title: “Week 2 Basic Classification Models” output: html_document —

R Packages

The packages you will need to install for the week are class, gmodels, rpart, rpart.plot, caret, party, and partykit.

library(class) library(gmodels) library(rpart) library(rpart.plot) library(caret) library(party) library(partykit)

Classification Models

Last week we used linear regression models to make numerical predictions (wages and insurance charges). Our task for this week is to create models that predict categorical class labels. In particular, we will look at two classifical models: **k-nearest neighbor (knn)** and **decision trees**.

And in case you are wondering, we are still working with supervised learning algorithms.

Background on the Drug Overdose Epidemic

In 2015, Angus Deacon and Anne Case, economists (and husband and wife!) from Princeton University, published a startling study. Deacon and Case found that mortality rate for middle aged (45 to 54 years old) non-Hispanic whites with a high school education or lower increased between 1999 and 2014, even though the mortality rates for all other age and racial groups were declining. This trend was happening even as the mortality rates of middle aged whites in other developed countries were declining. Deacon and Case found that the causes of death among less educated middle aged white Americans include suicide, alcohol, and drug overdose.

Since the publication of the Deacon & Case study, public interest in the drug overdose epidemic has increased. Gina Kolata and Sarah Cohen (2016) of the *New York Times* analyzed 60 million death certificates between 1999 and 2014 and found that the mortality rates among American non-Hispanic whites across all age groups under 65 years old were either rising or flattening. Kolata and Cohen reported:

In 2014, the overdose death rate for whites ages 25 to 34 was five times its level in 1999, and the rate for 35- to 44-year-old whites tripled during that period. The numbers cover both illegal and prescription drugs. . . .Rising rates of overdose deaths and suicide appear to have erased the benefits from advances in medical treatment for most age groups of whites Kolata and Cohen 2016.

The Dataset

We will be working with a dataset posted on Kaggle by Alan Pryor Jr. The dataset includes non-opioid prescription records and demographic information of 25,000 licensed medical professionals. The prescriptions were written for individuals covered under Class D Medicare. The source of the data is from the [Center for Medicare and Medicaid Services] (https://www.cms.gov/).

The dataset contains the following information:

*Gender of licensed medical professional

*Number of prescriptions written for each of 239 common non-opiate drugs in 2014 by the licensed medical professional

*A series of dummy variables for the state in which the medical professional practiced

*A series of dummy variables for the medical professional's specialty

*A factor variable named "Opioid.Prescriber" indicating whether the medical professional wrote at least 10 prescriptions for opioid drugs in 2014

Prediction Goal

Can we build a model to predict whether a medical professional is likely to be an opioid prescriber? Additionally, can we identify predictors that tell us if a medical professional is more likely to prescribe opioids?

Exploratory Data Analysis

```
setwd("C:/Users/corylowe/OneDrive/Code/R Practice Code/Applied Data Mining_Portfolio/Week 2")
prescribers<-read.csv("prescribers.csv")


#View(prescribers)

prescribers<-prescribers[,c(241,1:240,242:331)] #Rearranging the columns so that our target variable is

dim(prescribers)
```

```
## [1] 25000   331
```

```
#names(prescribers)

table(prescribers$Opioid.Prescriber)
```

```
##
##    no   yes
## 10312 14688
```

Classifical Models: A Two Step Process

Step #1: Model Construction

- What is my class label? (yes/no)
- Build a training set (portion of the data used to build a prediction model)
- Choose a classification algorithm to fit the training set

Step #2: Model Usage

- Run the classification algorithm on the test set
- Examine performance of the "prediction exercise" (i.e. Confusion Matrix)

k-Nearest Neighbor (kNN): A Lazy Classification Model

kNN is called a "lazy learner" because it does not perform abstraction. Lantz (2013) noted:

In a single sentence, nearest neighbor classifiers are defined by their characteristics of classifying unlabeled examples by assigning them the class of the most similar labeled examples (Lantz 2013, p. 66).

When comparing among neighbors, we need to use a distance function. The most common way to measure distance is **Euclidean distance**, or the shortest direct route.

How many neighbors (k)?

