

## Question 1

### Machine Learning:

Process of generating software systems that are capable of learning from historical data, for the purpose of generating knowledge representations and making predictions on future data.

Supervised and unsupervised algorithms are two approaches to machine learning problems.

### Decision Trees:

A machine learning method that generates decision boundaries by recursively partitioning the feature set. Many different decision tree algorithms exist, differing in characteristics such as how they select features to split on, how many nodes they produce, and how they determine when to stop splitting. Decision trees can also be used for regression problems. Although primarily used in supervised learning classification problems, they can be used in unsupervised clustering methods, with top-down and bottom-up approaches to generating hierarchical clusters. Smaller, less complex trees that maintain good accuracy are preferred over large complex trees, as they are more likely to generalize to new data better and are computationally less expensive.

### Overfitting:

Overfitting occurs when the machine learning model memorizes the training data or fits too closely to the training data, resulting in reduced test set accuracy. Overfitting can be detected by evaluating the model with an evaluation set. An indicator of overfitting is very high accuracy on the training data with a significantly lower accuracy on the evaluation data. Overfitting often occurs when models are too complex and contain too many parameters. The bias vs variance tradeoff is an important topic that considers overfitting vs underfitting in an attempt to produce an ideal model.

### Pre-pruning and post-pruning for decision tree learning:

To reduce overfitting, we often need to stop the decision tree from growing into a model that is too complex. Pre-pruning will stop the tree from growing beyond a certain point during the training process, when some pre-defined node count, instance per leaf count, or error threshold is met. Post pruning will grow the tree to its maximum size, and then remove nodes from the tree which contribute minimally to the overall accuracy, reducing the size of the tree until thresholds are met.

### Information Gain:

A measurement used in the decision tree training process, which calculates the change in entropy before and after a tree split. Entropy is a method for measuring impurity, and when selecting an attribute to split on we desire an attribute that will most reduce impurity. When selecting an attribute to split on, we will usually select the attribute which results in the highest information gain. I say usually, because some intuition must be applied. Unique attributes, like IDs, will lead to maximum information gains, but will yield useless models.

Information Gain Ratio:

Another method for measuring the change in entropy before and after a decision tree split, used in decision tree algorithms to select which attribute to split on, therefore similar to Information Gain. Unlike Information Gain however, it penalizes attributes which split into many nodes containing few instances. It achieves this by dividing the Gain by the Split Information. Split Information is a function of instances per subset divided by total number of subsets. Splitting on an attribute, such as a unique ID, will result in a high Split Information value, therefore reducing the Information Gain Ratio in favor of other attributes that produce a smaller Split Information.

Gini Index:

Gini Index is another method for selecting attributes to split on during decision tree learning. Gini Index produces a value in the range  $[0, 1]$ , and it is equal to 1 minus the summation of relative frequencies for all instances in the set. The Gini Split value is calculated as the weighted sum of Gini scores for each resulting subset after the split.

Explain why ID3 decision trees are said to be unstable, and why ID3 decision trees are said to be robust to errors:

An unstable learner is one that is very sensitive to small changes in the training data. The ID3 decision tree learner uses Information Gain to choose attributes for splitting on, so it always selects the attribute which produces the smallest entropy. A change in the training data can lead to a different attribute being selected for splitting, which will trickle down the tree affecting other nodes and potentially changing the entire tree structure. ID3 trees are robust to errors in the sense that they handle samples with missing values well. If a training sample is missing a value, the missing value can be replaced with the avg value for that attribute, and all other properties of the training sample can still be used to train the tree.

## Question 2

Explain roles of interior nodes vs leaf nodes of the decision trees:

Interior nodes are decision nodes which split the data on a specific attribute and value. In this example, interior nodes are splitting data samples on attributes  $x_1$  and  $x_2$  based on numeric values. Leaf nodes on a decision tree can provide a class, numeric value, or even a regression model. In this example the leaf nodes provide a class label (color/shape).

Explain process of building decision trees through recursive partitioning of the feature space.

In general, decision trees are built by:

1. Select best attribute to split on
2. Add corresponding decision node to tree
3. Apply decision node and evaluate the resulting child nodes' purity
4. For each child node where impurity > threshold, repeat #1 for given child node
5. Determine when to stop splitting
6. Label leaf nodes with majority class

In this example, the tree could also be built through visual examination because there are only two features, we can easily examine 2D space. A horizontal or vertical line (decision boundary) is drawn which partitions the feature space, separating a particular instance type from the others.

