CS 5593 - Fall 2019 - Dr. Le Gruenwald HOMEWORK 2

By Akhila Podupuganti

Problem 1 (30 points): Using the following dataset where the class attribute is "Survived" and using the Decision Tree Induction Algorithm 3.1 given on Page 137 in the textbook, answer the following questions:

Instance	Gender	Age	Ticket Class	Survived
1	Male	45	Second	**Yes**
2	Female	8	First	**Yes**
3	Male	32	Second	**No**
4	Male	26	Third	**No**
5	Female	55	Second	**Yes**
6	Female	47	Third	**No**
7	Male	20	First	**No**
8	Female	24	First	**Yes**
9	Female	43	Second	**No**
10	Male	12	First	**Yes**
11	Male	34	Second	**No**
12	Female	65	Third	**No**

1.1 Show your construction of a decision tree using the information gain for the attribute split test condition and the following stopping condition for a node: either all records in the node have the same class label or the same attribute values or the number of records in the node is less than 3. Show your work (including the information gain calculation) at each split step by step so that we understand how you have constructed the tree.

Information Gain:

Gain(S,X) = Entropy(S) – Entropy(S,X)

Entropy(S) = $\sum_{i=1}^{c} -p_i log_2 p_i$

Entropy(S,X) = $\sum_{c \in X} P(c)E(c)$

Survived E(s)		E(s) = **-[(0.417log$_2$0.417)+(0.583log$_2$0.583)]** E(s) = **-[(-0.526)+(-0.454)]** E(s) = **-[-0.97987]** E(s) = **0.97987**
Yes	**No**	
5	**7**	

Entropy(S,X) for on Ticket and Gender

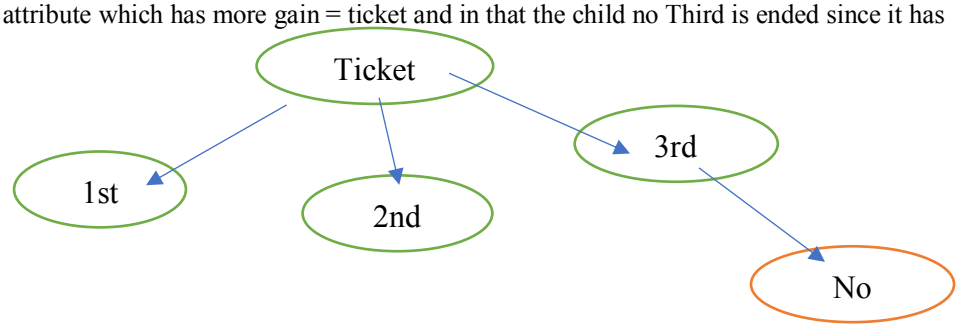
		Survived		**Gain = 0.97987 – 0.67 = 0.309**
		Yes	**No**	- **P(F)*E(3,1) + P(S)*E(2,3)+P(T)*E(0,3)** - **(4/12)*(0.81128) + (5/12)*(0.97095) + (3/12)*0** - **0.27 + 0.4 + 0** - **0.67**
Ticket	**First**	**3**	**1**	
	Second	**2**	**3**	
	Third	**0**	**3**	**Where p is for probability and E is for entropy**

		Survived		Gain = 0.97987 – 0.959 = 0.02087
		Yes	No	- P(M)*E(2,4) + P(F)*E(3,3) - 6/12 * 0.9183 + 6/12 * 1 - 0.959 Same here where p is for probability and E is for entropy
Gender	Male	2	4	
	Female	3	3	

For age which is a continuous attribute, firstly sort all values in increasing order (i.e. 8,12,20,24,26,32,34,43,45,47,55,65) and then partition each position (i.e. 4,10,16,22,25 ….) into 2 divisions which becomes a binary tree. The following calculation shows the information gain on No.3 split position, that is, 38.5.