When choosing the number of k, we need to consider the **bias-variance tradeoff**. A large k reduces the variance caused by noisy data but can cause bias in that we risk ignoring small (and important) patterns (Lantz 2013, p. 71).

kNN requires data transformation into a standard range

Min-Max Normalization

Also see page 73 of Lantz text

In our prescribers dataset, we have two factors: Gender and Opioid.Prescriber. We will leave the Opioid.Prescriber alone since this is our target variable. We need to change Gender into a dummy variable so that it will be on the same 0,1 scale as our other variables (once we perform min-max normalization).

```
prescribers$Male <-ifelse(prescribers$Gender=="M",1,0) #if Male = 1; if Female=0.
prescribers<-prescribers[,-2] #We do not need the Gender variable anymore.

names(prescribers)
```

```
##   [1] "Opioid.Prescriber"
##   [2] "ABILIFY"
##   [3] "ACYCLOVIR"
##   [4] "ADVAIR.DISKUS"
##   [5] "AGGRENOX"
##   [6] "ALENDRONATE.SODIUM"
##   [7] "ALLOPURINOL"
##   [8] "ALPRAZOLAM"
##   [9] "AMIODARONE.HCL"
##  [10] "AMITRIPTYLINE.HCL"
##  [11] "AMLODIPINE.BESYLATE"
##  [12] "AMLODIPINE.BESYLATE.BENAZEPRIL"
##  [13] "AMOXICILLIN"
##  [14] "AMOX.TR.POTASSIUM.CLAVULANATE"
##  [15] "AMPHETAMINE.SALT.COMBO"
##  [16] "ATENOLOL"
##  [17] "ATORVASTATIN.CALCIUM"
##  [18] "AVODART"
##  [19] "AZITHROMYCIN"
##  [20] "BACLOFEN"
```

```
##  [21] "BD.ULTRA.FINE.PEN.NEEDLE"
##  [22] "BENAZEPRIL.HCL"
##  [23] "BENICAR"
##  [24] "BENICAR.HCT"
##  [25] "BENZTROPINE.MESYLATE"
##  [26] "BISOPROLOL.HYDROCHLOROTHIAZIDE"
##  [27] "BRIMONIDINE.TARTRATE"
##  [28] "BUMETANIDE"
##  [29] "BUPROPION.HCL.SR"
##  [30] "BUPROPION.XL"
##  [31] "BUSPIRONE.HCL"
##  [32] "BYSTOLIC"
##  [33] "CARBAMAZEPINE"
##  [34] "CARBIDOPA.LEVODOPA"
##  [35] "CARISOPRODOL"
##  [36] "CARTIA.XT"
##  [37] "CARVEDILOL"
##  [38] "CEFUROXIME"
##  [39] "CELEBREX"
##  [40] "CEPHALEXIN"
##  [41] "CHLORHEXIDINE.GLUCONATE"
##  [42] "CHLORTHALIDONE"
##  [43] "CILOSTAZOL"
##  [44] "CIPROFLOXACIN.HCL"
##  [45] "CITALOPRAM.HBR"
##  [46] "CLINDAMYCIN.HCL"
##  [47] "CLOBETASOL.PROPIONATE"
##  [48] "CLONAZEPAM"
##  [49] "CLONIDINE.HCL"
##  [50] "CLOPIDOGREL"
##  [51] "CLOTRIMAZOLE.BETAMETHASONE"
##  [52] "COLCRYS"
##  [53] "COMBIVENT.RESPIMAT"
##  [54] "CRESTOR"
##  [55] "CYCLOBENZAPRINE.HCL"
##  [56] "DEXILANT"
##  [57] "DIAZEPAM"
##  [58] "DICLOFENAC.SODIUM"
##  [59] "DICYCLOMINE.HCL"
##  [60] "DIGOX"
##  [61] "DIGOXIN"
##  [62] "DILTIAZEM.24HR.CD"
##  [63] "DILTIAZEM.24HR.ER"
##  [64] "DILTIAZEM.ER"
##  [65] "DILTIAZEM.HCL"
##  [66] "DIOVAN"
##  [67] "DIPHENOXYLATE.ATROPINE"
##  [68] "DIVALPROEX.SODIUM"
##  [69] "DIVALPROEX.SODIUM.ER"
##  [70] "DONEPEZIL.HCL"
##  [71] "DORZOLAMIDE.TIMOLOL"
##  [72] "DOXAZOSIN.MESYLATE"
##  [73] "DOXEPIN.HCL"
##  [74] "DOXYCYCLINE.HYCLATE"
```

