Classification And Regression Trees for Machine Learning

by **Jason Brownlee** on April 8, 2016 in **Understand Machine Learning Algorithms**

Decision Trees are an important type of algorithm for predictive modeling machine learning.

The classical decision tree algorithms have been around for decades and modern variations like random forest are among the most powerful techniques available.

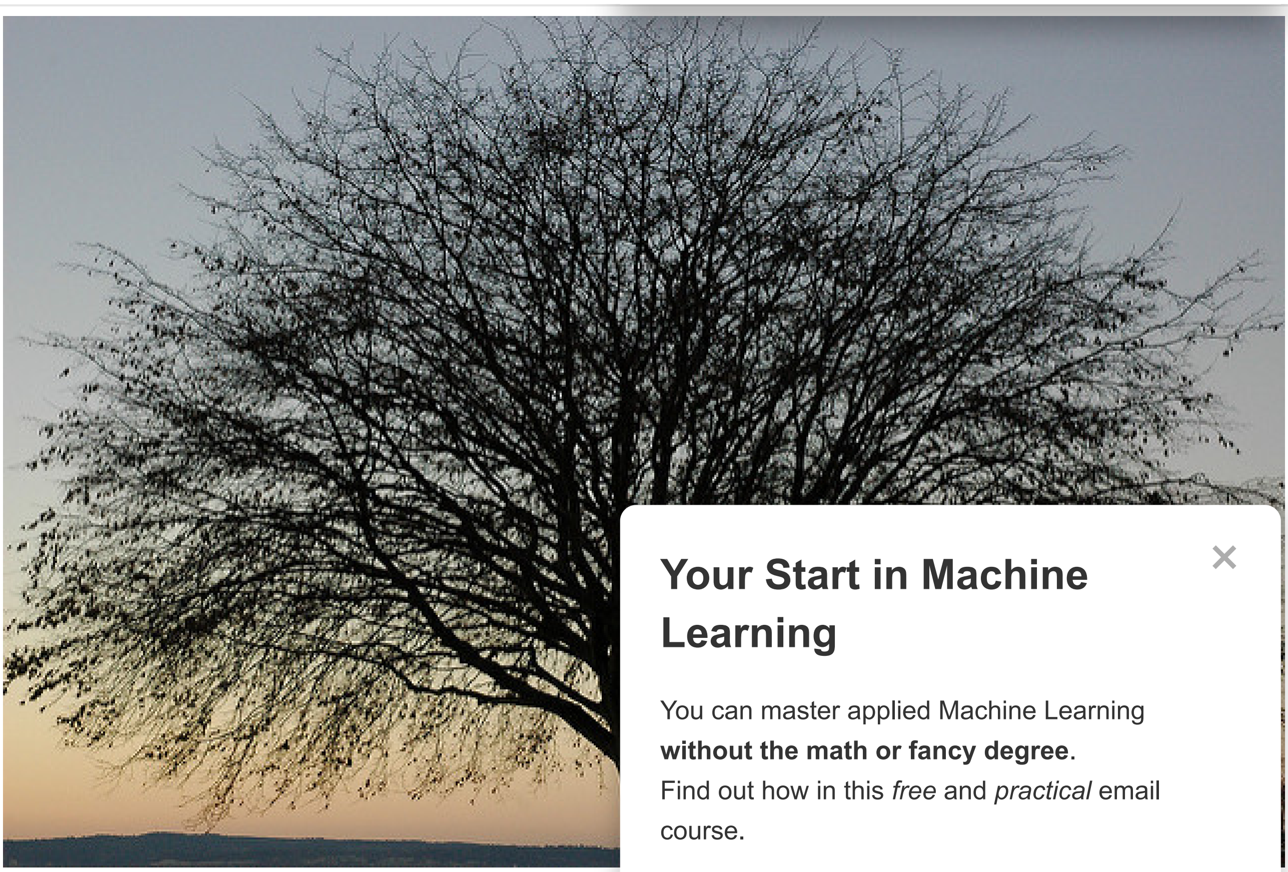
In this post you will discover the humble decision tree algorithm known by it's more modern name CART which stands for Classification And Regression Trees. After reading this post, you will know:

- The many names used to describe the CART algorithm for machine learning.
- The representation used by learned CART models that is actually stored on disk.
- How a CART model can be learned from training data.
- How a learned CART model can be used to make predictions on unseen data.
- Additional resources that you can use to learn more about CART and related algorithms.

If you have taken an algorithms and data structures course, it might be hard to hold you back from implementing this simple and powerful algorithm. And from there, you're a small step away from your own implementation of Random Forests.

Let's get started.

- **Update Aug 2017**: Fixed a typo that indicated that Gini is the count of instances for a class, should have been the proportion of instances. Also updated to show Gini weighting for evaluating the split in addition to calculating purity for child nodes.



Classification And Regress
Photo by Wonderlan

Decision Trees

Classification and Regression Trees or CART for short is a term introduced by Leo Breiman to refer to Decision Tree algorithms that can be used for classification or regression predictive modeling problems.