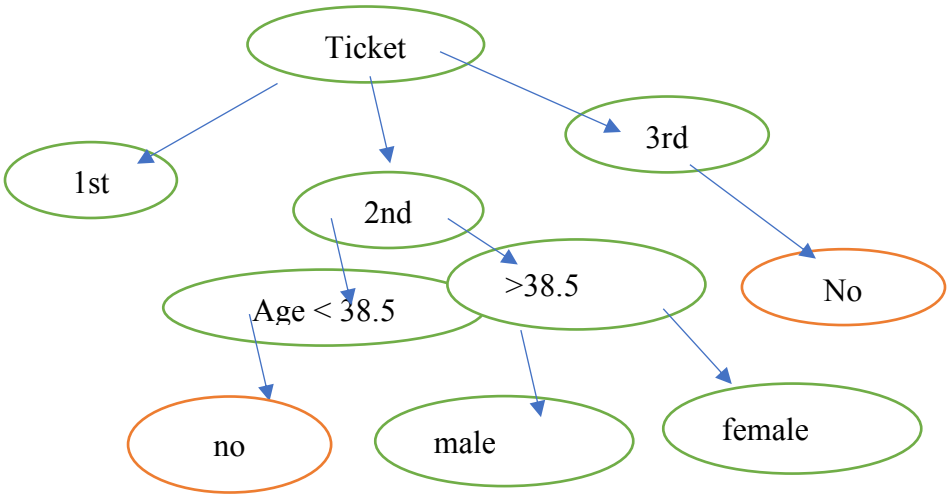
8	4	0
12	10	0
20	16	0.245
24	22	0.061
26	25	0.1685
32	29	0.0718
34	33	0.0207
43	38.5	0.0005
45	44	0.010
47	46	0.0069
55	51	0.004
65	60	0

Now we have to choose attribute which has more gain = ticket and in that the child no Third is ended since it has only No's

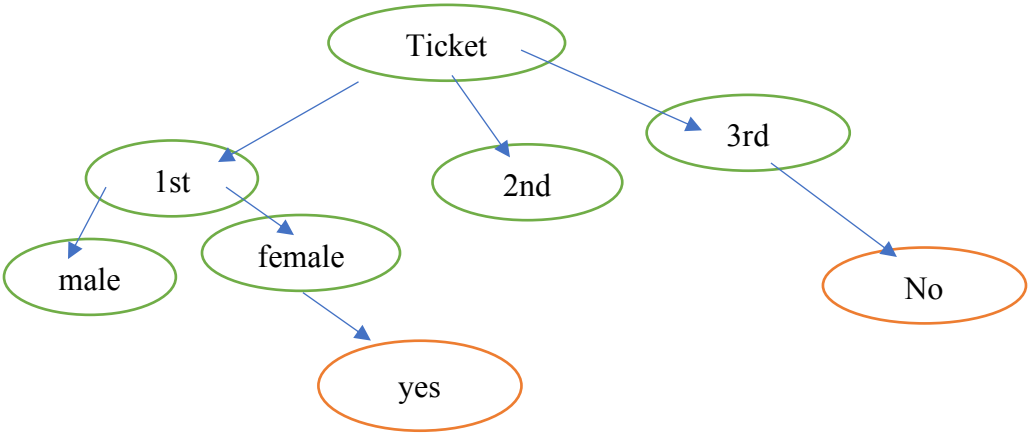


		Survived		Gain = 0.97987 – 0.95 = 0.02987
		Yes	No	- P(M)*E(1,2) + P(F)*E(1,1) - 3/5 * 0.9183 + 2/5 * 1 - 0.55 + 0.4 - 0.95 Same here where p is for probability and E is for entropy
Second/gender	Male	1	2	
	Female	1	1	

		Survived		Gain = 0.97987 – 0.36= 0.678
		Yes	No	- P(M)*E(0,2) + P(F)*E(2,1) - 3/5 * 0 + 2/5 * 0.918 - 0.3672 Same here where p is for probability and E is for entropy
Second/age	<38.5	0	2	
	>38.5	2	1	