This decision boundary is added to the tree by adding decision nodes which reflect the line's corresponding  $x_1$  or  $x_2$  value. For each resulting subset, this process is repeated until each subset is pure, or some threshold is met.

What are the meaning of the dashed lines and how does each dashed line correspond to the nodes in the tree.

Each dashed line is a decision boundary which separates instances of different types. A horizontal line has a corresponding  $x_2$  value. These lines result in tree nodes where  $x_2$  is greater than or less than the  $x_2$  value at said line. Vertical lines result in decision nodes that compare the value of  $x_1$ . Each horizontal or vertical line maps to 2 decision nodes in the tree, one whose child nodes are less than and one whose children nodes are greater than.

### Question 3

Calculate Entropy of dataset in table 1, then use information gain to determine which attribute has the highest information gain.

The original entropy of the data set in table 1 is 0.971. There are 9 positive classes (play) and 6 negative classes (don't play). Entropy is calculated using

$$\text{Entropy}(S) = -\sum p_i \log_2(p_i), \text{ where } p_{\text{positive}} = 9/15 \text{ and } p_{\text{negative}} = 6/15 = 0.971$$

Next the information gain is calculated for all 4 attributes. For each attribute, first the entropy of each possible value is calculated, and then these are combined to form the conditional entropy, which is the weighted sum of entropy for each subset produced by the attribute.

For example – the Humidity attribute information gain is calculated by:

Construct the subsets generated by the attribute,  $S_{\text{HIGH}}$  and  $S_{\text{NORMAL}}$ .

$S_{\text{HIGH}} = \{1, 2, 3, 4, 8, 12, 14, 15\}$  with 4 positive and 4 negative labels

$S_{\text{NORMAL}} = \{5, 6, 7, 9, 10, 11, 13\}$  with 5 positive and 2 negative labels

Calculate entropy for each subclass using entropy equation above.

$$\text{Entropy}(S_{\text{HIGH}}) = 1.0 \quad \text{Entropy}(S_{\text{NORMAL}}) = 0.936$$

Calculate conditional entropy for the attribute:

$$\text{Conditional Attribute (Humidity)} = (8/15) * (1.0) + (7/15) * (.936) = 0.936$$

Calculate information gain = previous entropy – new conditional entropy

$$\text{Info Gain(humidity)} = 0.971 - 0.936$$

This process is repeated for all possible attributes, then the attribute which results in highest information gain is selected.

Attribute	Conditional Entropy	Information Gain
Outlook	0.907	0.064
Temperature	0.943	0.028
Humidity	0.936	0.035
Wind	0.967	0.004

The outlook attribute yields the highest information gain, so we select outlook as first attribute to split on.

#### Question 4

The data set from Table 1 is used to create a decision tree, using Information Gain Ratio (IGR) as attribute selection criteria.

First IGR is calculated for all attributes to determine which attribute to split on first. Calculating IGR entails calculating the entropy of each subset, the conditional entropy of the split, the Information Gain of the split, and the Split Information. Equations to all listed below:

$$\text{Entropy}(S) = -\sum p_i \log_2(p_i)$$

$$\text{Conditional Entropy} = \sum (|S_i| / |S|) * \text{entropy}(S_i)$$

$$\text{Information Gain} = \text{entropy}(S) - \text{conditional entropy of split}$$

$$\text{Split Information} = -\sum (|S_i| / |S|) * \log_2(|S_i| / |S|)$$

$$\text{Information Gain Ratio} = \text{Information Gain} / \text{Split Information}$$

Calculating IGR for all attributes yields Info Gain Ratios:

Attribute	Conditional Entropy	Information Gain	Split Information	Information Gain Ratio
Outlook	0.907	0.064	1.566	0.041
Temp	0.943	0.028	1.53	0.018
Humidity	0.936	0.035	0.997	0.035
Wind	0.967	0.004	0.997	0.004

Therefore, the first split will be on the Outlook attribute. This generates 3 new subsets of data:  $S_2 = \{1, 2, 8, 9, 11\}$ ,  $S_3 = \{3, 7, 12, 13\}$ , and  $S_4 = \{4, 5, 6, 10, 14, 15\}$ . The preceding steps will be repeated for each subset, to determine what to split each subset on. This process is repeated until all leaf nodes are pure given the training data.

