high 760 ==> bread and cake=t 696 <conf:(0.92)>

The default parameters choose all itemsets that are not that frequent but the probability that the combination go together is high. During this the algorithm stops even before reaching the lowest support as the numRules is reached.

Low support and confidence:

1. biscuits=t frozen foods=t party snack foods=t fruit=t vegetables=t total=high 510 ==> bread and cake=t 478 <conf:(0.94)>
2. biscuits=t frozen foods=t cheese=t fruit=t total=high 495 ==> bread and cake=t 463

We find more number of rules when both the support and confidence are low as we find rules for not frequent itemsets in the dataset and for those which has a low value for how often the rule is true. In weka you need to change the number of rules or else the apriori algorithm stops when the number of rules is reached even before reaching the minimum support leading to different set of rules. The rules change as we consider low support we find some of the items that are not that frequent, but the rule holds good most of the time(confidence).

Medium support and confidence:

1. milk-cream=t 2939 ==> bread and cake=t 2337 <conf:(0.8)>
2. fruit=t 2962 ==> bread and cake=t 2325 <conf:(0.78)>

Weka finds 4 rules with a support and confidence 0.5. In this case the rules change again. Here we find itemsets that are almost 50% as frequent and have a decent confidence leading to lesser number of rules generated. We are interested in looking at itemsets not only with high confidence but also is frequent in the datasets. So the rules found are different and do not have very high confidence on the rules as in the previous cases.

High support and confidence: Modifying support and confidence is 0.9(High). The algorithm has no large itemsets and no rules are found! Here in this case both the support and the confidence must be higher than 0.9 hence even if we have many rules with very high confidence there are no rules with support greater than the 0.9 leading to no rules being found using this dataset. Any rules found using these parameters are very common and the two items are almost always bought together.

	Default	low	medium	high
Support	0.15	0.1	0.5	0.9
Confidence	0.9	0.1	0.5	0.9
No. of rules	10	183372	4	0

Conclusion: Generation of rules slowly decreased as the support and confidence is increased. The size of the largest itemset also decreases with the increase in both the parameter. The rules do not remain the same with the change in the parameters to satisfy the conditions. The way we choose depends on the type of application we are using this algorithm. If we choose rules with high support as these are the itemsets are most frequently occurring and a high confidence shows that the chances of the rule being True is high.

1.2

Removing bread and cake and fruit as it appears in most of the rules, we find our new set of rules. We can figure out bread and cake appear in most rules using the support count.

default values: (support 0.1 and confidence 0.9) There are some itemsets found as the support is low but the confidence has a very high which is why we don't see any rules when we apply default parameters.

For low- different rules that do not contain bread and cake as the antecedent or consequent. (same as before to get all the rule u need to have a big number for numRules, as stopping condition is met by numrules.) We now see that other frequent items have even lesser confidence then the previous rules.

```
1. biscuits=t prepared meals=t total=high 539 ==> frozen foods=t 474    <conf:(0.88)>
2. baking needs=t biscuits=t beef=t total=high 534 ==> frozen foods=t 467    <conf:(0.87)>
```

For Medium - No large itemsets and rules found! As we have removed the item with most frequent in the rule(high support like bread and cake and fruit), there are no itemset which has confidence greater than the 0.5 value.Hence we do not find any rules with these parameters.

For high- No large itemsets and rules found! Same as above.

Conclusion :The number of rules formed drastically reduces as we remove these main itemsets, as these are ones with high support.

	Default	low	medium	high
Support	0.1	0.1	0.5	0.9
Confidence	0.9	0.1	0.5	0.9
No. of rules	10	53854	0	0
No. of cycles performed	18	18	-	-

1.3

Output for Vote Dataset with default parameters. Only the first 5 rules are shown below.

```
1. adoption-of-the-budget-resolution=y physician-fee-freeze=n 219 ==> Class=democrat 219    <conf:(1)>
2. adoption-of-the-budget-resolution=y physician-fee-freeze=n aid-to-nicaraguan-contras=y 198 ==> Class=democrat 198    <conf:(1)>
3. physician-fee-freeze=n aid-to-nicaraguan-contras=y 211 ==> Class=democrat 210    <conf:(1)>
4. physician-fee-freeze=n education-spending=n 202 ==> Class=democrat 201    <conf:(1)>
5. physician-fee-freeze=n 247 ==> Class=democrat 245    <conf:(0.99)>
```

The first rule defines attributes adoption-of-the-budget-resolution(n) and physician-fee-freeze(y) tend to vote democrats, our decision tree classifier also selects physician-fee-freeze and then adoption-of-the-budget-resolution as the main feature selection attribute. The right-hand side of all the top rules contains democrat. As we had seen earlier assignment classifying to democrat was comparatively easy with the use of just physician-fee-freeze.