		Survived		Gain = 0.97987 – 0.5 = 0.47989
		Yes	No	- P(M)*E(1,1) + P(F)*E(2,0) - 2/4 * 1+ 2/4 * 0 - 1/2 - 0.5 Same here where p is for probability and E is for entropy
First/gender	Male	1	1	
	Female	2	0	
		Survived		Gain = 0.97987 – 0.97095 = 0.009
		Yes	No	- P(M)*E(0,0) + P(F)*E(3,2) - 0 + 1(0.97095) - 0.97095 Same here where p is for probability and E is for entropy
First/age	<38.5	0	0	
	>38.5	3	2	



2.1 Use the gain in Gini index

Gini index = Gini(S) = 1- $\sum_{i=1}^{k} p_i^2$

Gini Gain(A,S) = Gini(S) − Gini(A, S) = Gini(S) − $\sum_{i=1}^{k} \frac{s_I}{S} \boldsymbol{Gini(S)}$

Survived E(s)		1-[(5/12)^2 + (7/12)^2] 1-(0.1736+0.34027) 0.48613
Yes	**No**	
5	**7**	

Gini gain for tickets and gender

		Survived		**Gain = 0.48613 – 0.325 = 0.16113**
		Yes	**No**	- **P(F)*G(3,1) + P(S)*G(2,3)+P(T)*G(0,3)** - **(4/12)*[1-(3/4^2 + ¼^2)] + (5/12)*[1-(2/5^2+3/5^2)] + (3/12)*[1-(0+1)]** - **1/3*(0.375) + 5/12*(0.48)** - **0.125 + 0.2** - **0.325** **Where p is for probability and G is for gini index**
Ticket	**First**	**3**	**1**	
	Second	**2**	**3**	
	Third	**0**	**3**	

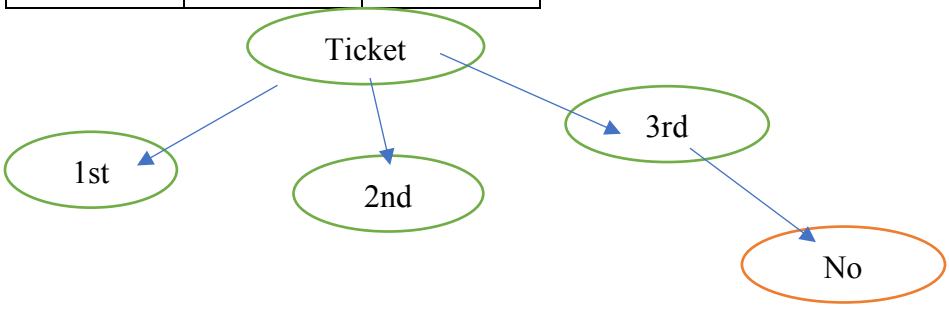
	Survived		**Gain = 0.48613 – 0.4722 = 0.01391**
	Yes	**No**	- **P(M)*G(2,4) + P(F)*G(3,3)**

Gender	**Male**	**2**	**4**	- **6/12 * [0.4444] + 6/12 * [0.5]** - **0.472222**
	Female	**3**	**3**	**Same here where p is for probability and G is for gini index**

According to the gini gain, the best split between tickets and gender is tickets due to it has the higher gini gain.

For age which is a continuous attribute, firstly sort all values in increasing order (i.e. 8,12,20,24,26,32,34,43,45,47,55,65) and then partition each position (i.e. 4,10,16,22,25 ….) into 2 divisions which becomes a binary tree. The following calculation shows the gini gain at 38.5

8	4	0
12	10	0.061
20	16	0.136
24	22	0.041
26	25	0.111
32	29	0.048
34	33	0.01388
43	38.5	0.0003
45	44	0.006
47	46	0.0046
55	51	0.0027
65	60	0.0315



1.3. Using the final tree that you have constructed for Question (1.1), compute the generalization error rate of the tree using the pessimistic approach assuming that the penalty term associated with each leaf node is 0.5.