Splitting  $S_2$ :

Attribute	Conditional Entropy	Information Gain	Split Information	Information Gain Ratio
Temp	0.4	0.571	1.522	0.375
Humidity	0	0.971	0.971	1
Wind	0.951	0.02	0.971	0.021

S2 will split on Humidity, yielding the Play classification if humidity is normal, and yielding Don't Play if humidity is high.

Splitting S3:

Attribute	Conditional Entropy	Information Gain	Split Information	Information Gain Ratio
Temp	0.5	0.311	1.5	0.207
Humidity	0.5	0.311	1	0.311
Wind	0.5	0.311	1	0.311

S3 can be split on humidity or wind because they have same IGR. We will arbitrarily select humidity. S3 will split on humidity and if humidity is high, the child node will return positive Play label. If humidity is normal, a new subset  $S5 = \{7, 13\}$  is generated which requires additional splitting.

Splitting S4:

Attribute	Conditional Entropy	Information Gain	Split Information	Information Gain Ratio
Temp	0.792	0.126	1.459	0.084
Humidity	0.918	0	1	0
Wind	0.459	0.459	1	0.459

Wind is selected as best attribute to split S4. Weak winds will yield node that returns positive class label, Play. Strong winds will yield a new subset, S6, which must be split again.

Splitting S5:

Attribute	Conditional Entropy	Information Gain	Split Information	Information Gain Ratio
Temp	0	1	1	1
Wind	0	1	1	1

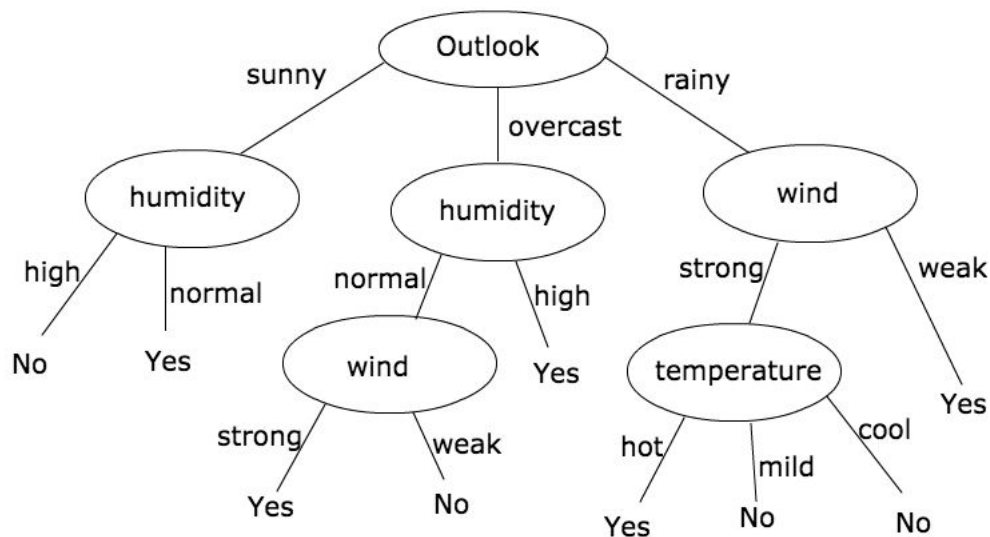
S5 split on temperature and wind attributes equally, separating the last two instances into positive and negative nodes. We've arbitrarily selected wind as the attribute to split on, since both options are equivalent information gain ratios.

Splitting S6:

Attribute	Conditional Entropy	Information Gain	Split Information	Information Gain Ratio
Temp	0	0.918	1.585	0.579
Wind	0.667	0.252	0.918	0.274

S6 will be split on temperature. Hot will yield Play, while Mild and Cool will both output negative Don't Play labels.

The resulting decision tree:



### Question 5

Use Gini Index to calculate correlation between the four attributes and class labels, then select the most important attribute to build the root node of the decision tree.