Classically, this algorithm is referred to as "decision trees", but on some platforms like R they are referred to by the more modern term CART.

The CART algorithm provides a foundation for important algorithms like bagged decision trees, random forest and boosted decision trees.

Sample of the handy machine learning algorithms mind map.

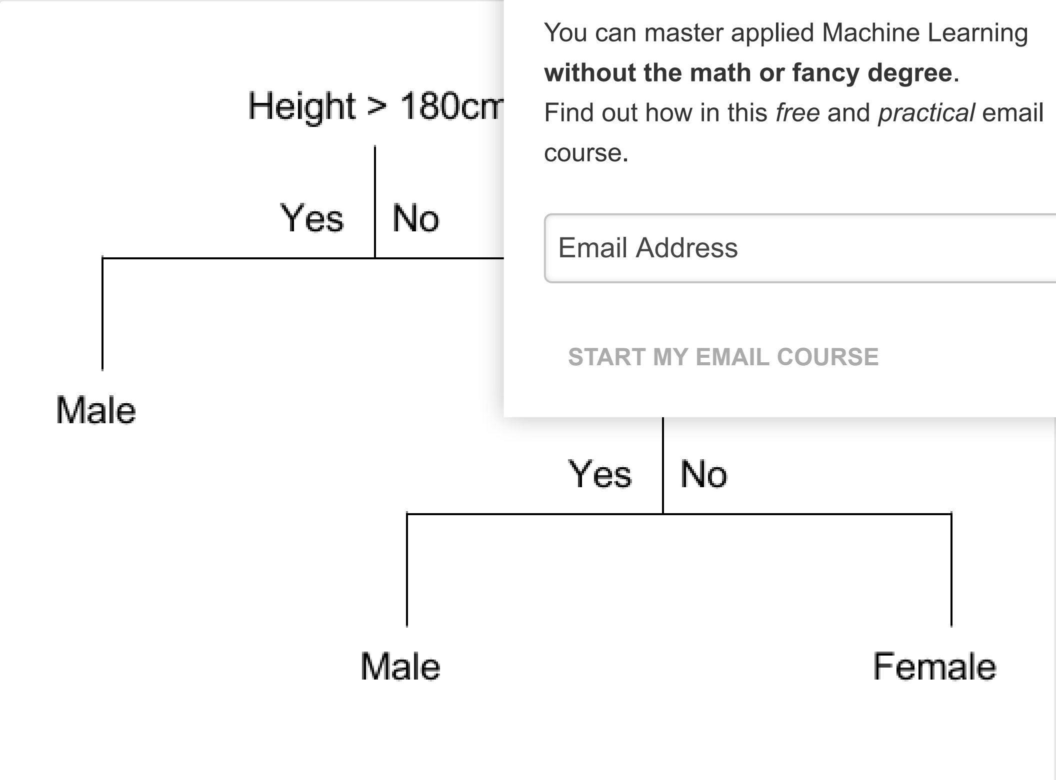
CART Model Representation

The representation for the CART model is a binary tree.

This is your binary tree from algorithms and data structures, nothing too fancy. Each root node represents a single input variable (x) and a split point on that variable (assuming the variable is numeric).

The leaf nodes of the tree contain an output variable (y) which is used to make a prediction.

Given a dataset with two inputs (x) of height in cen
as male or female, below is a crude example of a b
demonstration purposes only).



Example Decision Tree

The tree can be stored to file as a graph or a set of rules. For example, below is the above decision tree as a set of rules.

```
If Height > 180 cm Then Male
If Height <= 180 cm AND Weight > 80 kg Then Male
If Height <= 180 cm AND Weight <= 80 kg Then Female
Make Predictions With CART Models
```

With the binary tree representation of the CART model described above, making predictions is relatively straightforward.

Given a new input, the tree is traversed by evaluating the specific input started at the root node of the tree.

A learned binary tree is actually a partitioning of the input space. You can think of each input variable as a dimension on a p-dimensional space. The decision tree split this up into rectangles (when p=2 input variables) or some kind of hyper-rectangles with more inputs.

New data is filtered through the tree and lands in one of the rectangles and the output value for that rectangle is the prediction made by the model. This gives you some feeling for the type of decisions that a CART model is capable of making, e.g. boxy decision boundaries.

For example, given the input of [height = 160 cm, weight = 65 kg], we would traverse the above tree as follows:

```
Height > 180 cm: No
Weight > 80 kg: No
Therefore: Female
```


Learn a CART Model From Da

Creating a CART model involves selecting input va suitable tree is constructed.

The selection of which input variable to use and th dy algorithm to minimize a cost function. Tree constru uch as a minimum number of training instances assign

Greedy Splitting

Creating a binary decision tree is actually a proces ch is used to divide the space called recursive binary splitting.

This is a numerical procedure where all the values are lined up and different split points are tried and tested using a cost function. The split with the best cost (lowest cost because we minimize cost) is selected.

All input variables and all possible split points are evaluated and chosen in a greedy manner (e.g. the very best split point is chosen each time).