The penality for each node is 0.5

generalization error (T) = sum(e(ti) + Ω(ti)) / sum(n)*ti =
= (e(T)+ Ω(T))/N

Total no of records = 12
Penality Ω cost -> Ω(T) 0.5
Total penality : 0.5 * 4 = 2
E(T): classification error = 2 (at two points I cannot do classification.)

E'(T) = (2+2) /12 = 0.3

Problem 2 (70 points):

2.1: C5.0 and CART are two well-known decision tree algorithms

C5.0

overview describing how the algorithm works

C5.0 algorithm is a successor of C4.5 algorithm also developed by Quinlan (1994)
This node uses the C5.0 algorithm to build either a **decision tree** or a **rule set (**Gives a binary tree or multi branches tree **)**. A C5.0 model works by splitting the sample based on the field that provides the maximum **information gain which is calculated using entropy**. Each subsample defined by the first split is then split again, usually based on a different field, and the process repeats until the subsamples cannot be split any further. Finally, the lowest-level splits are reexamined, and those that do not contribute significantly to the value of the model are removed or **pruned**.

- A **decision tree** is a straightforward description of the splits found by the algorithm. Each terminal (or "leaf") node describes a particular subset of the training data, and each case in the training data belongs to exactly one terminal node in the tree. In other words, exactly one prediction is possible for any particular data record presented to a decision tree.
- On other side a **rule set** is a set of rules that tries to make predictions for individual records. Rule sets are derived from decision trees and, in a way, represent a simplified or distilled version of the information found in the decision tree.

Requirements: To train using this algorithm, there must be one categorical (i.e., nominal or ordinal) *Target* field, and one or more *Input* fields of any type. Fields set to *Both* or *None* are ignored. Fields used in the model must have their types fully instantiated. A weight field can also be specified.

discuss the impurity measure it uses for the attribute split test condition

The first thing the algorithm has to do is decide which feature to split upon (since we may well have many). As described earlier, this is done so as to increase the information gain. The gain is defined as the difference between a parent's impurity and the sum of its children's impurities. There are a number of different impurity measures, entropy being the one favored in the implementation C5.0. The formula for entropy is given by:

$$\text{Entropy}(S) = \sum_{i \text{ to } c} - p_i \log_2(p_i)$$

where

S is a segment of data (i.e. a bunch of records),
c is the number of different class levels, and
pi refers to the proportion of values falling into class level ii.

discuss one advantage and one disadvantage of the algorithm

Advantage. C5.0 models are quite robust in the presence of problems such as missing data and large numbers of input fields. They usually do not require long training times to estimate. In addition, C5.0 models tend to be easier to understand than some other model types, since the rules derived from the model have a very straightforward interpretation. C5.0 also offers the powerful **boosting** method to increase accuracy of classification. In a case of handling missing values, C5.0 allows to whether estimate missing values as a function of other attributes or apportions the case statistically among the results.

Disadvantage

A small variation in data can lead to different decision trees and may not work well for small datasets.

references to the published literature

https://www.ibm.com/support/knowledgecenter/en/SS3RA7_15.0.0/com.ibm.spss.modeler.help/c50node_general.htm
https://www.sciencedirect.com/topics/computer-science/decision-tree-algorithm
https://www.ijert.org/research/comparison-of-c5.0-cart-classification-algorithms-using-pruning-technique-IJERTV1IS4104.pdf
https://rpubs.com/mzc/mlwr_dtr_credit_mushrooms
https://pdfs.semanticscholar.org/fd39/e1fa85e5b3fd2b0d000230f6f8bc9dc694ae.pdf

CART

overview describing how the algorithm works

ID3 uses information gain whereas C4.5 uses gain ratio for splitting. Here, CART is an alternative decision tree building algorithm. It can handle both classification and regression tasks. This algorithm uses a new metric named gini index to create decision points for classification tasks

Gini index calculation for each class and choose which has less:

$Gini = 1 - \Sigma (Pi)^2$ for i=1 to number of classes

Now apply the same principle for the sub datasets, like with root and its children value. This process will be repeated until we reach to leaf node.