5th Rule was one the conclusion we came to while visually interpreting the graph of decision tree.

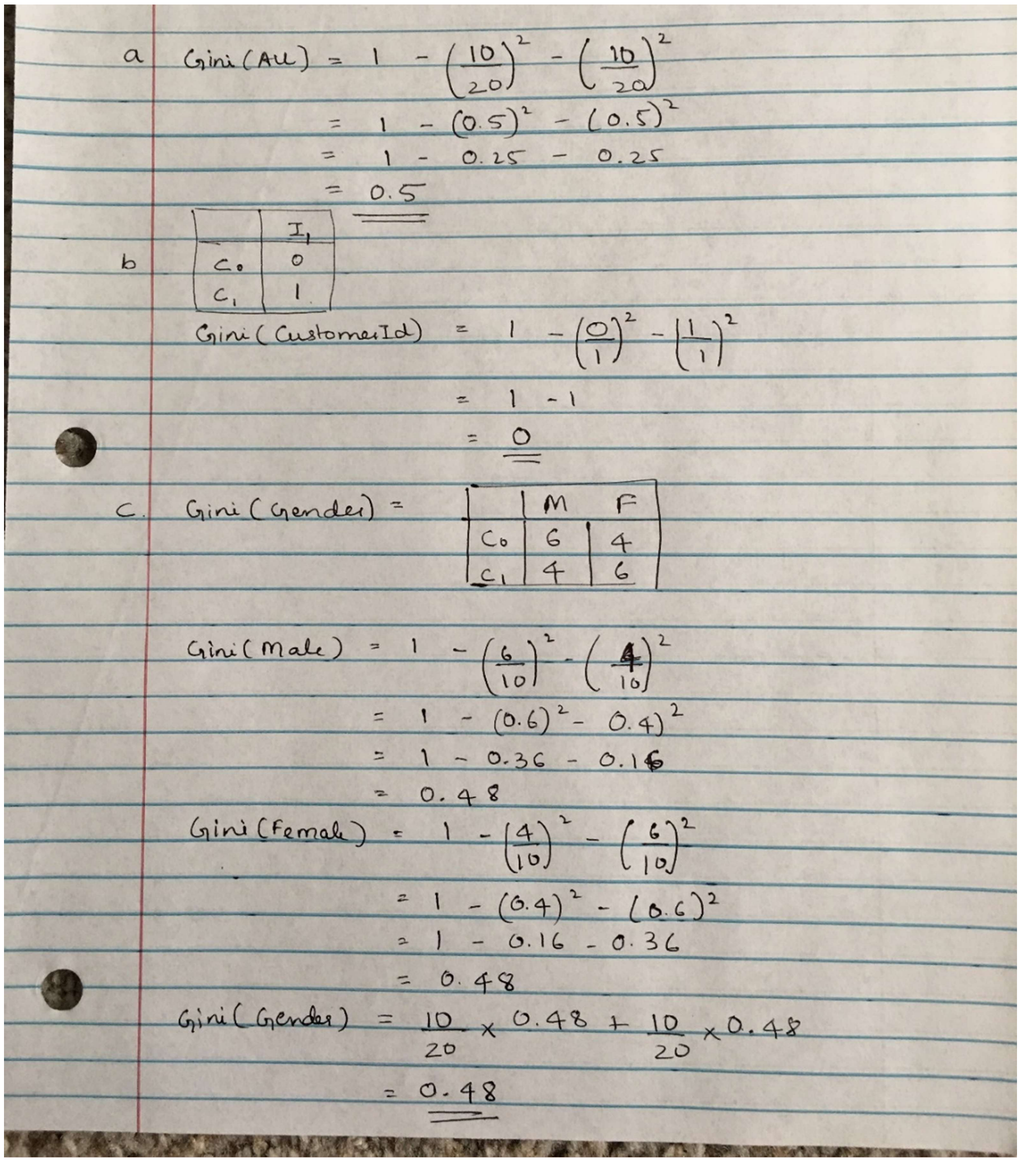
physician-fee-freeze is one attribute which classified majority of the democrats. It has a pure node which can classify all the data-instances correctly. It also classifies Republicans correctly majority of the time but has a few instances of the Democrats which can be further classified considering the next best features. Conclusion: The most important features selected using feature selection match with the rules formed using the apriori algorithm. The RHS mostly consists of Democrat in the top rule but not Republicans as the Support for Democrats is higher compared to Republicans this is because the dataset contains more democrats than Republicans.

