Business Data Mining (IDS 572)

Homework 2-Solution

Question 1

Handling absurd values

- The number of data instances being only 768, leaving out examples with absurd values was not a viable alternative.
- The following attributes values were considered absurd:
  - Glucose concentration = 0
  - Diastolic blood pressure <= 30
  - Triceps skinfold thickness = 0
  - Insulin = 0
  - Bmi <= 10
- Absurd values could replace with missing values NA and handled by creating a maximum 5 surrogates where possible.
- Outliers are also replaced by median.

(a) There are 769 samples, 8 quantitative variables and one target variable with two factors: with or without signs of diabetes.

> # Read data
> setwd('.')
> pim= read.csv("./pim.csv", header = FALSE)
> na.pim = pim
>str(na.pim)

```
'data.frame': 768 obs. of  9 variables:
 $ pregnantnum: int  6 1 8 1 0 5 3 10 2 8 ...
 $ glucose    : int  148 85 183 89 137 116 78 115 197 125 ...
 $ bp         : int  72 66 64 66 40 74 50 0 70 96 ...
 $ triceps    : int  35 29 0 23 35 0 32 0 45 0 ...
 $ insulin    : int  0 0 0 94 168 0 88 0 543 0 ...
 $ bmi        : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
 $ pedigree   : num  0.627 0.351 0.672 0.167 2.288 ...
 $ age        : int  50 31 32 21 33 30 26 29 53 54 ...
 $ diabetes   : int  1 0 1 0 1 0 1 0 1 1 ...
```

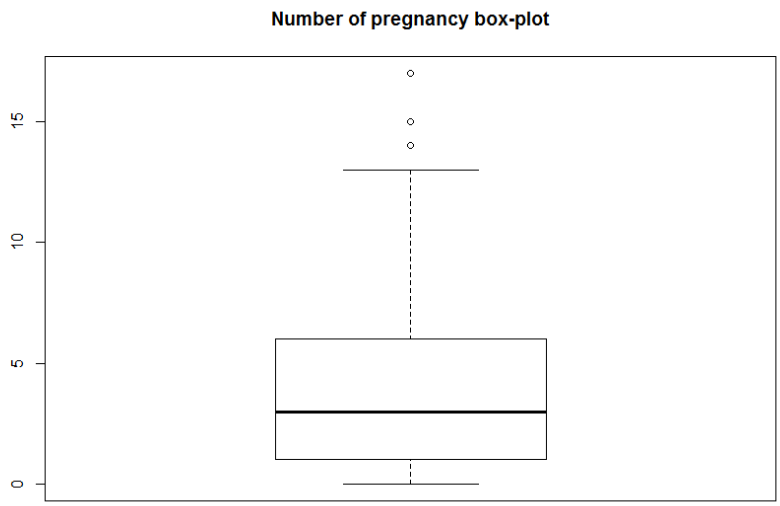
As you can see the target variable is considered as a continuous variable. So we have to convert it the "binary" variable.

```
>#Convert the class variable from integer to factor
>na.pim$diabetes = factor(na.pim$diabetes)
```

```
> colnames(na.pim) = c(“pregnant”, “glucose”, “bp”, “triceps”, “insulin”, “bmi”, “diabetes”, “age”, “class”)
```

We first check each attribute in the data set.

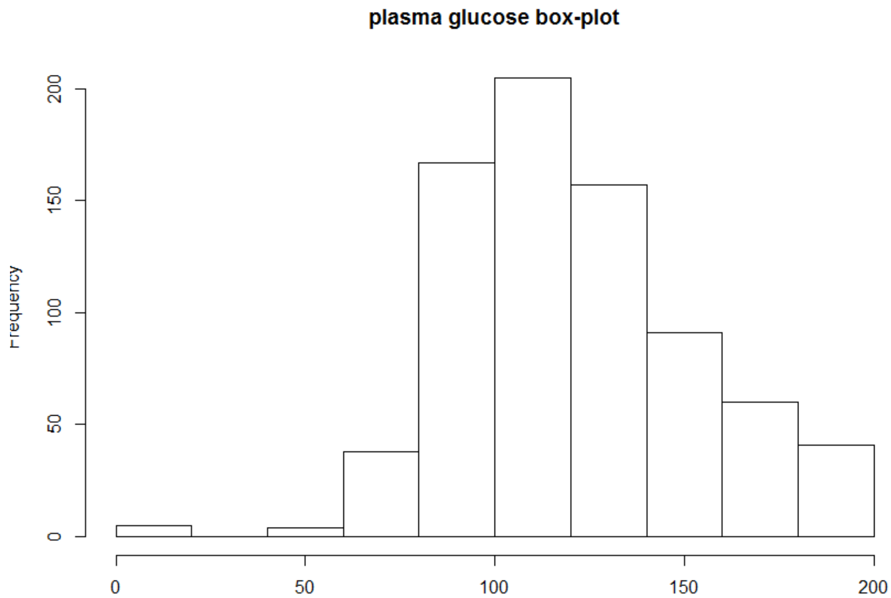
1. Pregnancy



There are few outliers in this variable.