CART can handle both nominal and numeric attributes to construct a decision tree.
CART uses Cost – Complexity Pruning to remove redundant branches from the decision tree to improve the accuracy.
CART handles missing values by surrogating tests to approximate outcomes

discuss the impurity measure it uses for the attribute split test condition

Gini Index is an attribute selection measure used by the CART decision tree algorithm. The Gini Index measures the impurity D, a data partition or set of training tuples as:

$Gini = 1 - \Sigma (Pi)^2$ for i=1 to number of classes

Where pi is the probability that a tuple in D belongs to class Ci and is estimated by |Ci,D|/|D|. The sum is computed over m classes. The attribute that reduces the impurity to the maximum level (or has the minimum gini index) is selected as the splitting attribute.

Discuss one advantage and one disadvantage of the algorithm

Advantages:

- CART handles missing values automatically Using “surrogate splits”
- Invariant to monotonic transformations of predictive variable
- Not sensitive to outliers in *predictive* variables Unlike regression
- Great way to explore, visualize data

Disadvantages:

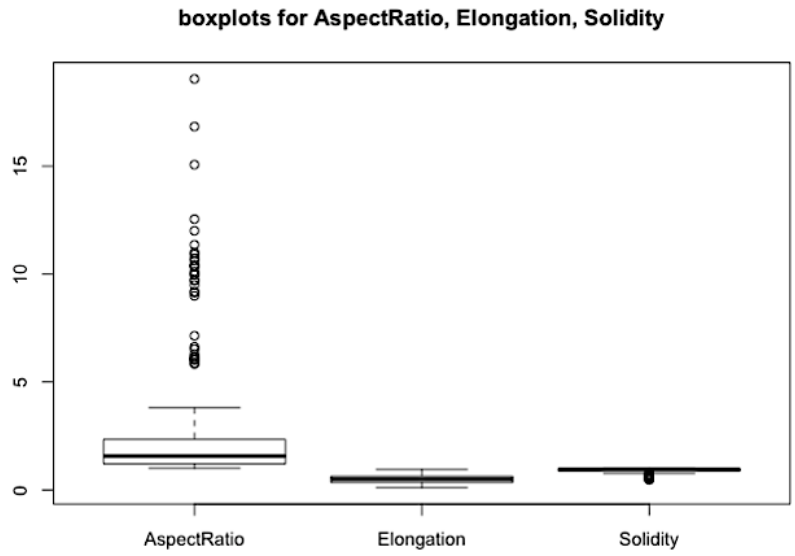
- The model is a *step function*, not a continuous score. So if a tree has 10 nodes, that can only take on 10 possible values.
- Might take a large tree to get good lift
- Instability of model structure like Correlated variables random data fluctuations could result in entirely different trees.
- Splits only on one variable

references to the published literature

https://www.sciencedirect.com/topics/computer-science/decision-trees
https://cds.cern.ch/record/2253780
https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/
http://mercury.webster.edu/aleshunas/Support%20Materials/C4.5/Nguyen-Presentation%20Data%20mining.pdf
https://pdfs.semanticscholar.org/fd39/e1fa85e5b3fd2b0d000230f6f8bc9dc694ae.pdf