For regression predictive modeling problems the cost function that is minimized to choose split points is the sum squared error across all training samples that fall within the rectangle:

sum(y – prediction)^2

Where y is the output for the training sample and prediction is the predicted output for the rectangle.

For classification the Gini index function is used which provides an indication of how “pure” the leaf nodes are (how mixed the training data assigned to each node is).

G = sum(pk * (1 – pk))

Where G is the Gini index over all classes, pk are the proportion of training instances with class k in the rectangle of interest. A node that has all classes of), where as a G that has a 50-50 split of classes for a binary classification problem (worst purity) will have a G=0.5.

For a binary classification problem, this can be re-written as:

$$G = 2 * p1 * p2$$

or

$$G = 1 – (p1^2 + p2^2)$$

The Gini index calculation for each node is weighted by the total number of instances in the parent node. The Gini score for a chosen split point in a binary classification problem is therefore calculated as follows:

G = ((1 – (g1_1^2 + g1_2^2)) * (ng1/

Where G is the Gini index for the split point, g1_1 i ... , g1_2 for class 2, g2_1 for group 2 and class 1, g2_ ... ber of instances in group 1 and 2 and n are the total nu ... parent node.

Stopping Criterion

The recursive binary splitting procedure described ... works its way down the tree with the training data.

The most common stopping procedure is to use a ... es assigned to each leaf node. If the count is less than some minimum then the split is not accepted and the node is taken as a final leaf node.

The count of training members is tuned to the dataset, e.g. 5 or 10. It defines how specific to the training data the tree will be. Too specific (e.g. a count of 1) and the tree will overfit the training data and likely have poor performance on the test set.

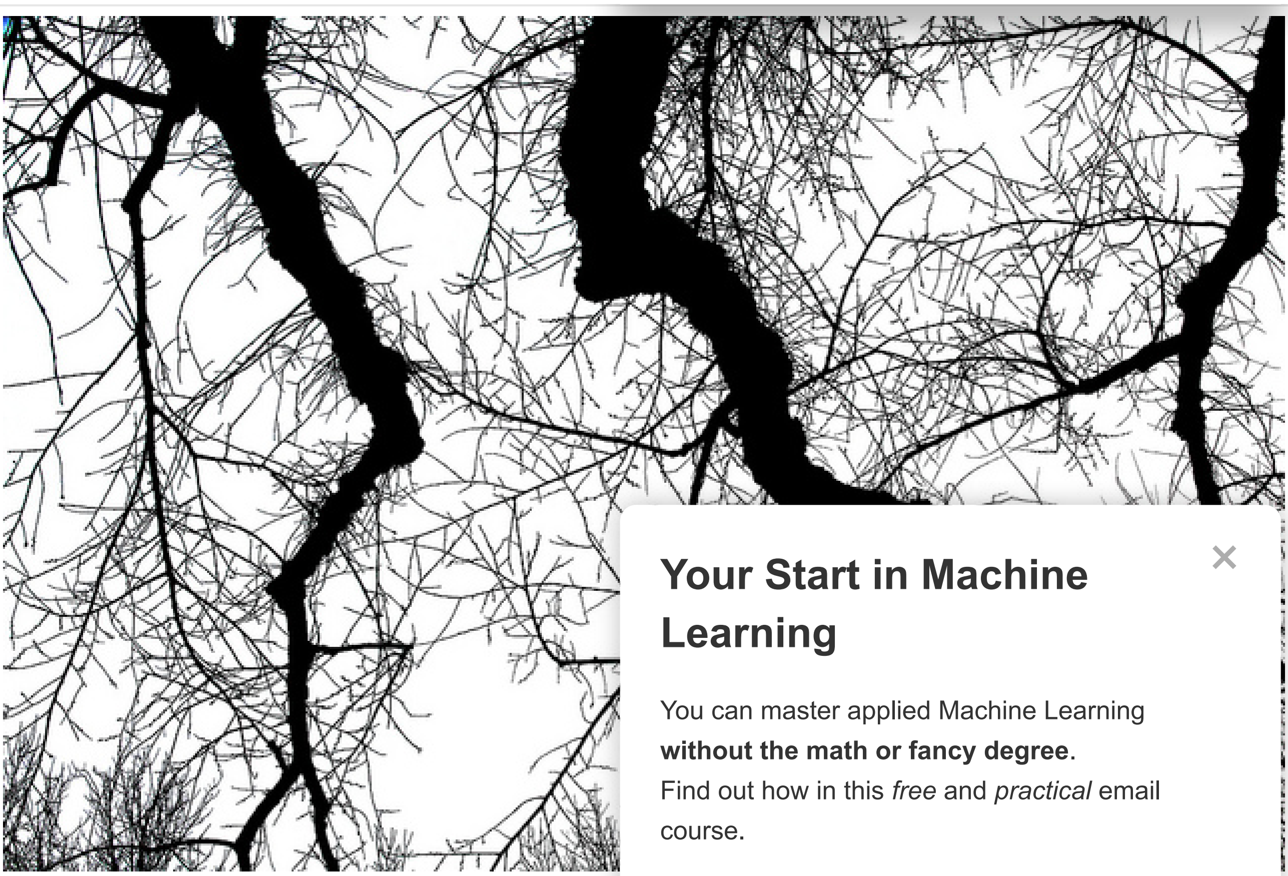
Pruning The Tree

The stopping criterion is important as it strongly influences the performance of your tree. You can use pruning after learning your tree to further lift performance.