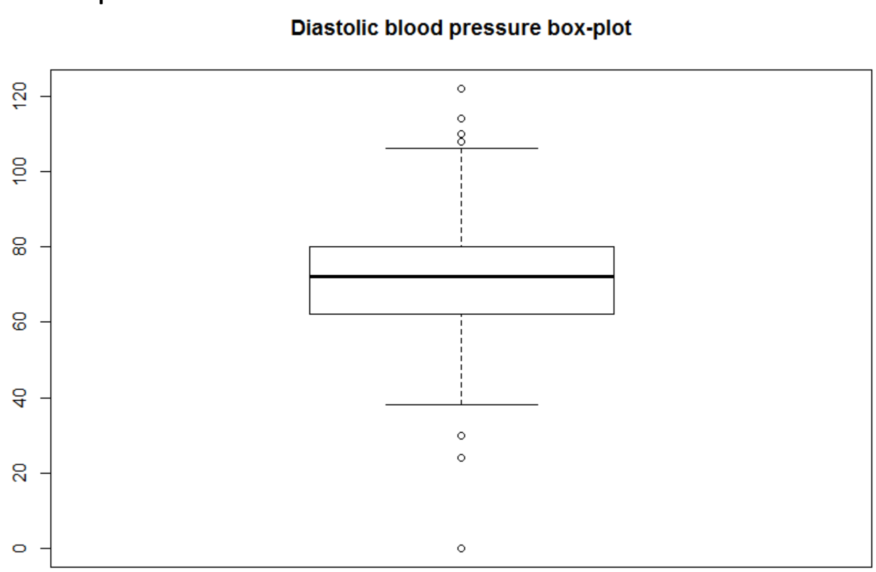
```
> outliers = boxplot.stats(na.pim$pregnancy)$out
> pregnancy=ifelse(na.pim$pregnancy % in % outliers, “NA”, na.pim$pregnancy)
> na.pim [, “pregnancy”] = pregnancy
```

2. Glucose



The 0s are outliers in glucose variable.
>na.pim[na.pim$glucose ==0, “glucose”] = “NA”

3. Blood pressure



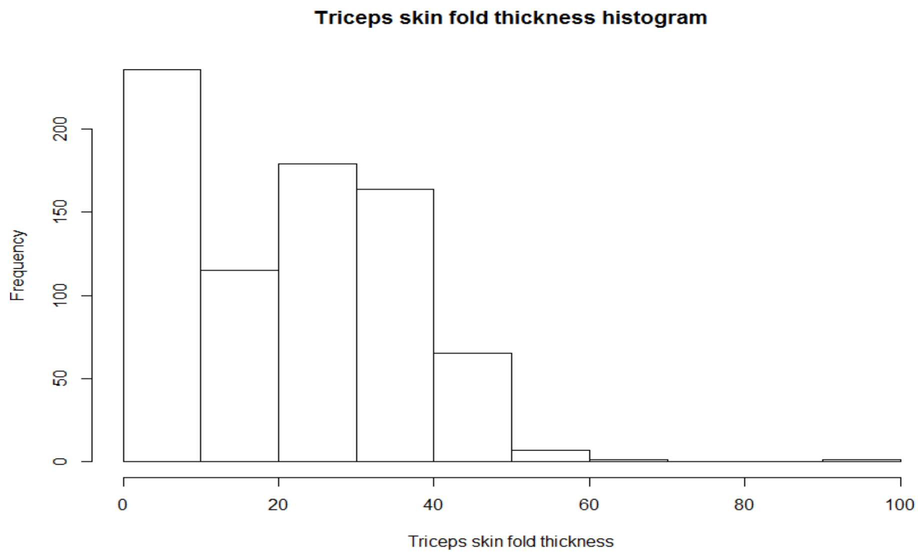
> outliers = boxplot.stats(na.pim$bp)$out

> bp=ifelse(na.pim$bp % in % outliers, “NA”, na.pim$bp)

>bp[bp <= 30] = “NA”