2

2.1

GINI(t) = 1 - $\sum_i p(j|t)]^2$

a. Fraction of instances in class 0 (10/20), Fraction of instances in class 1 (10/20) out of 20 instances.



a $\text{Gini(All)} = 1 - \left(\frac{10}{20}\right)^2 - \left(\frac{10}{20}\right)^2$

$= 1 - (0.5)^2 - (0.5)^2$

$= 1 - 0.25 - 0.25$

$= \underline{\underline{0.5}}$

b

	I_1
C_0	0
C_1	1

$\text{Gini(CustomerId)} = 1 - \left(\frac{0}{1}\right)^2 - \left(\frac{1}{1}\right)^2$

$= 1 - 1$

$= \underline{\underline{0}}$

c. Gini(Gender) =

	M	F
C_0	6	4
C_1	4	6

$\text{Gini(Male)} = 1 - \left(\frac{6}{10}\right)^2 - \left(\frac{4}{10}\right)^2$

$= 1 - (0.6)^2 - 0.4)^2$

$= 1 - 0.36 - 0.16$

$= 0.48$

$\text{Gini(Female)} = 1 - \left(\frac{4}{10}\right)^2 - \left(\frac{6}{10}\right)^2$

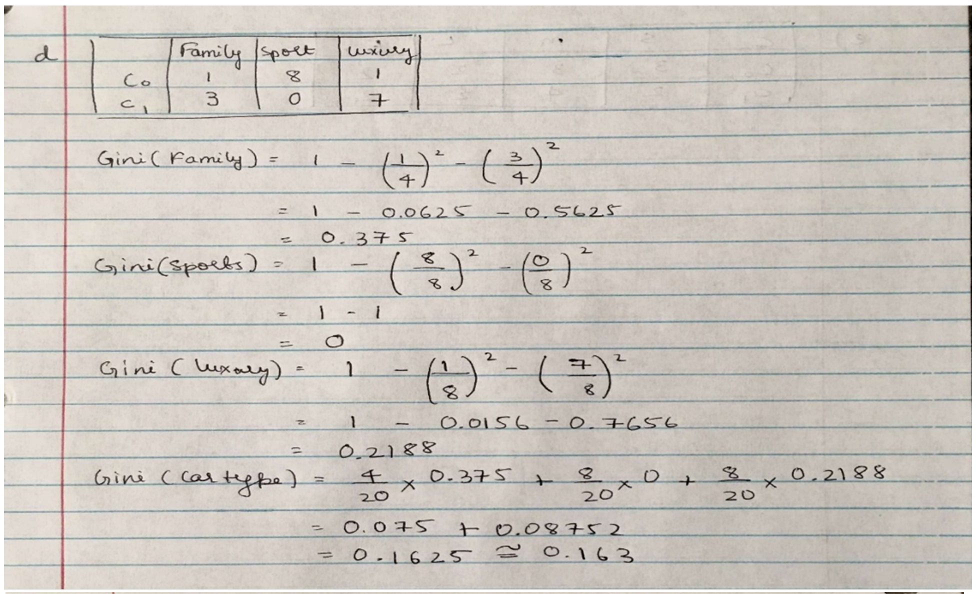
$= 1 - (0.4)^2 - (0.6)^2$

$= 1 - 0.16 - 0.36$

$= 0.48$

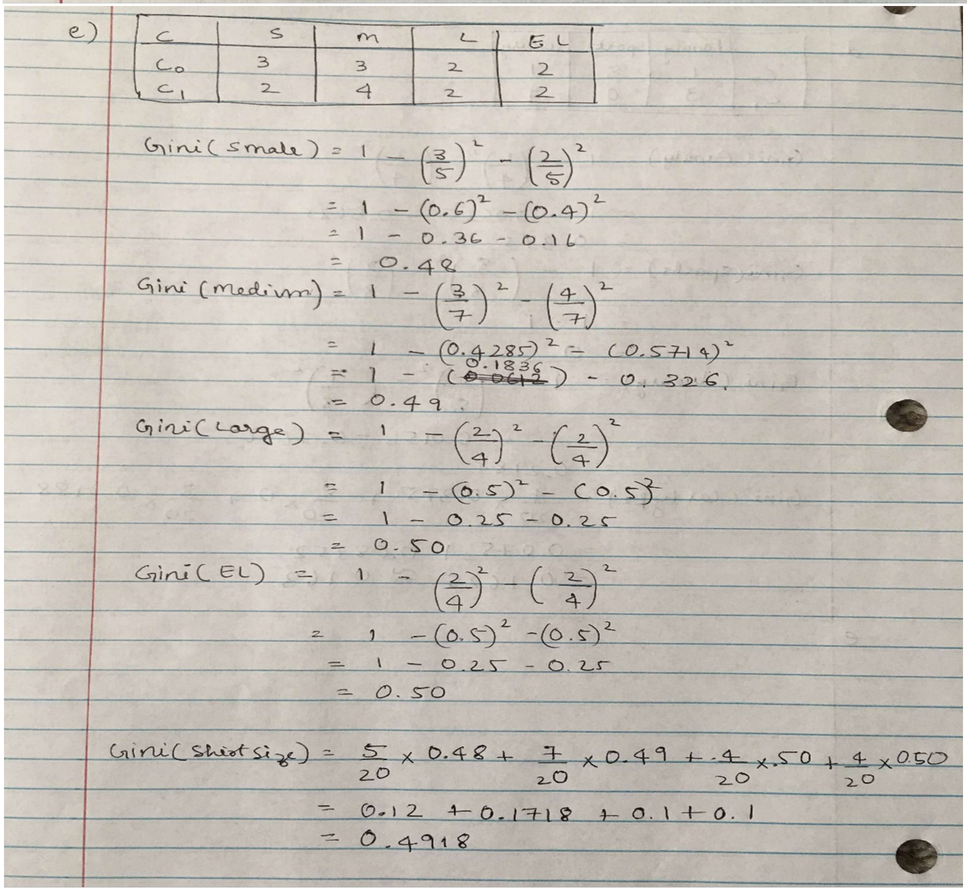
$\text{Gini(Gender)} = \frac{10}{20} \times 0.48 + \frac{10}{20} \times 0.48$