All 4 attributes are evaluated for splitting using Gini Index:

$$\text{Gini}(\text{subset}) = 1 - \sum f_i^2$$

$$\text{Gini Split}(\text{attribute}) = \sum (|S_i|/|S|) * \text{Gini}(S_i)$$

Outlook

$Gini(sunny) = 0.48$ ,  $Gini(overcast) = 0.375$ ,  $Gini(rain) = 0.5$

$Gini\ Split(outlook) = 0.460$

Temperature

$Gini(hot) = 0.5$ ,  $Gini(mild) = 0.489$ ,  $Gini(cool) = 0.375$

$Gini\ Split(temperature) = 0.462$

Humidity

$Gini(high) = 0.5$ ,  $Gini(normal) = 0.408$

$Gini\ Split(humidity) = 0.457$

Wind

$Gini(strong) = 0.489$ ,  $Gini(weak) = 0.469$

$Gini\ Split(wind) = 0.478$

The 4 attributes are ranked from best to worst score, alongside the rankings obtained using Information Gain Ratio in the following table:

Ranking	Attribute	
	Gini Index	Information Gain Ratio
1	humidity	outlook
2	outlook	humidity
3	temperature	temperature
4	wind	wind

Unlike Information Gain and Information Gain Ratio, the Gini Index method of attribute selection chose humidity as the best attribute to split on first. With both methods of measurement, wind is determined to be the worst attribute to split on first.

## Question 6

The following R script is used to execute parts 1 – 4 of Question 6:

```
1 # load data
2 filePath = "data/housing.header.txt"
3 housing = read.table(filePath, header=TRUE, sep=",")
4
5 # create correlation matrix
6 # allows us to view all attribute pairwise correlations in tabular format
7 corrMatrix = cor(housing)
8
9 # create level plot for viewing attribute correlations
10 library('lattice')
11 levelplot(corrMatrix)
12
13 # LStat has the lowest correlation (-0.7376627)
14 # LStat = % lower status of population
15
16 # Rm attribute has highest correlation (0.69535995)
17 # Rm = average number of rooms per dwelling
18
19 # Plotting both LStat vs Medv and Rm vs Medv
20 dev.off()
21 par(mfrow=c(2, 1))
22 plot(housing$Lstat, housing$Medv, xlab='Lstat', ylab='Medv')
23 plot(housing$Rm, housing$Medv, xlab='Rm', ylab='Medv')
24
25
```

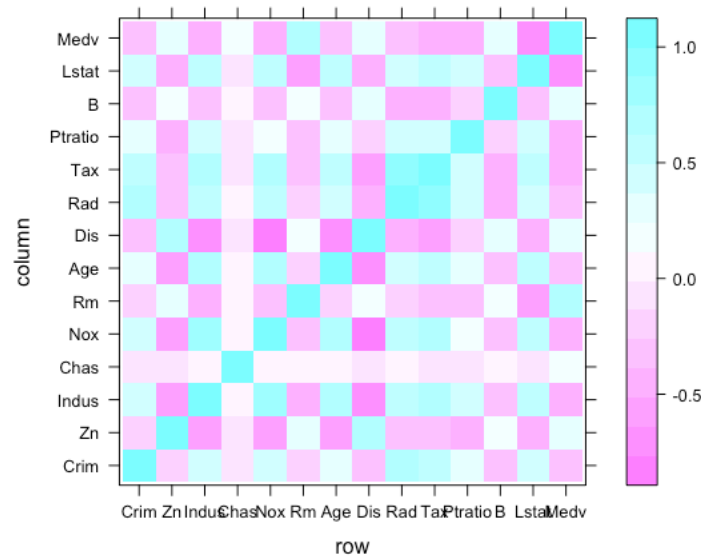
Report pairwise correlation between every two variables using a matrix or level plot. This is accomplished using R's built in 'cor' function on line 7. The resulting matrix displays all pairwise attribute correlations.

Below is a preview of the matrix, note that the diagonal consists of 1.0s, as expected.