The complexity of a decision tree is defined as the number of splits in the tree. Simpler trees are preferred. They are easy to understand (you can print them out and show them to subject matter experts), and they are less likely to overfit your data.

The fastest and simplest pruning method is to work through each leaf node in the tree and evaluate the effect of removing it using a hold-out test set. Leaf nodes are removed only if it results in a drop in the overall cost function on the entire test set. You stop removing nodes when no further improvements can be made.

More sophisticated pruning methods can be used such as cost complexity pruning (also called weakest link pruning) where a learning parameter (alpha) is ... based on the size of the sub-tree.



Recursive Binary Sp
Photo by Paul L Dine

Data Preparation for CART

CART does not require any special data preparation other than a good representation of the problem.

Further Reading

This section lists some resources that you can refer to if you are looking to go deeper with CART.

- Classification and Regression Trees

Below are some good machine learning texts that describe the CART algorithm from a machine learning perspective.

- An Introduction to Statistical Learning: with Applications in R, Chapter 8
- Applied Predictive Modeling, Chapter 8 and 14
- Data Mining: Practical Machine Learning Tools and Techniques, chapter 6.

Summary

In this post you have discovered the Classification And Regression Trees (CART) for machine learning. You learned:

- The classical name Decision Tree and the more Modern name CART for the algorithm.
- The representation used for CART is a binary tree.
- Predictions are made with CART by traversing

- The tree is learned using a greedy algorithm on the training data to pick splits in the tree.
- Stopping criteria define how much tree learns and pruning can be used to improve a learned tree.

Do you have any questions about CART or this post?
Ask in the comments and I will do my best to answer.

About Jason Brownlee

Jason Brownlee, PhD is a machine learning specialist who teaches developers how to get results with modern machine learning methods via hands-on tutorials.
View all posts by Jason Brownlee →

‹ Deep Learning Books Naive Bayes for Machine Learning ›

---

47 Responses to *Classification And Regression Trees for Machine Learning*

Audio Alief Kautsar Hartama April 27, 2016 at 10:39 pm # REPLY how about C 4.5 (which is called J48 in weka) and C 5.0, please make tutorial for that sir, i need it

Yogesh June 10, 2016 at 3:52 pm # REPLY

I have one sentence amd its polarity -ve or +ve, I want use CART for accuricy.But I am not able to understand how?

joe August 29, 2016 at 5:46 pm # REPLY

is it possible to infuse CART in GA?

Jason Brownlee August 30, 2016 at 8:24

No idea Joe.

Mynose September 8, 2016 at 5:20 pm #

Sir, i am wanting to compare CART and G

Jason Brownlee September 9, 2016 at 7:18 am # REPLY

Hi Mynose, they are very different. CART is a function approximation method and a GA is a function optimization method.

I guess you could use the GA to optimize a set of rules or a tree and compare that to the CART. Sounds fun.

Lee November 2, 2016 at 4:48 pm # REPLY

Hi, Jason! How can I avoid over-fitting problem when using a CART model. When I used a CART tree to classify different fault types of data, the cp is the only parameter for obtained a optimal CART model. But the tree structure of the training model is obviously over-fitted from my domain knowledge. So what should I do to avoid overfitting?

Jason Brownlee November 3, 2016 at 7:54 am # REPLY

Hi Lee, great question.

The main approach is to constrain the depth of the tree.

You can do this when growing the tree, but the preferred method is to prune a deep tree after it is constructed.

Rohit November 10, 2016 at 4:36 am # REPLY

Hello Sir,

How is the variable selection of input variables done while implementing the greedy algorithm . For calculating the minimum cost function you need the predicted values , but how does the algorithm select the variable from input variable for the first split.

Regards,
Rohit

Jason Brownlee November 10, 2016 at 7

Hi Rohit, we have the predicted values data.

Splits are fixed after training, then we just use t

madhav January 31, 2017 at 4:31 pm # REPLY

Sir, i have following questions. It would be of great help if you could answer them for me.

1)Is CART the modern name for decision tree approach in Data Science field.
2)What are the scenarios where CART can be used.
3)what are the advantages of using CART over other techniques of predicition.

Jason Brownlee February 1, 2017 at 10:45 am # REPLY

Hi madhav,

1. Yes, CART or classification and regression trees is the modern name for the standard decision tree.
2. Very widely on classification and regression predictive modeling problems. Try it and see.
3. Fast to train, easy to understand result and generally quite effective.

Farina March 29, 2017 at 4:17 pm # REPLY

Hi Jason

I am wondering why my CART produced only one nodes when I exclude one variable for example ID? I tried to change the cp but it is still giving the same results. Can you assist me on this?

Winston Jade Molit April 23, 2017 at 5:10 pm # REPLY

Is CART algorithm appropriate for decision making projects?