$= \underline{\underline{0.48}}$


d

	Family	Sport	Luxury
C_0	1	8	1
C_1	3	0	7

$$\text{Gini(Family)} = 1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2$$
$$= 1 - 0.0625 - 0.5625$$
$$= 0.375$$
$$\text{Gini(sports)} = 1 - \left(\frac{8}{8}\right)^2 - \left(\frac{0}{8}\right)^2$$
$$= 1 - 1$$
$$= 0$$
$$\text{Gini(luxury)} = 1 - \left(\frac{1}{8}\right)^2 - \left(\frac{7}{8}\right)^2$$
$$= 1 - 0.0156 - 0.7656$$
$$= 0.2188$$
$$\text{Gini(Car type)} = \frac{4}{20} \times 0.375 + \frac{8}{20} \times 0 + \frac{8}{20} \times 0.2188$$
$$= 0.075 + 0.08752$$
$$= 0.1625 \cong 0.163$$


e)

C	S	m	L	EL
C_0	3	3	2	2
C_1	2	4	2	2

$$\text{Gini(small)} = 1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2$$
$$= 1 - (0.6)^2 - (0.4)^2$$
$$= 1 - 0.36 - 0.16$$
$$= 0.48$$
$$\text{Gini(medium)} = 1 - \left(\frac{3}{7}\right)^2 - \left(\frac{4}{7}\right)^2$$
$$= 1 - (0.4285)^2 - (0.5714)^2$$
$$= 1 - (\cancel{0.0612}\ 0.1836) - 0.326.$$
$$= 0.49$$
$$\text{Gini(Large)} = 1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2$$
$$= 1 - (0.5)^2 - (0.5)^2$$
$$= 1 - 0.25 - 0.25$$
$$= 0.50$$
$$\text{Gini(EL)} = 1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2$$
$$= 1 - (0.5)^2 - (0.5)^2$$
$$= 1 - 0.25 - 0.25$$
$$= 0.50$$

$$\text{Gini(Shirt Size)} = \frac{5}{20} \times 0.48 + \frac{7}{20} \times 0.49 + \frac{4}{20} \times .50 + \frac{4}{20} \times 0.50$$
$$= 0.12 + 0.1718 + 0.1 + 0.1$$
$$= 0.4918$$

f)

1. Fraction of instances in class 0 (10/20), Fraction of instances in class 1 (10/20) out of 20 instances with a gini of 0.5 on the root node. To make the first split I would choose the attribute Car type as the gini index is minimum, implying most of records belong to one class after the split based on this attribute (pure node) and by visual interpretation as explained in point 4.
2. Based on the table the customer ID has a gini index of 0 even when taking the weighted average of all the customers, but this is not a useful attribute as it is used for unique identification of the data, based on this