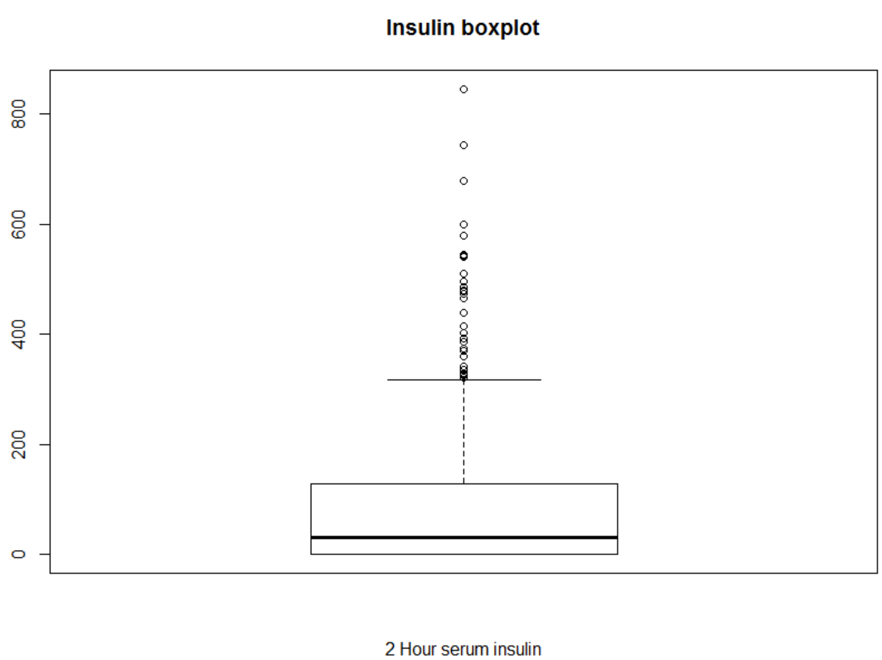
> na.pim[, “bp”] = bp

4. Triceps skin fold thickness



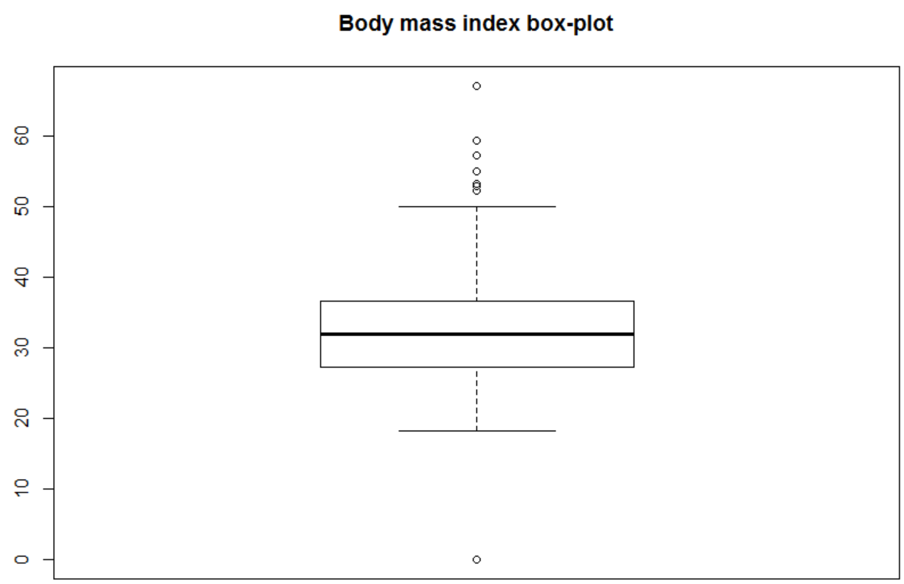
> na.pim[na.pim$trecipes == 0 & na.pim$trecipes == 99, “triceps”] = “NA”

5. Insulin



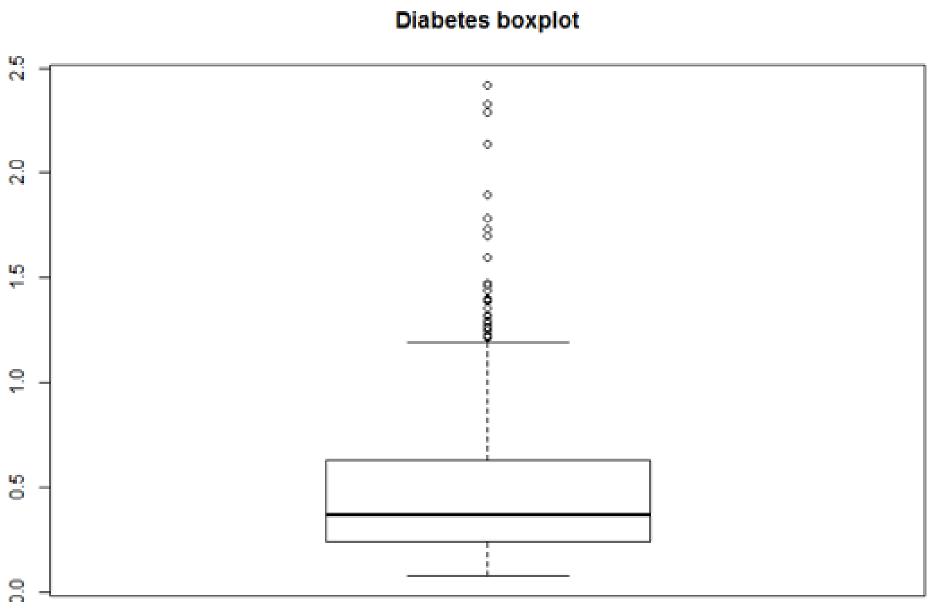
```
> outliers = boxplot.stats(na.pim$insulin)$out
> insulin = ifelse(na.pim$insulin % in % outliers, “NA”, na.pim$insulin)
> insulin[insulin ==0] = Na
> na.pim[, “insulin”] = insulin
```

6. Body mass index




```
> outliers = boxplot.stats(na.pim$bmi)$out
> bmi = ifelse(na.pim$bmi % in % outliers, NA, na.pim$bmi)
> na.pim[, “bmi”] = bmi
```

7. Diabetes pedigree function

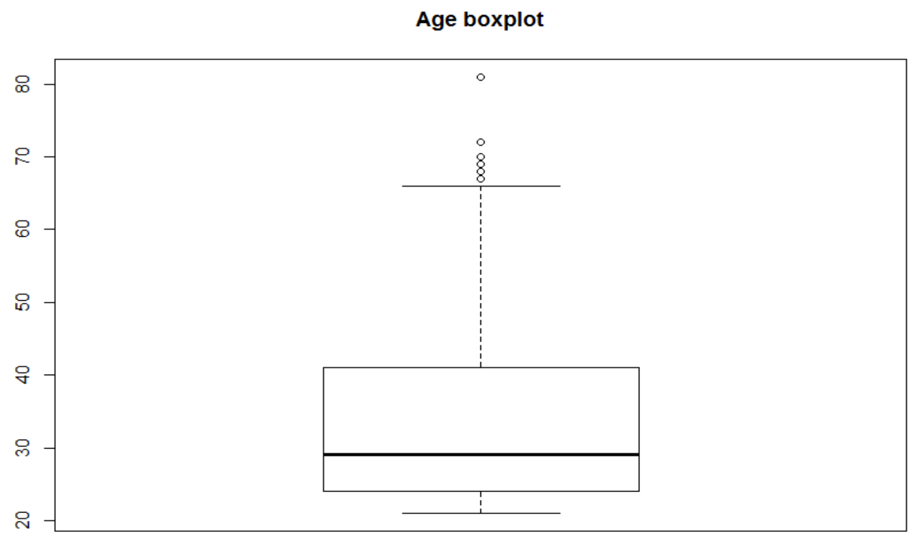


> outliers = boxplot.stats(na.pim$diabetes)$out

> diabetes = ifelse(na.pim$disbetes % in % outliers, NA, na.pim$diabetes)