2.2. Write an R program to perform the following tasks

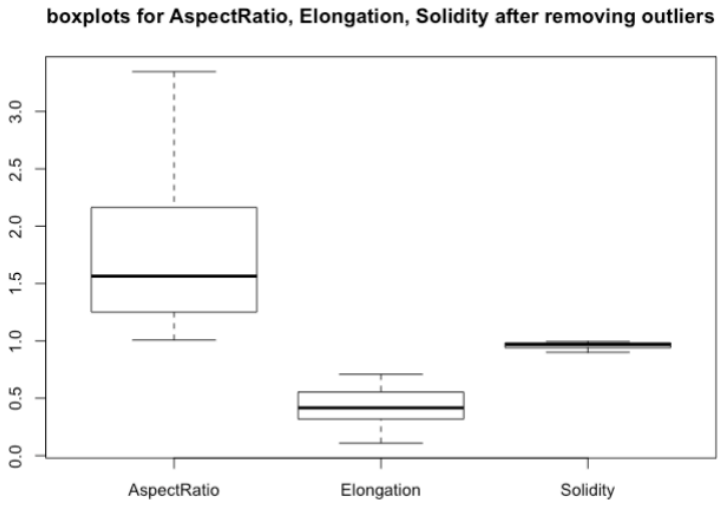
a. Aspect Ratio, Elongation, and Solidity BoxPlots



b. Which attribute has outliers, what are those values, and remove those rows

- Among those three attributes Aspect Ratio and solidity has outliers. And those values are above 5 and below 0.9 respectively. (for solidity I may not visible in 1st glance. On zooming we can see outliers)
- So lets remove these outliers where Aspect Ratio will be >= 5 and <= 0.9
- `data<-data[!(data$`Aspect Ratio` >= 5 | data$Solidity <= 0.9),] # code to remove outliers`

c. Boxplot after removing outliers



d. C5.0 decision tree in both textual, graphical format and error evaluation using k fold cross validation

```
-   library(C50)
-   tree_model <- C50::C5.0(x = data[, -c(1)], y = as.factor(data$Class)) #
    as.factor to give input in factors since c5.0 is only accepting that
-   tree_model
-   summary(tree_model) # printing in textual format
-   plot(tree_model) # printing in graphical format

# model validation on the same data

-   c50.prediction <- predict(tree_C50, newdata=data, type="class")
-   confusion.matrix <- table(data$Class, c50.prediction)
-   print(confusion.matrix)
-   accuracy.percent <- 100*sum(diag(confusion.matrix))/sum(confusion.matrix)
-   print(paste("accuracy:",accuracy.percent,"%"))
```

```
> print(paste("accuracy:",accuracy.percent,"%"))
[1] "accuracy: 89.2241379310345 %"
```

From Reference website: link
K-fold cross-validation is used for determining the performance of statistical models. How it works is the data is divided into a predetermined number of folds (called 'k'). One fold is used to determine the model estimates and the other folds are used for evaluating. This is done k times and the results are average based on a statistic such as kappa to see how the model performs.

```
library(caret)
library(plyr)
folds<-createFolds(as.factor(data$Class), k=3)
str(folds)
errs.c50 <- rep(NA, length(folds)) # to store all the folds error.
form <- "as.factor(Class) ~ ."
for (i in 1:length(folds)) {
  test <- data[unlist(folds[i]),]
  train <- data[unlist(folds[-i]),]
  tmp.model <- C50::C5.0(as.formula(form),train)
  tmp.predict <- predict(tmp.model, newdata=test)
  conf.mat <- table(test$Class, tmp.predict)
```

```
  print(conf.mat)
  errs.c50[i] <- 1 - sum(diag(conf.mat))/sum(conf.mat)
  print(sprintf("error using k-fold cross validation %.0f and C5.0 decision
tree algorithm: %.3f percent", i,100*(errs.c50[i])))
}
```

```
> errs.c50
[1] 0.5394737 0.4683544 0.4805195
```