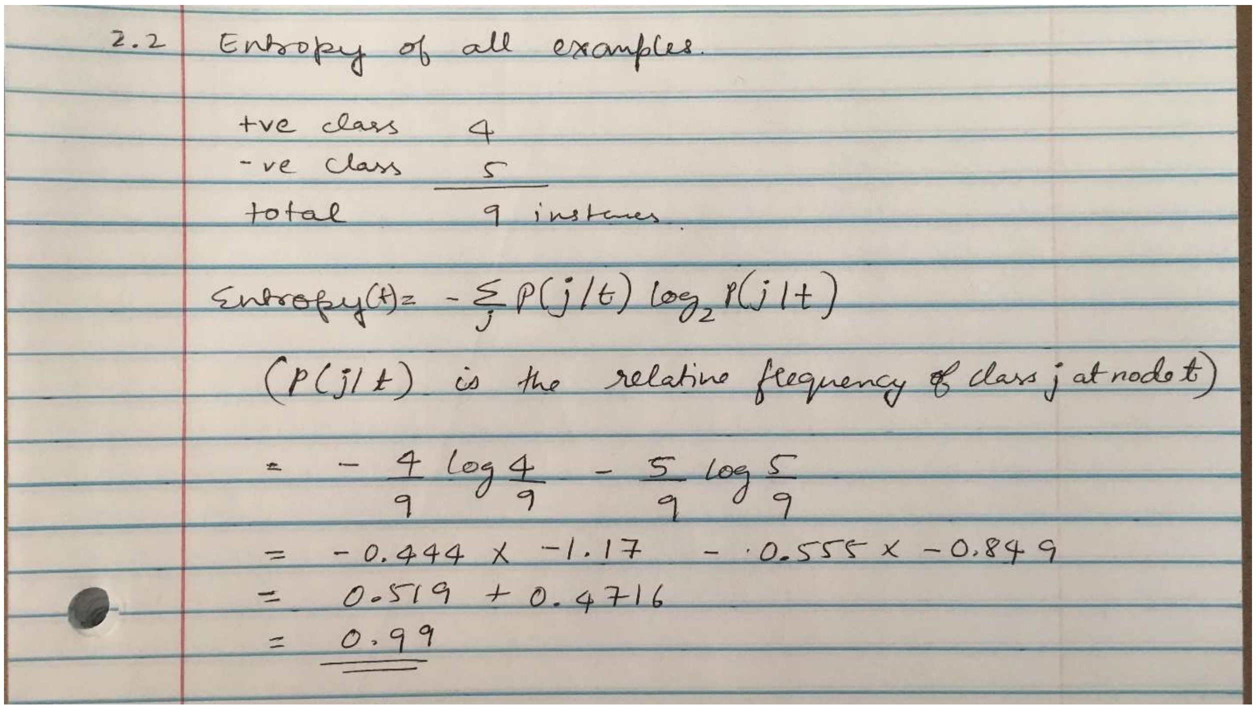
attribute no new data can be correctly classified. Hence customer ID is not the correct attribute to make our decision to split even though it has the lowest Gini. This is a very good example for overfitting.

3. Classifying based on the Gender, we cannot come to any conclusion as the number of instances in both the class are almost equally distributed (6:4 class0 4:6 class1) which is also implied by the gini index with a value of 0.48 where 0 being the purest nodes and 0.5 meaning the most impure node, hence from this node we do not gain much information based on splitting with this attribute.
4. Classifying based on car-type is one reasonable consideration as most of the sports car belong to class 0 and family and luxury car belong to class 1. The gini index for this attribute is 0.161 which is almost close to 0 (pure node) implying a better attribute make a split.
5. Classification based on the shirt size is not a good choice as all the values of the attributes are almost equally distributes among the class letting the classifier not being a good judge on which class label to apply. This has a gini index of 0.4918 making it an impure node giving a 50% chance for both classes.

Conclusion: Gini with the smallest value shows the purest node which is the best attribute to make a split on. Here in this example we have customer id with gini 0 but we do not consider this as it doesn't generalize well leading to overfitting (as the id is different for different customer) next comes the car-type, which has the lowest gini of 0.163 and which would be our choice as explained why in the 4th point.

2.2

a)



2.2 Entropy of all examples.

+ve class 4
-ve class 5
total 9 instances.

Entropy(t) = $-\sum_j P(j/t) \log_2 P(j|t)$

($P(j/t)$ is the relative frequency of class j at node t)