> na.pim[, “diabetes”] = diabetes

8. Age



> outliers = boxplot.stats(na.pim$age)$out

> age = ifelse(na.pim$age % in % outliers, “NA”, na.pim$age)

> na.pim[, “age”] = age

(b)

> set.seed(1234)

> ind = sample(2, nrow(na.pim), replace = T, prob = c(0.8, 0.2))
> pimTrain = na.pim[ind ==1,]
> pimTest = na.pim[ind == 2,]

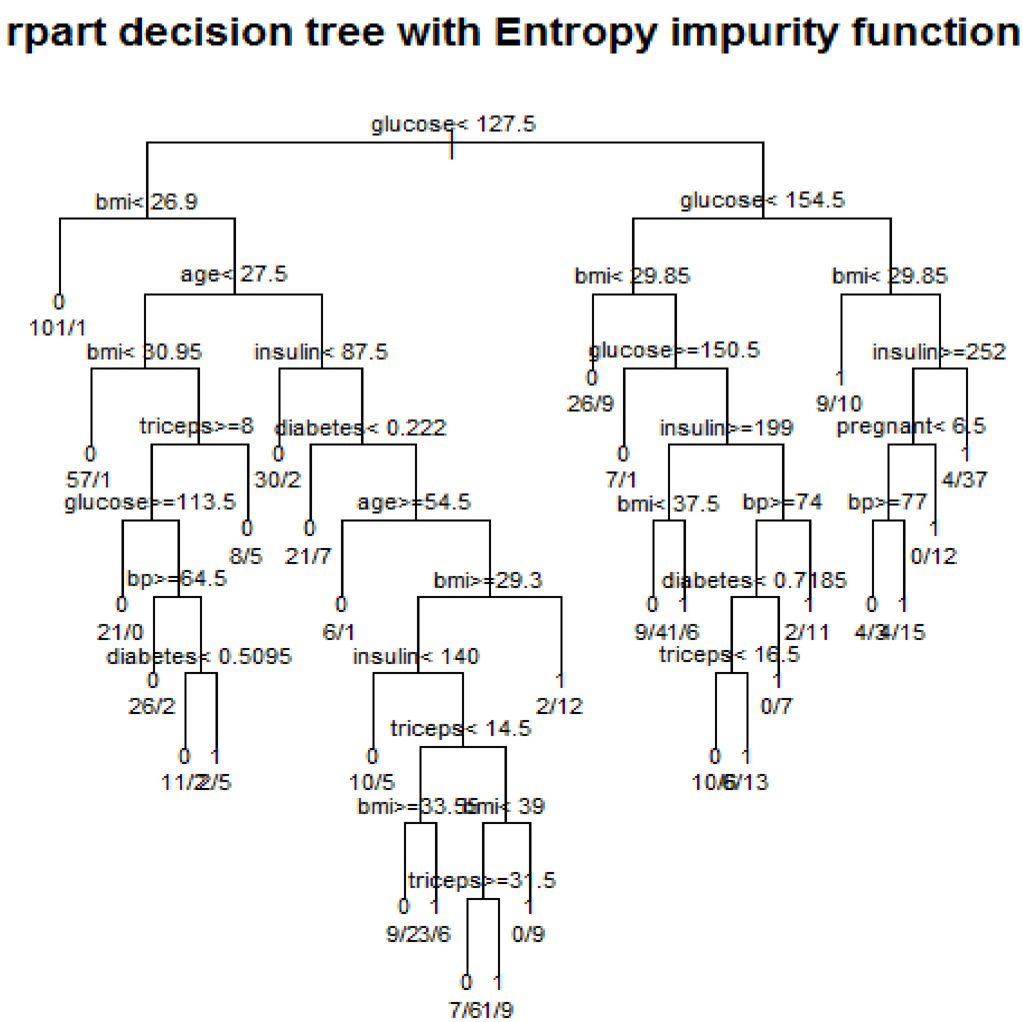
(c), (d), and (f)

We use the following code to build a rpart decision tree

>#------------------ INFORMATION -----------------

>fit = rpart(class ~ ., method = “class”, data = pimTrain, parms = list(split =”information”), control = rpart.control(xval = 5, cp = 0))

```
>plot(fit, uniform = T, main = "rpart decision tree with Entropy impurity function")
>text(fit, use.n = T, all = T, cex = 0.7, xpd = T)
>printcp(fit)
```



	CP	nsplit	rel error	xerror	xstd
1	0.2488038	0	1.00000	1.00000	0.055987
2	0.0406699	1	0.75120	0.79426	0.052529
3	0.0287081	3	0.66986	0.76077	0.051817
4	0.0143541	4	0.64115	0.78947	0.052430
5	0.0119617	9	0.55981	0.78947	0.052430
6	0.0063796	15	0.47847	0.82775	0.053197
7	0.0028708	18	0.45933	0.82297	0.053104
8	0.0023923	23	0.44498	0.87560	0.054077
9	0.0011962	25	0.44019	0.87560	0.054077
10	0.0000000	29	0.43541	0.87560	0.054077

There are 30 leaves (terminal nodes) in this decision tree. The terminal nodes are not completely pure because the rpart function prunes the tree slightly based on the setting of “control” argument.