Jason Brownlee April 24, 2017 at 5:33 am # REPLY

That depends if the decision can be fr

Luc May 12, 2017 at 10:41 pm #

Hi Jason,

I am new into machine learning. For an intership it
method. The input variables are a small number of
Is it possible to apply CART for this problem? I am
different algorithms are used, do you have tips?

Jason Brownlee May 13, 2017 at 6:14 am # REPLY

I would recommend that you follow this process:
http://machinelearningmastery.com/start-here/#process

Radhakrishna July 10, 2017 at 1:28 am # REPLY

Hi Jason,

I would like to know what parameters to change in CART, CHAID and QUEST decision tree algorithms for effective modeling.

Jason Brownlee July 11, 2017 at 10:20 am # REPLY

Sorry I do not have this information.

Marco July 13, 2017 at 7:32 am # REPLY

Hi Jason,
I didn't understand how the algorithm selects the input variables for the splits. In your example, why was the height split before weight? Thank you.

Jason Brownlee July 13, 2017 at 10:04 am # REPLY

It was just an example.

Amine July 18, 2017 at 9:16 pm # REPLY

Hi Jason,
I'am working on a highly unbalanced data, I have 4 prune the tree using rpart package. Using X-Var re exactly one node. Is it possible to change X-Var rel another one) that takes more on consideration the

Thank you !!

Jason Brownlee July 19, 2017 at 8:23 ar

Ouch

Sorry I am not familiar with that package.

Taylor Nelson July 19, 2017 at 6:57 am # REPLY

Hi Jason,

I haven't seen many examples of decision trees used for regression, just for classification. Have any favorite examples you know of, or will you do a post on that? It would be interesting to talk about the difference between OLS and other linear regression methods methinks.

Thanks!

Jason Brownlee July 19, 2017 at 8:31 am # REPLY

Here are some examples:
http://machinelearningmastery.com/non-linear-regression-in-r-with-decision-trees/

Use the search feature on the blog.

Pia Laine August 9, 2017 at 1:38 am #

REPLY

Hi,

just a little remark about the Gini function – I think there is a typo:

G = sum(pk * (1 – pk))
-> G = sum(pk/p * (1 – pk/p)), where p is the total number of instances in the rectangle.

As we seem to be looking at the relative portions of instances per class.

Jason Brownlee August 9, 2017 at 6:40

Thanks Pia, I'll investigate.

Yes, it's a typo. Fixed. Thank you!

Sthembiso August 16, 2017 at 5:24 pm #

Hi! Jason Brownlee

Could you help me with this question, i'm new on m

You are a junior data scientist within Standard Ban
tasked to explain to the Investment Bankers how d
can assist them in running their day to day activities.
The investment bankers receive a lot of information on a daily basis from internal and external sources such as journals, newsfeeds, macro-economic data, company financials to name but a few. They use this information to assess where the next big deals are likely to emanate from and prioritise those opportunities which they perceive to have the highest chance of materialisation. They also take into account factors such as:
I. Value of the deal
II. Potential commission
III. Presence of Standard Bank in country where deal is taking place
IV. Type of deal (merger, acquisition, equity deal etc.)
V. Credit ratings of companies involved in deal
VI. Geographical region
VII. Industrial Sector (e.g. Agriculture, Tourism, Financial Service etc.)

Please note that deals occur few and far between.
You then decide to showcase to them the power of Decision trees and how they can be used to evaluate all potential deals. Using the information above:

1. Explain the steps in making a decision tree and how they can be applied to this business challenge.

Jason Brownlee August 17, 2017 at 6:37 am #

REPLY

This looks like homework, I would rec

venkat September 11, 2017 at 4:25 am #

REPLY

#rm(list=ls(all=TRUE))
setwd("C:\\Users\\hp\\Desktop\\R")
version
#Reading from a CSV file
univ=read.table('dataDemographics.csv',
header=T,sep=',',
col.names=c("ID", "age", "exp", "inc",
"zip", "family",
"edu", "mortgage"))
dim(univ)
head(univ)
str(univ)
names(univ)
sum(is.na(univ))
sum(is.na(univ[[2]])) #see missig values in col 2
sapply(univ, function(x) sum(is.na(x)))
row.names.data.frame(is.na(univ))
# Reading Second Table
loanCalls <- read.table("dataLoanCalls.csv", heade
col.names=c("ID", "infoReq", "loan"),
dec=".", na.strings="NA")
head(loanCalls)
dim(loanCalls)

sum(is.na(loanCalls))
sapply(loanCalls, function(x) sum(is.na(x)))

# Reading third Table
cc <- read.table("dataCC.csv", header=T, sep=",",
col.names=c("ID", "Month", "Monthly"),
dec=".", na.strings="NA")
head(cc)
dim(cc)
sum(is.na(cc))
sapply(cc, function(x)sum(is.na(x)))

#We have the monthly credit card spending over 12 months.