$= -\frac{4}{9}\log\frac{4}{9} - \frac{5}{9}\log\frac{5}{9}$

$= -0.444 \times -1.17 - 0.555 \times -0.849$

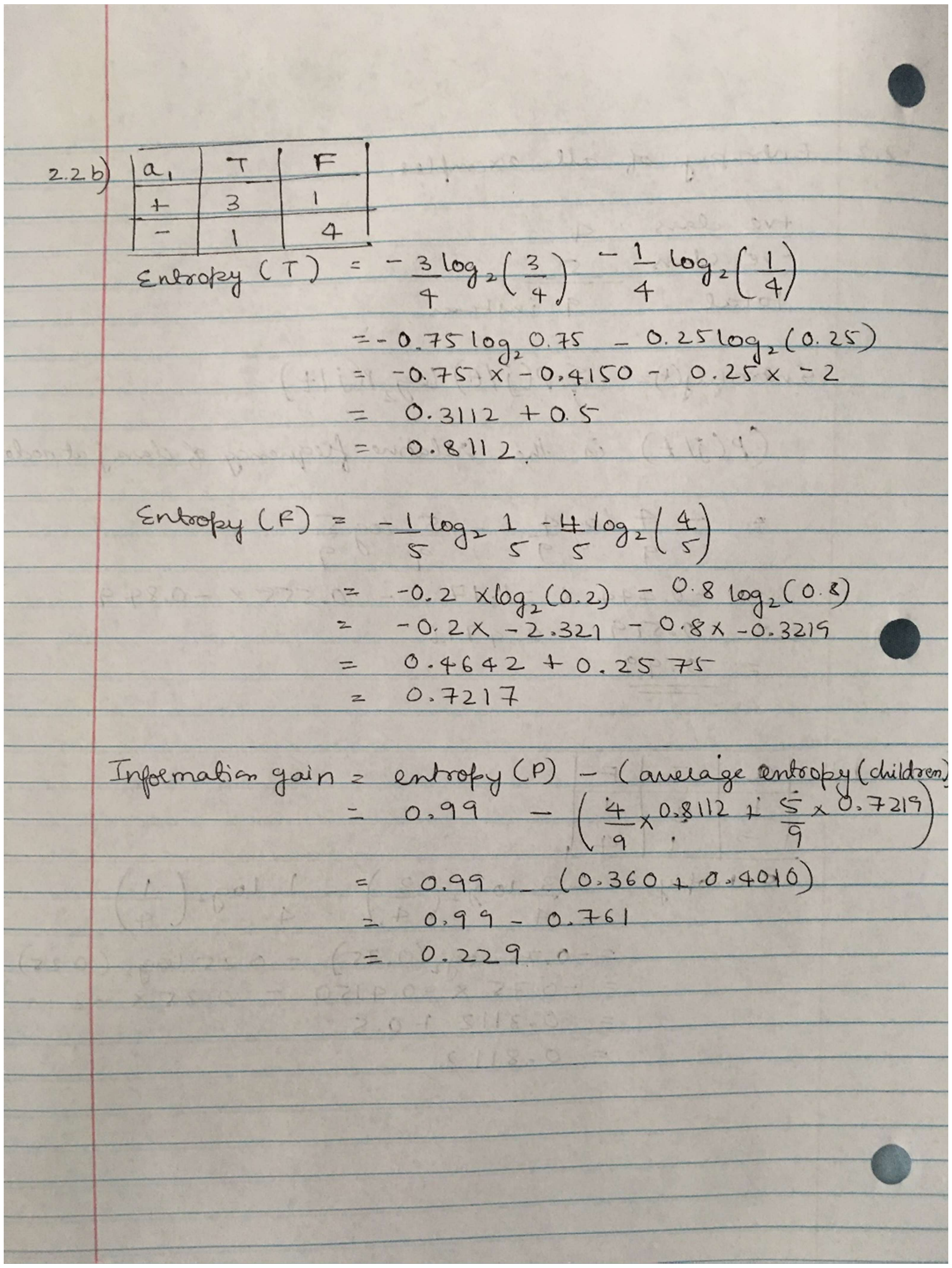
$= 0.519 + 0.4716$

$= \underline{0.99}$

The value is very Large. It means that the data has a lot of information and it is hard to predict. Higher the entropy the more unpredictable the event being measured is. It means that the class distribution is almost equally split between the two of the classes + and - , visual analysis also shows that the class + has 4 instances and class – has 5 instances making it a non-homogenous node. As told in the class for such cases the element of surprise for this is very high. The classifier has not much information on how to group them into homogenous node hence have a high degree of impurity. The probability for classifier to classify a instance to class is 50:50, the entropy reflects how expected is the class label which for this case is very less implying high element of surprise.

---

b)