The prediction on training data is

<table>
<tr><td></td><td></td><td colspan="2">Actual Class</td></tr>
<tr><td></td><td></td><td>0</td><td>1</td></tr>
<tr><td rowspan="2">Predicted Class</td><td>0</td><td>363</td><td>57</td></tr>
<tr><td>1</td><td>34</td><td>152</td></tr>
</table>

The error rate on training data: (34+57)/nrow(pimTrain) = 15.01%

The prediction of the decision tree on test data is

<table>
<tr><td></td><td></td><td colspan="2">Actual Class</td></tr>
<tr><td></td><td></td><td>0</td><td>1</td></tr>
<tr><td rowspan="2">Predicted Class</td><td>0</td><td>91</td><td>28</td></tr>
<tr><td>1</td><td>12</td><td>31</td></tr>
</table>

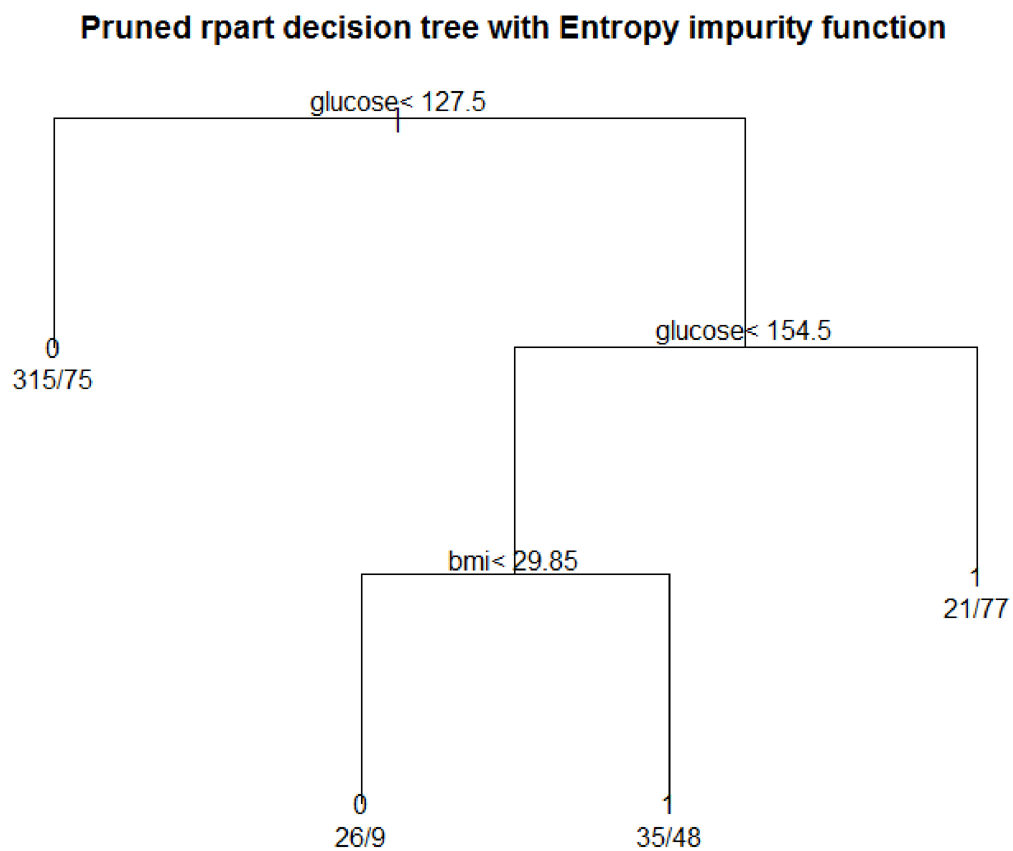
The error rate on test data: (28+12)/nrow(pimTest) = 24.7%

As you can see the prediction on error rate is higher than the test data.

We next prune the tree:

```
>opt = which.min(fit$cptable[,"xerror"])
```

```
>cp = fit$cptable[opt, "CP"]
>tree.prune = prune(fit, cp = cp)
>plot(tree.prune, uniform = T, main = "Pruned rpart decision tree with Entropy
>impurity function")
>text(tree.prune, use.n = T, xpd = T )
```



After pruning data the size of the tree decreased a lot. This tree has 4 terminal nodes.