#We need to compute monthly spendings

tapply
head(cc)
summary(cc)
str(cc)
cc$ID <- as.factor(cc$ID)
cc$Month <- as.factor(cc$Month)

```
sapply(cc,function(x) length(unique(x)))
summary(cc)

# function to cal. mean
meanNA <- function(x){
a <-mean(x, na.rm=TRUE)

return(a)
}

ccAvg <- data.frame(seq(1,5000),
tapply(cc$Monthly, cc$ID, meanNA))
ccAvg
head(ccAvg)
dim(ccAvg)
names(ccAvg)
colnames(ccAvg) <- c("ID", "ccavg")
str(ccAvg)
ccAvg$ID <- as.factor(ccAvg$ID)
summary(ccAvg)
str(ccAvg)
rm(cc)

# Reading fourth table
otherAccts <- read.table("dataOtherAccts.csv", hea
col.names=c("ID", "Var", "Val"),
dec=".", na.strings="NA")
dim(otherAccts)
head(otherAccts)
summary(otherAccts)
otherAccts$ID <- as.factor(otherAccts$ID)
otherAccts$Val <- as.factor(otherAccts$Val)
summary(otherAccts)
str(otherAccts)

# to transpose
library(reshape)
otherAcctsT=data.frame(cast(otherAccts,
ID~Var,value="Val"))
head(otherAcctsT)
dim(otherAcctsT)

#Merging the tables
univComp <- merge(univ,ccAvg,
by.x="ID",by.y="ID",
all=TRUE) #Outer join

univComp <- merge(univComp, otherAcctsT,
by.x="ID", by.y="ID",
all=TRUE)

univComp <- merge(univComp, loanCalls,
by.x="ID", by.y="ID",
all=TRUE)
```


Your Start in Machine Learning

```
dim(univComp)
head(univComp)
str(univComp)
summary(univComp)
names(univComp)
sum(is.na(univComp))

#Dealing with missing values
#install.packages("VIM")
library(VIM)
matrixplot(univComp)

#Filling up missing values with KNNimputation
library(DMwR)
univ2 <- knnImputation(univComp,
k = 10, meth = "median")
sum(is.na(univ2))
summary(univ2)
head(univ2,10)
univ2$family <- ceiling(univ2$family)
univ2$edu <- ceiling(univ2$edu)
head(univ2,15)

str(univ2)
names(univ2)
# converting ID, Family, Edu, loan into factor
attach(univ2)
univ2$ID <- as.factor(ID)
univ2$family <- as.factor(family)
univ2$edu <- as.factor(edu)
univ2$loan <- as.factor(loan)
str(univ2)
summary(univ2)
sapply(univ2, function(x) length(unique(x)))

# removing the id, Zip and experience as experience
# is correlated to age
names(univ2)
univ2Num <- subset(univ2, select=c(2,3,4,8,9))
head(univ2Num)
cor(univ2Num)

names(univ2)
univ2 <- univ2[,-c(1,3,5)]
str(univ2)
summary(univ2)

# Converting the categorical variables into factors
# Discretizing age and income into categorial variables
library(infotheo)

#Discretizing the variable 'age'
age <- discretize(univ2$age, disc="equalfreq",
```

```
nbins=10)
class(age)
head(age)
age=as.factor(age$X)

#Discretizing the variable 'inc'
inc=discretize(univ2$inc, disc="equalfreq",
nbins=10)
head(inc)
inc=as.factor(inc$X)

#Discretizing the variable 'age'
ccavg=discretize(univ2$ccavg, disc="equalwidth",
nbins=10)
ccavg=as.factor(ccavg$X)

#Discretizing the variable 'age'
mortgage=discretize(univ2$mortgage, disc="equal
nbins=5)
mortgage=as.factor(mortgage$X)

# *** Removing the numerical variables from the or
# *** data and adding the categorical forms of them
head(univ2)
univ2 <- subset(univ2, select= -c(age,inc,ccavg,mo
head(univ2)
univ2 <- cbind(age,inc,ccavg,mortgage,univ2)
head(univ2,20)
dim(univ2)
str(univ2)
summary(univ2)

# Let us divide the data into training, testing
# and evaluation data sets
rows=seq(1,5000,1)
set.seed(123)
trainRows=sample(rows,3000)
set.seed(123)
remainingRows=rows[-(trainRows)]
testRows=sample(remainingRows, 1000)
evalRows=rows[-c(trainRows,testRows)]

train = univ2[trainRows,]
test=univ2[testRows,]
eval=univ2[evalRows,]
dim(train); dim(test); dim(eval)
rm(age,ccavg, mortgage, inc, univ)

#### Building Models

#Decision Trees using C50

names(train)
#install.packages("C50")
```

```
library(C50)
dtC50 <- C5.0(loan ~ ., data = train, rules=TRUE)
summary(dtC50)

predict(dtC50, newdata=train, type="class")
a=table(train$loan, predict(dtC50,
newdata=train, type="class"))
rcTrain=(a[2,2])/(a[2,1]+a[2,2])*100
rcTrain

# Predicting on Testing Data
predict(dtC50, newdata=test, type="class")
a=table(test$loan, predict(dtC50,
newdata=test, type="class"))
rcTest=(a[2,2])/(a[2,1]+a[2,2])*100
rcTest

# Predicting on Evaluation Data
predict(dtC50, newdata=eval, type="class")
a=table(eval$loan, predict(dtC50,
newdata=eval, type="class"))
rcEval=(a[2,2])/(a[2,1]+a[2,2])*100
rcEval

cat("Recall in Training", rcTrain, '\n',
"Recall in Testing", rcTest, '\n',
"Recall in Evaluation", rcEval)

#Test by increasing the number of bins in inc and ccavg to 10
#Test by changing the bin to euqalwidth in inc and ccavg

library(ggplot2)
#using qplot
qplot(edu, inc, data=univ2, color=loan,
size=as.numeric(ccavg))+
theme_bw()+scale_size_area(max_size=9)+
xlab("Educational qualifications") +
ylab("Income") +
theme(axis.text.x=element_text(size=18),
axis.title.x = element_text(size =18,
colour = 'black'))+
theme(axis.text.y=element_text(size=18),
axis.title.y = element_text(size = 18,
colour = 'black',
angle = 90))

#using ggplot
ggplot(data=univ2,
aes(x=edu, y=inc, color=loan,
size=as.numeric(ccavg)))+
geom_point()+
scale_size_area(max_size=9)+
xlab("Educational qualifications") +
```

```
ylab("Income") +
theme_bw()+
theme(axis.text.x=element_text(size=18),
axis.title.x = element_text(size =18,
colour = 'black'))+
theme(axis.text.y=element_text(size=18),
axis.title.y = element_text(size = 18,
colour = 'black',
angle = 90))

rm(a,rcEval,rcTest,rcTrain)

#————————————————————

#Decision Trees using CART

#Load the rpart package
library(rpart)

#Use the rpart function to build a classification tree
dtCart <- rpart(loan ~ ., data=train, method="class"

#Type churn.rp to retrieve the node detail of the
#classification tree
dtCart

#Use the printcp function to examine the complexit
printcp(dtCart)

#use the plotcp function to plot the cost complexity
plotcp(dtCart)

#plot function and the text function to plot the classification tree
plot(dtCart,main="Classification Tree for loan Class",
margin=.1, uniform=TRUE)
text(dtCart, use.n=T)

## steps to validate the prediction performance of a classification tree
———————————————————————————

predict(dtCart, newdata=train, type="class")
a <- table(train$loan, predict(dtCart,
newdata=train, type="class"))
dtrain <- (a[2,2])/(a[2,1]+a[2,2])*100

a <-table(test$loan, predict(dtCart,
newdata=test, type="class"))
dtest <- (a[2,2])/(a[2,1]+a[2,2])*100

a <- table(eval$loan, predict(dtCart,
newdata=eval, type="class"))
deval <- (a[2,2])/(a[2,1]+a[2,2])*100

cat("Recall in Training", dtrain, '\n',
"Recall in Testing", dtest, '\n',
"Recall in Evaluation", deval)
```