	Crim	Zn	Indus	Chas	Nox	Rm	Age	Dis	Rad	Tax
Crim	1.00000000	-0.20046922	0.40658341	-0.055891582	0.42097171	-0.21924670	0.35273425	-0.37967009	0.625505145	0.58276431
Zn	-0.20046922	1.00000000	-0.53382819	-0.042696719	-0.51660371	0.31199059	-0.56953734	0.66440822	-0.311947826	-0.31456351
Indus	0.40658341	-0.53382819	1.00000000	0.062938027	0.76365145	-0.39167585	0.64477851	-0.70802699	0.595129275	0.72076011
Chas	-0.05589158	-0.04269672	0.06293803	1.00000000	0.09120281	0.09125123	0.08651777	-0.09917578	-0.007368241	-0.03558651
Nox	0.42097171	-0.51660371	0.76365145	0.091202807	1.00000000	-0.30218819	0.73147010	-0.76923011	0.611440563	0.66802321
Rm	-0.21924670	0.31199059	-0.39167585	0.091251225	-0.30218819	1.00000000	-0.24026493	0.20524621	-0.209846668	-0.29204781
Age	0.35273425	-0.56953734	0.64477851	0.086517774	0.73147010	-0.24026493	1.00000000	-0.74788054	0.456022452	0.50645551
Dis	-0.37967009	0.66440822	-0.70802699	-0.099175780	-0.76923011	0.20524621	-0.74788054	1.00000000	-0.494587930	-0.53443151
Rad	0.62550515	-0.31194783	0.59512927	-0.007368241	0.61144056	-0.20984667	0.45602245	-0.49458793	1.00000000	0.91022811
Tax	0.58276431	-0.31456352	0.72076018	-0.035586518	0.66802320	-0.29204783	0.50645559	-0.53443158	0.910228189	1.00000000
PtRatio	0.28994558	-0.39167855	0.38324756	-0.121515174	0.18893268	-0.35550149	0.26151501	-0.23247054	0.464741179	0.46085301
B	-0.38506394	0.17552032	-0.35697654	0.048788485	-0.38005064	0.12806864	-0.27353398	0.29151167	-0.444412816	-0.44180801
Lstat	0.45562148	-0.41299457	0.60379972	-0.053929298	0.59087892	-0.61380827	0.60233853	-0.49699583	0.488676335	0.54399934
Medv	-0.38830461	0.36044534	-0.48372516	0.175260177	-0.42732077	0.69535995	-0.37695457	0.24992873	-0.381626231	-0.46853351

This correlation matrix can then be fed to a levelplot function made available by the 'Lattice' package. The levelplot provides a nice visual representation of attribute correlations:



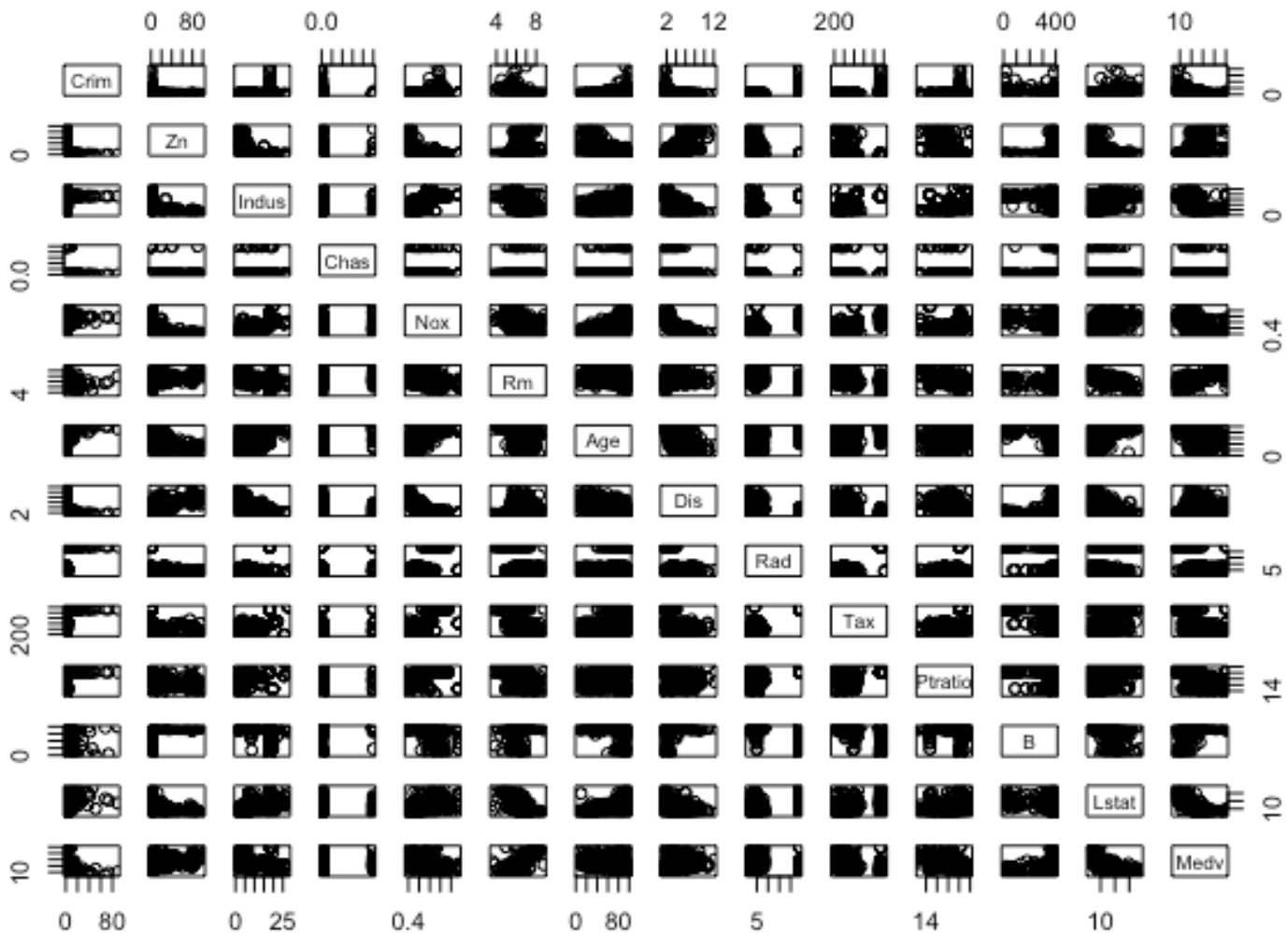


The opacity of the boxes represents the strength of the correlation. Blue boxes are positive correlations and pink boxes are negative correlations. The top row displays the correlations between Medv and all other attributes. The strongest blue is found at intersection with Rm, and the strongest pink is found at intersection with Lstat.

Explain which attribute is most positively/negatively correlated to Medv:

By examining correlation matrix, the Rm attribute is most positively correlated to Medv, with a correlation value of 0.695. The positive correlation reveals Medv is expected to increase as Rms increase. This makes sense – we expect housing values to increase as the number of rooms increase. The Lstat is most negatively correlated to Medv, with a correlation value of -0.737. As Lstat (% lower status population) decreases, the housing values increase, and vice versa.

Draw scatterplots to show relationship between the attributes:



This matrix of scatter plots displays the correlations between all pairwise attributes. It is difficult to read at this resolution but is able to be blown up and viewed much larger, providing efficient method for quickly examining correlations between all attributes.

Explain how to use scatterplots to find attributes which are positively correlated, negatively correlated, or independent of Medv.

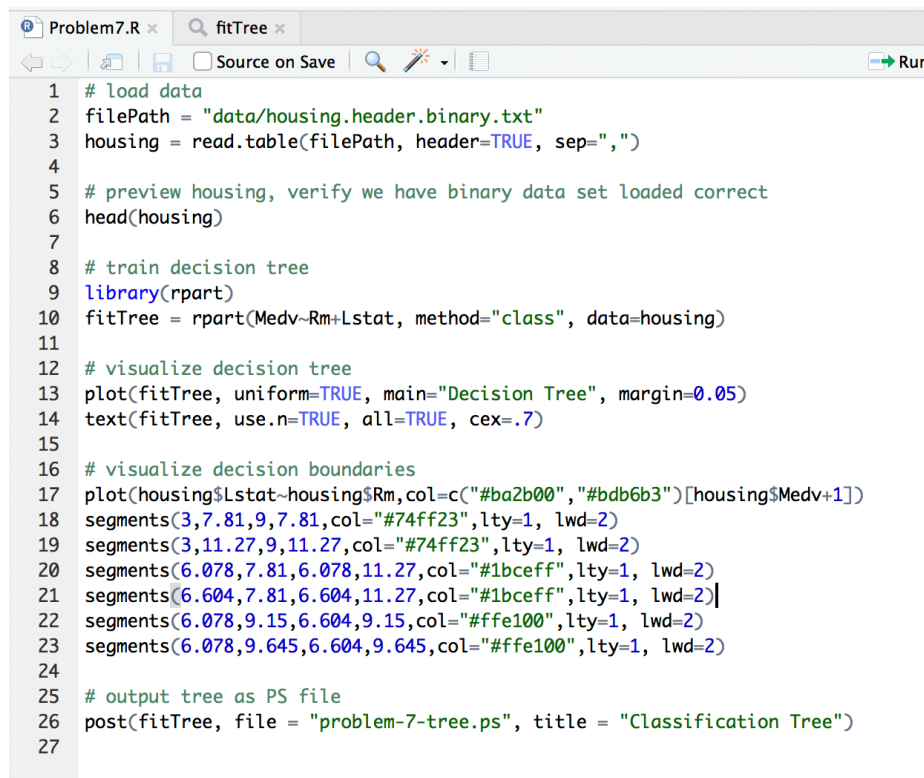
Detecting correlations with the Medv attribute can be achieved by first plotting all attributes vs Medv. Then, visually examine each plot in search of a positive or negative trendline. In the above chart we can refer to the downward trend in the 14<sup>th</sup> column, 13<sup>th</sup> row. This displays the decrease in Medv as Lstat increases. The trends that are detected through visualization are representative of the correlation between the two attributes. Attributes which are independent of