The prediction of the prune data on training data is

		Actual Class	
		0	1
Predicted Class	0	341	84
	1	56	125

The error rate on training data: (84+56)/nrow(pimTrain)= 23.10%

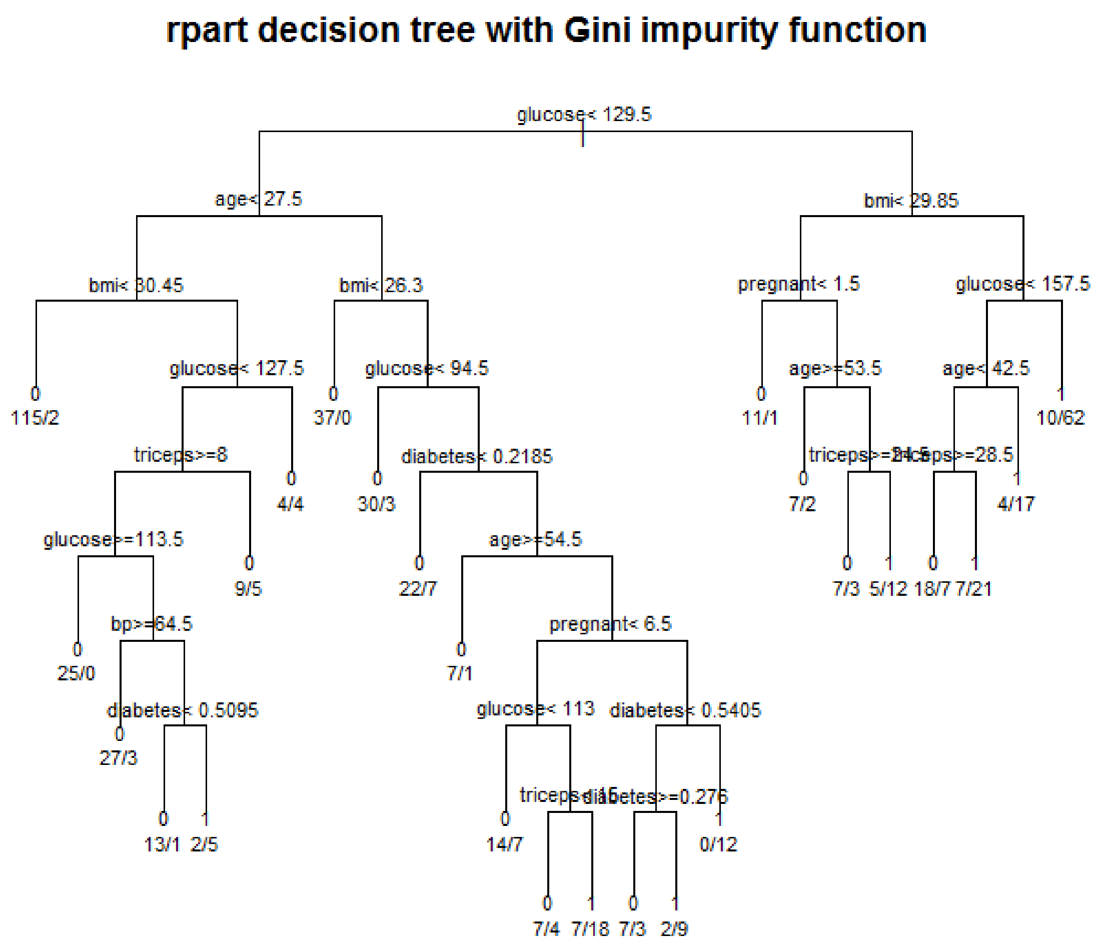
The prediction of the pruned decision tree on test data is

		Actual Class	
		0	1
Predicted Class	0	89	21
	1	14	38

The error rate on test data: (21+14)/nrow(pimTest) = 21.6%

As you can see the error increases on Training data after pruning the tree but the test data error decreases.

```
>#------------------ GINI -----------------
>fit = rpart(class ~ ., method = "class", data = pimTrain, parms = list(split ="gini", control = rpart.control(xval = 5, cp = 0))
>plot(fit, uniform = T, main = "rpart decision tree with Gini impurity function")
>text(fit, use.n = T, all = T, cex = 0.7, xpd =T)
>printcp(fit)
```




```
         CP nsplit rel error  xerror     xstd
1 0.2679426      0   1.00000 1.00000 0.055987
2 0.0574163      1   0.73206 0.75120 0.051605
3 0.0175439      2   0.67464 0.72727 0.051058
4 0.0119617      5   0.62201 0.82297 0.053104
5 0.0111643     13   0.49761 0.77033 0.052025
6 0.0095694     16   0.46411 0.77033 0.052025
7 0.0023923     18   0.44498 0.78469 0.052330
8 0.0000000     24   0.43062 0.84689 0.053559
```

This decision tree has 25 leaves in this tree. This tree is smaller than the tree with Entropy function.

The prediction on training data is

<table>
<tr><td colspan="2"></td><td colspan="2">Actual Class</td></tr>
<tr><td colspan="2"></td><td>0</td><td>1</td></tr>
<tr><td rowspan="2">Predicted Class</td><td>0</td><td>360</td><td>53</td></tr>
<tr><td>1</td><td>37</td><td>156</td></tr>
</table>