Your Start in Machine Learning

```
#### Pruning a tree
———————————

#Finding the minimum cross-validation error of the
#classification tree model
min(dtCart$cptable[,"xerror"])

#Locate the record with the minimum cross-validation errors
which.min(dtCart$cptable[,"xerror"])

#Get the cost complexity parameter of the record with
#the minimum cross-validation errors
dtCart.cp <- dtCart$cptable[5,"CP"]
dtCart.cp

#Prune the tree by setting the cp parameter to the
#of the record with minimum cross-validation errors
prune.tree <- prune(dtCart, cp= dtCart.cp)
prune.tree

#Visualize the classification tree by using the plot a
#text function
plot(prune.tree, margin= 0.01)
text(prune.tree, all=FALSE , use.n=TRUE)

## steps to validate the prediction performance of a
———————————————————————————————

a <- table(train$loan, predict(prune.tree,
newdata=train, type="class"))
dtrain <- (a[2,2])/(a[2,1]+a[2,2])*100

a <-table(test$loan, predict(prune.tree,
newdata=test, type="class"))
dtest <- (a[2,2])/(a[2,1]+a[2,2])*100

a <- table(eval$loan, predict(prune.tree,
newdata=eval, type="class"))
deval <- (a[2,2])/(a[2,1]+a[2,2])*100

cat("Recall in Training", dtrain, '\n',
"Recall in Testing", dtest, '\n',
"Recall in Evaluation", deval)

#————————————————————————

# Decision tree using Conditional Inference

library(party)
ctree.model= ctree(loan ~ ., data = train)
plot(ctree.model)

a=table(train$loan, predict(ctree.model, newdata=train))
djtrain <- (a[2,2])/(a[2,1]+a[2,2])*100

a=table(test$loan, predict(ctree.model, newdata=test))
djtest <- (a[2,2])/(a[2,1]+a[2,2])*100
```

a=table(eval$loan, predict(ctree.model, newdata=eval))
djeval <- (a[2,2])/(a[2,1]+a[2,2])*100

cat("Recall in Training", djtrain, '\n',
"Recall in Testing", djtest, '\n',
"Recall in Evaluation", djeval)

Jason Brownlee September 11, 2017 at 12:09 pm # REPLY

I cannot debug your code.