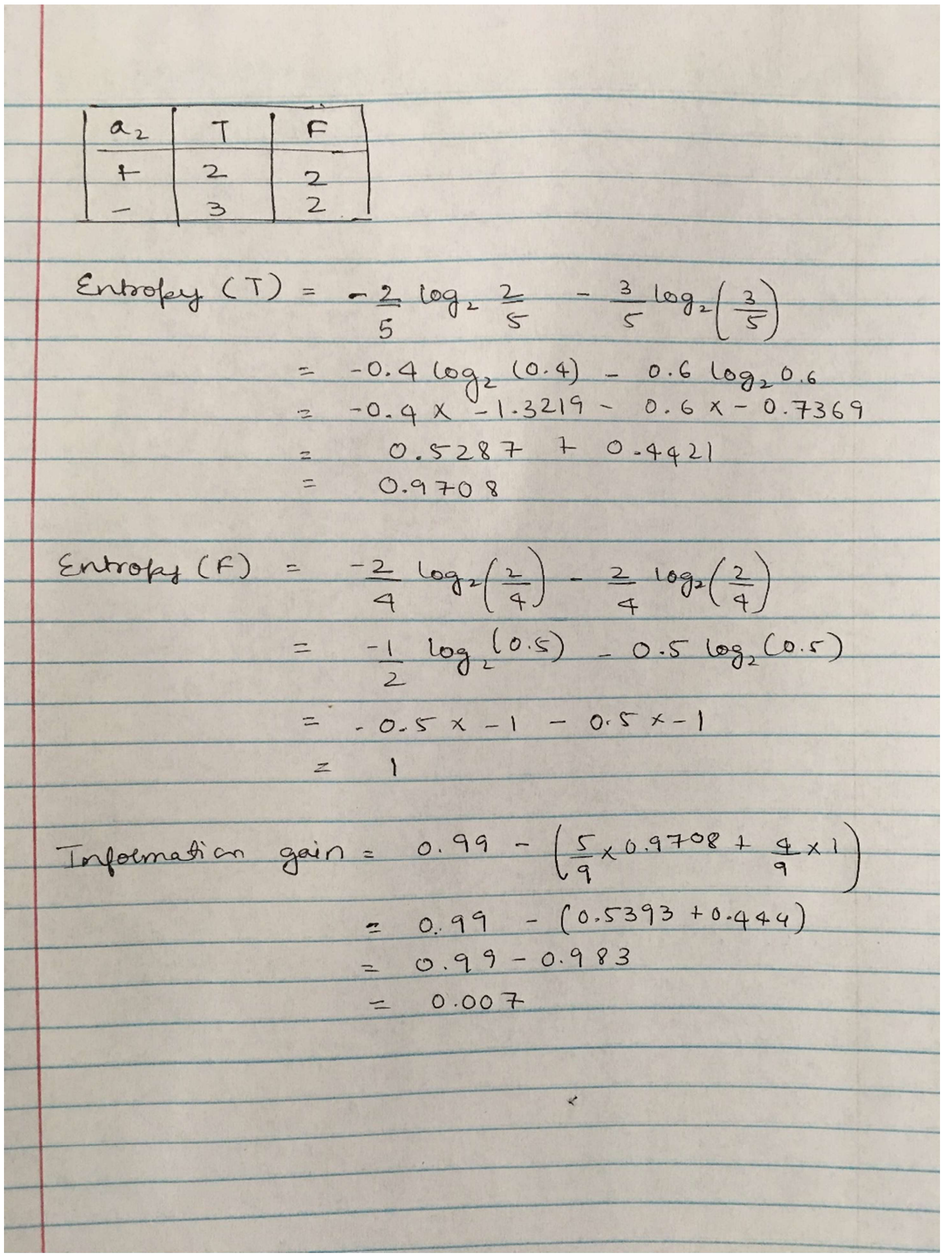
2.2 b)

a_1	T	F
+	3	1
−	1	4

$$\text{Entropy}(T) = -\frac{3}{4}\log_2\left(\frac{3}{4}\right) - \frac{1}{4}\log_2\left(\frac{1}{4}\right)$$
$$= -0.75\log_2 0.75 - 0.25\log_2(0.25)$$
$$= -0.75 \times -0.4150 - 0.25 \times -2$$
$$= 0.3112 + 0.5$$
$$= 0.8112.$$

$$\text{Entropy}(F) = -\frac{1}{5}\log_2\frac{1}{5} - \frac{4}{5}\log_2\left(\frac{4}{5}\right)$$
$$= -0.2 \times \log_2(0.2) - 0.8\log_2(0.8)$$
$$= -0.2 \times -2.321 - 0.8 \times -0.3219$$
$$= 0.4642 + 0.2575$$
$$= 0.7217$$

Information gain = entropy (P) − (average entropy (children)
$$= 0.99 - \left(\frac{4}{9} \times 0.8112 + \frac{5}{9} \times 0.7219\right)$$
$$= 0.99 - (0.360 + 0.4010)$$
$$= 0.99 - 0.761$$
$$= 0.229$$



a_2	T	F
+	2	2
−	3	2

$$\text{Entropy}(T) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\left(\frac{3}{5}\right)$$
$$= -0.4\log_2(0.4) - 0.6\log_2 0.6$$
$$= -0.4 \times -1.3219 - 0.6 \times -0.7369$$
$$= 0.5287 + 0.4421$$
$$= 0.9708$$

$$\text{Entropy}(F) = -\frac{2}{4}\log_2\left(\frac{2}{4}\right) - \frac{2}{4}\log_2\left(\frac{2}{4}\right)$$
$$= -\frac{1}{2}\log_2(0.5) - 0.5\log_2(0.5)$$
$$= -0.5 \times -1 - 0.5 \times -1$$
$$= 1$$

$$\text{Information gain} = 0.99 - \left(\frac{5}{9}\times 0.9708 + \frac{4}{9}\times 1\right)$$
$$= 0.99 - (0.5393 + 0.444)$$
$$= 0.99 - 0.983$$
$$= 0.007$$

The table a1 and a2 is constructed at the beginning of the previous 2 images.