e. CART decision tree in both textual, graphical format and error evaluation using k fold cross validation

```
library(rpart)
tree_CART <- rpart(as.factor(Class) ~ ., data = data, method="class") #
as.factor to give input in factors since c5.0 is only accepting that
tree_CART
summary(tree_CART) # printing in textual format
plot(tree_CART)
text(tree_CART, use.n=TRUE, all=TRUE, cex=.8)

# model validation on the same data

cart.prediction <- predict(tree_CART, newdata=data, type="class")
confusion.matrix <- table(data$Class, cart.prediction)
print(confusion.matrix)
accuracy.percent <- 100*sum(diag(confusion.matrix))/sum(confusion.matrix)
print(paste("accuracy:",accuracy.percent,"%"))



# cross validation for cart

errs.cart <- rep(NA, length(folds)) # to store all the folds error.
for (i in 1:length(folds)) {
  test <- data[unlist(folds[i]),]
  train <- data[unlist(folds[-i]),]
  tmp.model <- rpart(form , train, method = "class")
  tmp.predict <- predict(tmp.model, newdata=test, type = 'class')
  conf.mat <- table(test$Class, tmp.predict)
  print(conf.mat)
  errs.cart[i] <- 1 - sum(diag(conf.mat))/sum(conf.mat)
  print(sprintf("error using k-fold cross validation %.0f and CART decision
tree algorithm: %.3f percent", i,100*(errs.cart[i])))
}
```

```
> errs.cart
[1] 0.5657895 0.5696203 0.5974026
```

f. Use hypothesis testing to determine to whether or not the error rate difference between the two classification algorithms is statistically significant given the confidence level of 98%.

```
> h <- t.test(errs.cart, errs.c50, alternative = 'two.sided',conf.level =
0.98,conf.int = TRUE, paired = TRUE)
```

```
> h <- t.test(errs.cart, errs.c50, alternative = 'two.sided',conf.level = 0.98,conf.int = TRUE, paired = TRUE)
> h

        Paired t-test

data:  errs.cart and errs.c50
t = 2.9153, df = 2, p-value = 0.1003
alternative hypothesis: true difference in means is not equal to 0
98 percent confidence interval:
 -0.1131864  0.2761629
sample estimates:
mean of the differences
             0.08148824
```

Here the p value we got is 0.1003. and interval [-0.11 to 0.27] where the error mean difference 0.081 lies in.

- If the p-value is greater than to the significance level 0.02, we can accept the null hypothesis and reject the alternative hypothesis. In other words, we can conclude that the mean values of group A and B are significantly equal.

g. Predict class for new label for at least three tuples

```
newtuple <-
c(NA,1,0.72694,1.4742,0.32396,0.98535,1,0.83592,0.0046566,0.0039465,0.04779,0.1
2795,0.016108,0.0052323,0.00027477,1.1756)
lev <- c(1,2,3,4,7,9,10,12,13,14,22,23,24,25,26,27,28,29,30,32,33,35)
df <- data.frame(matrix(newtuple, nrow=1))
names(df) <- headers
pre <- predict(tree_C50, newdata=df[,-1])
cat("the predicted class for given data is ",lev[pre])
```

```
the predicted class for given data is  13
```

```
newtuple <- as.numeric(as.vector(data[200,])) # taking from exisitng data only
lev <- c(1,2,3,4,7,9,10,12,13,14,22,23,24,25,26,27,28,29,30,32,33,35)
df <- data.frame(matrix(newtuple, nrow=1))
names(df) <- headers
pre <- predict(tree_C50, newdata=df[,-1])
cat("the predicted class for given data is ",lev[pre])
```

```
the predicted class for given data is  32
```

```
newtuple <- as.numeric(as.vector(data[150,])) # taking from exisitng data only
lev <- c(1,2,3,4,7,9,10,12,13,14,22,23,24,25,26,27,28,29,30,32,33,35)
df <- data.frame(matrix(newtuple, nrow=1))
names(df) <- headers
pre <- predict(tree_C50, newdata=df[,-1])
cat("the predicted class for given data is ",lev[pre])
```

```
the predicted class for given data is  26
```