Rishabh October 10, 2017 at 10:58 pm #

Hey Jason!

"Get your FREE Algorithms Mind Map": Top of the
which does not exist.

Please help.

Jason Brownlee October 11, 2017 at 7:5

The link works for me.

You can sign-up to get the mind map here:
https://machinelearningmastery.leadpages.co/machine-learning-algorithms-mini-course/

Aniket Saxena October 29, 2017 at 2:34 am # REPLY

At this link(www.saedsayad.com/decision_tree.htm), Saed wrote at the bottom of the page some issues about decision trees. So my question is,

1. What if we dealt with missing values in the dataset prior to fit the model to our dataset, how decision trees will work as Saed mention that decision trees only work with missing values?

2. What are continuous attributes as Saed allude that decision trees algorithm works with continuous attributes(binning)?

Please help regarding these questions……..

Jason Brownlee October 29, 2017 at 5:56 am # REPLY

Indeed, missing values can be treated as another value to split on.

Continuous means real-valued.

If you have questions for Saed, perhaps ask him directly?

Karthick January 9, 2018 at 9:50 pm # REPLY

Can CART be used in SuperMarket??

Jason Brownlee January 10, 2018 at 5:25 am # REPLY

What is supermarket?

Dhanunjaya Mitta January 26, 2018 at 10:08

Hello Jason,
I would like to know what actually the greedy appro
them

Jason Brownlee January 27, 2018 at 5:5

I show how to implement the gini scor
https://machinelearningmastery.com/implement-decision-tree-algorithm-scratch-python/

Adiputra Simanjuntak February 12, 2018 at 2:25 pm # REPLY

Good morning Sir, I want to ask you about CART algorithm for predicting continuous number… I have read another reference and it tells that to predict continuous number, we should replace the gini index using standard deviation reduction. is it true??
thanks Sir

Jason Brownlee February 12, 2018 at 2:52 pm # REPLY

The gini index is for CART for classification, you will need to use a different metric for regression. Sorry I do not have an example.

Adiputra Simanjuntak July 26, 2018 at 2:11 pm # REPLY

Thanks for the response Sir. According to that, I want to ask your opinion about standard deviation reduction can be used f explain to me, Sir. Thanks Sir

Jason Brownlee July 26, 2018 at 2:29 pm # REPLY

Sorry, I don't understand your question. Perhaps you can provide more context?

What is "standard deviation reduction" and how does it relate to regression trees?

Adiputra Simanjuntak July 26, 2018 at 3:18 pm #

I have read this article htt|

He used the Standard Deviation R
confuse to build the tree using suc

Thanks before

Jason Brownlee July 27,

Sorry, I don't know about

Hninnyuhlaing June 11, 2018 at 3:21 pm # REPLY

hello!sir
I want to know,CART can use in my system including three class labels .

Jason Brownlee June 12, 2018 at 6:35 am # REPLY

CART can support 2 or more class labels.

Leave a Reply

Name (required)

Email (will not be published) (required)

Website

SUBMIT COMMENT

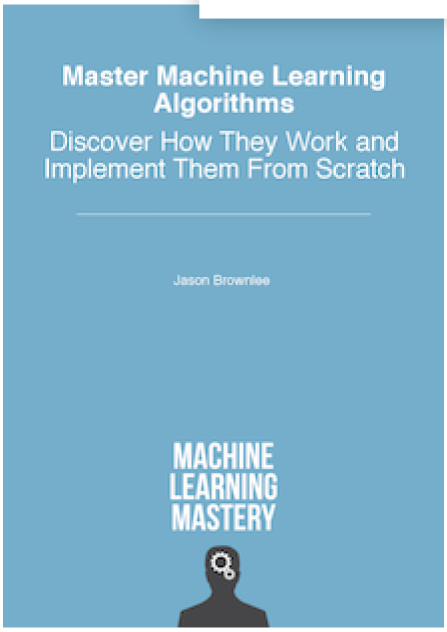
Welcome to Machine Learning Mastery!

Hi, I'm Jason Brownlee, PhD
I write tutorials to help developers (lik

Read More

Understand Ho
No theory, just

Click to Ge



POPULAR

How to Develop a Deep Learning Photo Caption Generator from Scratch
NOVEMBER 27, 2017

How to Develop a Neural Machine Translation System from Scratch
JANUARY 10, 2018


Difference Between Classification and Regression in Machine Learning
DECEMBER 11, 2017


How to Develop a Word-Level Neural Language Model and Use it to Generate Text
NOVEMBER 10, 2017


How to Develop an N-gram Multichannel Convolutional Neural Network for Sentiment Analysis
JANUARY 12, 2018


So, You are Working on a Machine Learning Problem…
APRIL 4, 2018


Encoder-Decoder Models for Text Summarizati
DECEMBER 8, 2017

You might also like…

- How to Install Python for Machine Learning
- Your First Machine Learning Project in Python
- Your First Neural Network in Python
- Your First Classifier in Weka
- Your First Time Series Forecasting Project