Medv will produce plots that appear either random, or straight horizontal/vertical lines. We will not be able to visually detect any positive or negative sloped trends if the attributes are not correlated.

## Question 7

Download housing.header.binary.txt and use R to complete below tasks. Note this is same as data set from Question 6, except: Medv values are now binary = 1 if value > 200K, else = 0

The following R script was created to solve all tasks associated with question 7:

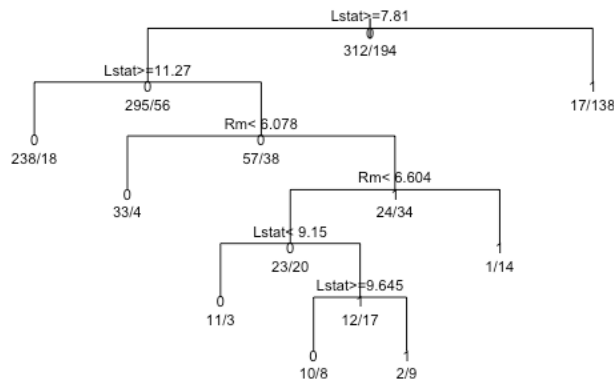


```
1 # load data
2 filePath = "data/housing.header.binary.txt"
3 housing = read.table(filePath, header=TRUE, sep=",")
4
5 # preview housing, verify we have binary data set loaded correct
6 head(housing)
7
8 # train decision tree
9 library(rpart)
10 fitTree = rpart(Medv~Rm+Lstat, method="class", data=housing)
11
12 # visualize decision tree
13 plot(fitTree, uniform=TRUE, main="Decision Tree", margin=0.05)
14 text(fitTree, use.n=TRUE, all=TRUE, cex=.7)
15
16 # visualize decision boundaries
17 plot(housing$Lstat~housing$Rm,col=c("#ba2b00", "#bdb6b3")[housing$Medv+1])
18 segments(3,7.81,9,7.81,col="#74ff23",lty=1, lwd=2)
19 segments(3,11.27,9,11.27,col="#74ff23",lty=1, lwd=2)
20 segments(6.078,7.81,6.078,11.27,col="#1bceff",lty=1, lwd=2)
21 segments(6.604,7.81,6.604,11.27,col="#1bceff",lty=1, lwd=2)
22 segments(6.078,9.15,6.604,9.15,col="#ffe100",lty=1, lwd=2)
23 segments(6.078,9.645,6.604,9.645,col="#ffe100",lty=1, lwd=2)
24
25 # output tree as PS file
26 post(fitTree, file = "problem-7-tree.ps", title = "Classification Tree")
27
```

The data is loaded (line 2), and then the 'rpart' library is imported for the purpose of training decision tree (line 9). The rpart library is used to create a decision tree using attributes 'Rm' and 'Lstat' to classify attribute 'Medv' as 1 or 0. A 'Medv' value of 1 corresponds to house values greater than 200K.

The resulting decision tree is plotted (lines 13 – 14):

## Decision Tree



We can see that the resulting tree contains 6 decision nodes and 7 leaf nodes. As expected the decision nodes only use the Lstat and Rm attributes to split on.