The error rate on training data: (53+37)/nrow(pimTrain) = 14.85%

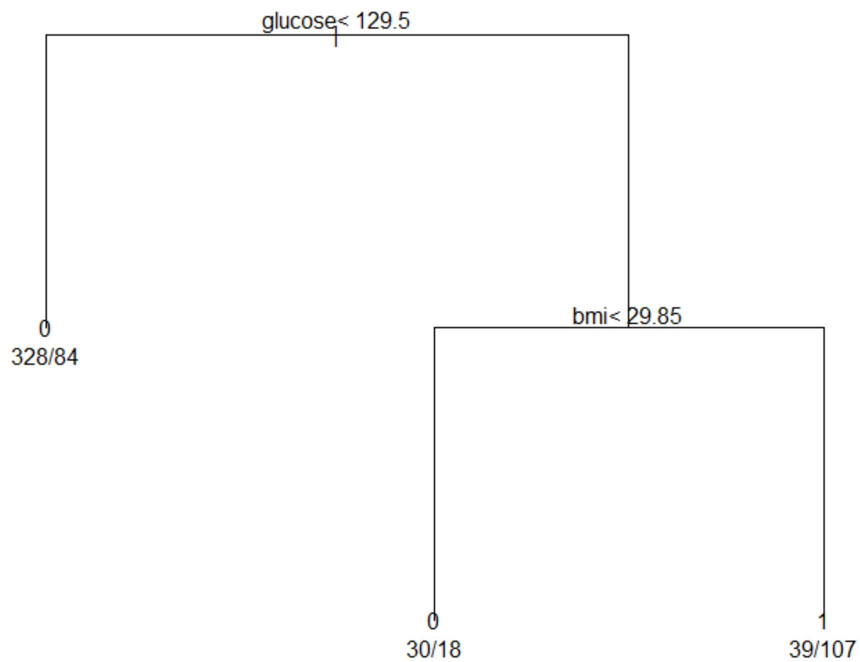
The prediction of the decision tree on test data is

<table>
<tr><td colspan="2"></td><td colspan="2">Actual Class</td></tr>
<tr><td colspan="2"></td><td>0</td><td>1</td></tr>
<tr><td rowspan="2">Predicted Class</td><td>0</td><td>82</td><td>26</td></tr>
<tr><td>1</td><td>21</td><td>33</td></tr>
</table>

The error rate on test data: (26+21)/nrow(pimTest) = 29.01%

After pruning this tree we get

Pruned rpart decision tree with Gini impurity function



The pruned tree has three leaves. The prediction of the prune data on training data is

		Actual Class	
		0	1
Predicted Class	0	358	102
	1	39	107

The error rate on training data: (102+39)/nrow(pimTrain)= 23.26%

The prediction of the pruned decision tree on test data is

		Actual Class	
		0	1
Predicted Class	0	90	26
	1	13	33

The error rate on test data: (26+13)/nrow(pimTest) = 24.07%

	Threshold	Entropy	Gini
Threshold on impurity	xval = 5, cp = 0	24.7%	29.01%
Grow tree and prune it	Prune; cp = the value for which xerror is minimum	21.6%	24.07%

The Entropy tree performs better the Gini decision tree.

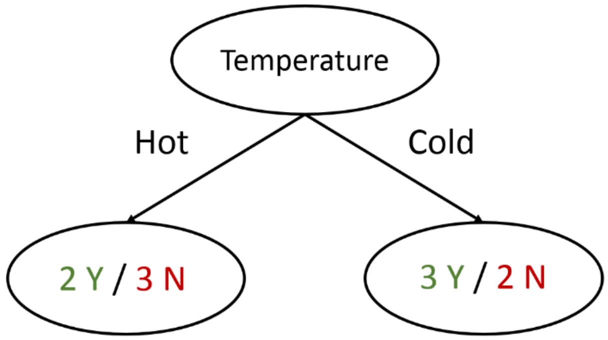
(e) An example of a strong if-then decision rule using the decision tree with Entropy measure (without pruning) is “if glucose < 127.5 and bmi < 26.9 then diabetes = 0”. This rule is strong because it has high confidence (98%) and high support (20%).

(g) Run the ctree model and compare it with rpart in the following items:

Size of trees (depth and the number of leaves), accuracy on test data, and important variables.

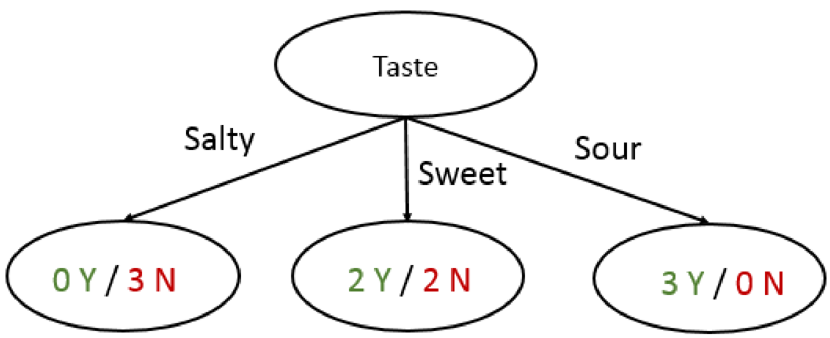
Question 2

(a) By splitting the attribute "Temperature" we get two subsets.

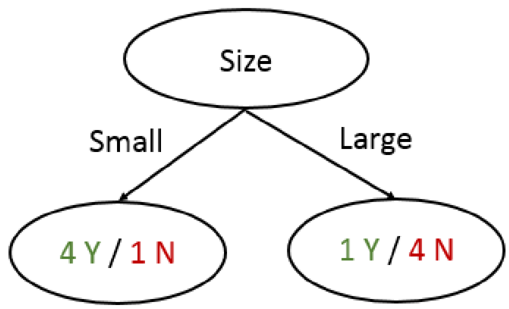


Info[Hot] = $-3/5 \log_2 3/5 - 2/5 \log_2 2/5 = \ 0.44 + 0.53 = 0.97$
Info[Cold] = Info[Hot] = 0.97 (These two subsets are symmetric)
Info[Temp](after split) = 5/10(0.97) + 5/10(0.97) = 0.97
Info[Temp] = 1 (since the distribution of Yes and No class is 50%-50%)
Information Gain = 1-0.97 = 0.03