A1	T	F
+	3	1
-	1	4

A2	T	F
+	2	2
-	3	2

Decision tree uses entropy to calculate the homogeneity of a node. If the sample is completely homogeneous the entropy is zero and if the sample is non-homogeneous (class labels are equally distributed) entropy is one. The a1 and a2 table are used to get the frequency or the probability of occurrence of a specific value for each of the class. Explanation of how the table is being used – In table a1 we write the number of instances that are T and belong to + class and that belong to – class. Similarly, we write the number of instances that are F and belong to + class and that belong to – class. Now we use the first column where the attribute value is T to find how many of the instances which are T belong to the +class and how many belong to the -class. This fraction of instance in +class to the total in

T column is used to find the find the entropy. Similarly, the same thing is done for the F column. We then take the weighted sum of these two values to find the total entropy of the node for splitting based on this attribute.

For the calculation of IG, the weighted sum of these 2 entropies is subtracted from the entropy of the parent node.

Equations used:

- Entropy(t) = $-\sum P(j \mid t) \log_2 P(j \mid t)$

  $P(j \mid t)$ is the relative frequency of class j at node t.
  Number of instances belonging to a category to the total number of instances.
  The value must be as low as possible indicating that the node is pure, 0 being the purest.

- $\text{GAIN} = Entropy(p) - \sum_{i=1}^{k} \left(\frac{n_i}{\text{n}}\right) \text{Entropy(i)}$

  $Entropy(p)$: Entropy of parent
  $\sum_{i=1}^{k} \left(\frac{n_i}{\text{n}}\right) \text{Entropy(i)}$: Weighted sum of the entropy of children

  Information gain must be as high indicating that splitting by this attribute, we will have maximized the information for splitting which means we can have a homogenous node.
  If the Weighted sum of the entropy of children is less, we will get high Information gain.
  <u>If the entropy is more information gain is less and vice-versa.</u>

----

c)
The Information gain on a1 is more compared to the a2 hence we should use a1 to make our split.
Information gain gives the value indicating how much better the classifier can do while splitting on an attribute. It gives the difference between the entropy of class distribution before the split and after the split. If the value is more, we are training the classifier to do better compared to the previous node.
When we look at whole dataset the class distribution is almost having equal no of instances (i.e. 4 + and 5 -). When we split on a1 we notice that the most T goes to class + and F goes to class -. Hence new instances can be classified with a small element of surprise. But in case of splitting on a2 we can see that there is 50% chance for T and F for both the classes. The element of surprise is high in this case. Hence this is a good node for classification.

2.3
a)
16 attributes, the cost for each internal node in the decision tree is:
$\text{Log}_2(16) = 4$
3 classes, the cost for each leaf node is:
$\text{Log}_2(3) = 1.58$
The cost for each misclassification error is:
$\log_2(n)$. n is the number of instances.

The overall cost for the decision tree (a) is
<u>$2 \times 4 + 3 \times 1.58 + 7 \times \log_2 n = 12.74 + 7 \log_2 n$</u>

the overall cost for the decision tree (b) is
$4 \times 4 + 5 \times 1.58 + 4 \times \log_2 n = 23.9 + 4 \log_2 n$.

According to the MDL principle, tree (a) is better than (b) As the principle states that Given two models of similar generalization errors, one should prefer the simpler model over the more complex model. Complex models have a greater chance of being fitted accidentally by errors in data. Hence model complexity should be considered while choosing a model. Complex trees are sometime a reason of overfitting which might work well on the training set but does not generalize to new unseen data.

	n=1	n=8	n=16	N=32
a	12.74	33.74	40.74	47.74
b	23.9	35.9	39.9	43.9

According to the MDL, tree a is better than b if n < 16 and is worse than b if n > =16 as seen in the table.

----

b) Post-pruning – from the MDL calculations we can see that If the number of instances is less than 16, the model b overfits the data as the generalization error for a smaller tree is better than the more complex tree hence we can prune the model b.

In post pruning the decision tree is let to grow to its entirety and then trim the nodes of the decision tree in a bottom-up fashion. If the generalization error improves after trimming, replace the sub-tree by a leaf node which is labeled based on the majority class instances in the sub-tree. It useful to reduce overfitting in the decision tree.

Steps in decision tree induction:

The root node is split into 2 non-homogenous nodes.

- The left node is split into a pure node belonging to class C1 and an impure node.
  - The impure node is again split to form 2 classes left C1 and right C2.
- The right node is split into two homogenous nodes left C2 and right C3.

Once the induction tree is built we see the generalization error from the leaf nodes.

- The right node generalization error does not improve on trimming the leaf node hence we no not prune.
- The left leaf nodes generalization error improves on trimming hence we need to prune and assign it to the majority class.
  - Once the leaf nodes are pruned we again check the generalization error and prune and assign it to the majority class which in this case is C1.
  - Again, repeat the process and stop if the generalization error doesn't improve.