All decision nodes take the same basic form: the attribute being split on, a comparison operator, and a value that the attribute is being compared to. Instances will split and fall to the left if true or to the right if false. Lstat is the first attribute that the tree splits on, and all instances with a Lstat value greater than or equal to 7.81 will split to the left of the tree, while all other instances will split to the right and receive label 1 (> \$200K).

All leaf nodes contain the label of the leaf node (0 or 1 in this case). This is the label that will be assigned to new instances being classified that end up in this node. This label is determined by selecting the majority class from the training data instances.

All nodes contain the distribution of the instances at that node (instances with label 0 / instances with label 1). The text of the root node's right child for example, 17/138, corresponds to the 17 instances with label 0 and the 138 instances with label 1 that will fall into this leaf node. This allows us to quickly see the purity of the nodes during the training process.

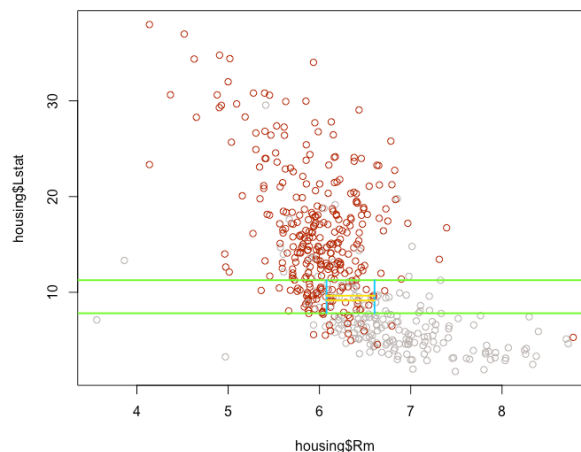
Each node explained in BFS order:

1. Decision node - Samples with Lstat  $\geq 7.81$  split to left, otherwise to right. Out of all instances in this node, 312 have Medv of 0, 194 have Medv of 1.
2. Decision node - Samples with Lstat  $\geq 11.27$  split to left. 295 training instances have Medv of 0, 56 have Medv of 1.
3. Leaf node – instances are labelled with Medv of 1. 17 training instances have Medv 0 and 138 instances have Medv of 1.
4. Leaf node – instances are labelled with Medv of 0. 238 training instances have Medv of 0, 18 have Medv label of 1.
5. Decision node – RM  $< 6.078$  split left, others split right. 57 instances at this node have Medv label 0, 38 have Medv label 1.

6. Leaf node – instances are labelled with Medv of 0. 33 training instances have label 0 and 4 have label 1.
7. Decision node –  $Rm < 6.604$  split left. 24 training instances have label 0, 34 instances have label 1.
8. Decision node –  $Lstat < 9.15$ . 23 training instances have label 0, 20 instances have label 1.
9. Leaf node – instances labelled with Medv of 1. 1 training instance has label 0, and 14 have label 1.
10. Leaf node – instances labelled with Medv of 0. 11 training instances were labelled 0 and 3 were labelled 1.
11. Decision node –  $Lstat \geq 9.645$  split left, others split right. 12 training instances here have label 0, 17 have label 1.
12. Leaf node – instances are labelled with Medv of 0. 10 instances have label 0 and 8 have label 1.
13. Leaf node – Instances are labelled with Medv 1. 2 instances have label 0, 9 have label 1.

Examining the tree, we can see that most instances fall into two categories –  $Lstat < 7.81$  (155 instances) and  $Lstat \geq 11.27$  (256 instances). These two decision boundaries are marked in green in the plot to follow. The tree then handles the few remaining instances by continuing to recursively partition until the learner's splitting threshold is met and splitting halts.

Since the decision tree was constructed with just 2 attributes, we can plot the instances in this 2-dimensional space (Medv vs Lstat) and draw horizontal/vertical line segments for every decision node in the tree, allowing us to visualize the decision boundaries created. The 2-dimensional space and line segments corresponding to the tree's decision nodes are presented on the next page.



The above image depicts the decision boundaries generated by the decision tree learner. The learner recursively partitioned the 2-D space in order to maximize subset purity.

Finally, the tree was saved as a PS file and added to the report. A preview of the PS file is below:

Classification Tree