Similarly, for Taste and Size we have

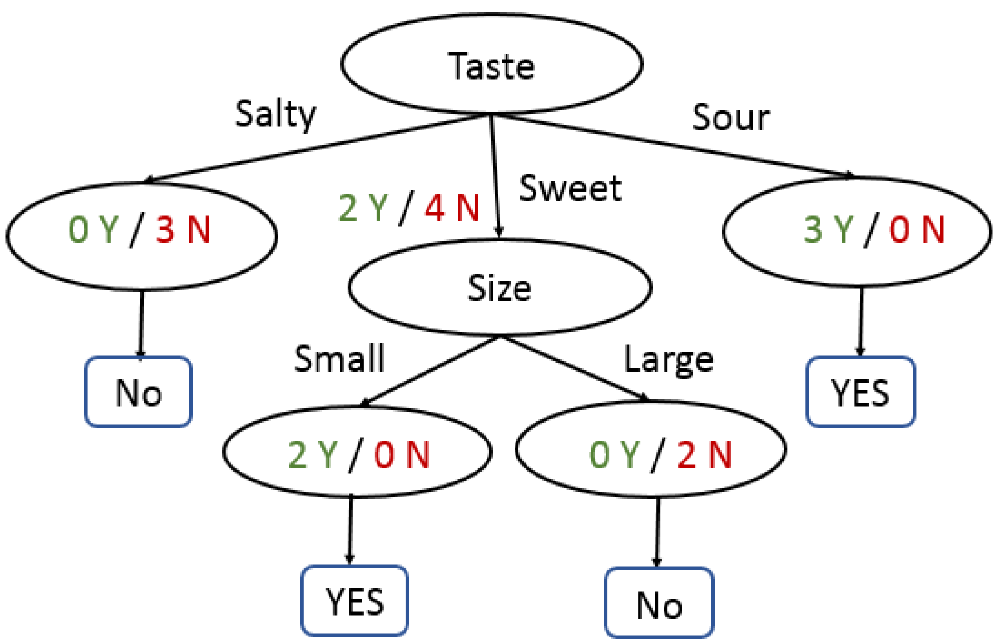


Info[Salty] = Info[Sour] = 0 (These two sets are pure)
Info[Sweet] = 1 (since the distribution of Yes and No class is 50%-50%)
Info[Taste](after split) = 3/10(0) + 4/10(1) + 3/10(0) = 0.4
Info[Taste] = 1
Information Gain = 1-0.4 = 0.6



Info[Small] = Info[Large] = $-4/5\log_2 4/5 - 1/5\log_2 1/5 = \ 0.72$
(These two sets are symmetric)
Info[Size](after split) = 5/10(0.72) + 5/10(0.72) = 0.72
Info[Size] = 1
Information Gain = 1-0.72 = 0.28

(b) The final decision tree is

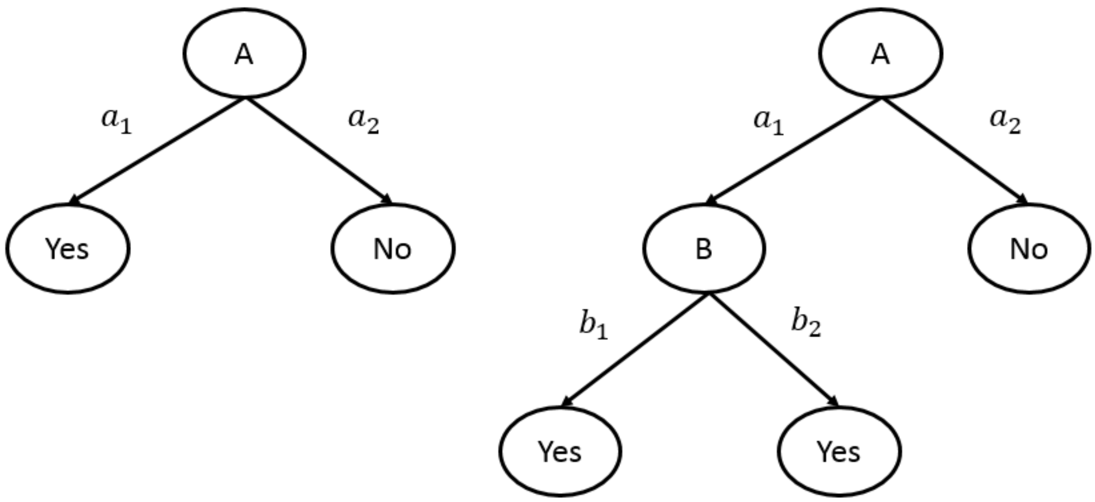


We chose the attribute "Taste" for the first split because it has the highest gain. The decision tree stop splitting other attributes in the subsets "Taste = Sour" and "Taste = Salty" because these subsets are pure. But the subset "Taste = Size" requires further splitting. There are two variable left: Temperature and Size. Doing similar calculation as part (a), we see that Size has higher Information

Gain. So this attribute will be our second candidate for splitting. By splitting Size we observe that all the subsets are pure and we have the final decision tree.

Question 3.

(a) No, the following decision trees have different number of nodes but they always predict the same class label.



(b) Training data is used to build the model, and test data to test it. Just the training data by itself is not able to measure to what extend the model will perform (i.e. generalize to) on unseen data. Test data measures this. We create our training set to increase the accuracy of the classifier, which we use on the data. The more data we train the more accurate the resulting model will be. The test data is used to evaluate the performance of the classifier on a new data.

(c) Overfitting and lack of generalization beyond training data, i.e. models that describe the training data (too) well, but do not model the principles and characteristics underlying the data.