```
##  [75] "DULOXETINE.HCL"
##  [76] "ENALAPRIL.MALEATE"
##  [77] "ESCITALOPRAM.OXALATE"
##  [78] "ESTRADIOL"
##  [79] "EXELON"
##  [80] "FAMOTIDINE"
##  [81] "FELODIPINE.ER"
##  [82] "FENOFIBRATE"
##  [83] "FINASTERIDE"
##  [84] "FLOVENT.HFA"
##  [85] "FLUCONAZOLE"
##  [86] "FLUOXETINE.HCL"
##  [87] "FLUTICASONE.PROPIONATE"
##  [88] "FUROSEMIDE"
##  [89] "GABAPENTIN"
##  [90] "GEMFIBROZIL"
##  [91] "GLIMEPIRIDE"
##  [92] "GLIPIZIDE"
##  [93] "GLIPIZIDE.ER"
##  [94] "GLIPIZIDE.XL"
##  [95] "GLYBURIDE"
##  [96] "HALOPERIDOL"
##  [97] "HUMALOG"
##  [98] "HYDRALAZINE.HCL"
##  [99] "HYDROCHLOROTHIAZIDE"
## [100] "HYDROCORTISONE"
## [101] "HYDROXYZINE.HCL"
## [102] "IBANDRONATE.SODIUM"
## [103] "IBUPROFEN"
## [104] "INSULIN.SYRINGE"
## [105] "IPRATROPIUM.BROMIDE"
## [106] "IRBESARTAN"
## [107] "ISOSORBIDE.MONONITRATE.ER"
## [108] "JANTOVEN"
## [109] "JANUMET"
## [110] "JANUVIA"
## [111] "KETOCONAZOLE"
## [112] "KLOR.CON.10"
## [113] "KLOR.CON.M10"
## [114] "KLOR.CON.M20"
## [115] "LABETALOL.HCL"
## [116] "LACTULOSE"
## [117] "LAMOTRIGINE"
## [118] "LANSOPRAZOLE"
## [119] "LANTUS"
## [120] "LANTUS.SOLOSTAR"
## [121] "LATANOPROST"
## [122] "LEVEMIR"
## [123] "LEVEMIR.FLEXPEN"
## [124] "LEVETIRACETAM"
## [125] "LEVOFLOXACIN"
## [126] "LEVOTHYROXINE.SODIUM"
## [127] "LIDOCAINE"
## [128] "LISINOPRIL"
```

```
## [129] "LISINOPRIL.HYDROCHLOROTHIAZIDE"
## [130] "LITHIUM.CARBONATE"
## [131] "LORAZEPAM"
## [132] "LOSARTAN.HYDROCHLOROTHIAZIDE"
## [133] "LOSARTAN.POTASSIUM"
## [134] "LOVASTATIN"
## [135] "LOVAZA"
## [136] "LUMIGAN"
## [137] "LYRICA"
## [138] "MECLIZINE.HCL"
## [139] "MELOXICAM"
## [140] "METFORMIN.HCL"
## [141] "METFORMIN.HCL.ER"
## [142] "METHOCARBAMOL"
## [143] "METHOTREXATE"
## [144] "METHYLPREDNISOLONE"
## [145] "METOCLOPRAMIDE.HCL"
## [146] "METOLAZONE"
## [147] "METOPROLOL.SUCCINATE"
## [148] "METOPROLOL.TARTRATE"
## [149] "METRONIDAZOLE"
## [150] "MIRTAZAPINE"
## [151] "MONTELUKAST.SODIUM"
## [152] "MUPIROCIN"
## [153] "NABUMETONE"
## [154] "NAMENDA"
## [155] "NAMENDA.XR"
## [156] "NAPROXEN"
## [157] "NASONEX"
## [158] "NEXIUM"
## [159] "NIACIN.ER"
## [160] "NIFEDICAL.XL"
## [161] "NIFEDIPINE.ER"
## [162] "NITROFURANTOIN.MONO.MACRO"
## [163] "NITROSTAT"
## [164] "NORTRIPTYLINE.HCL"
## [165] "NOVOLOG"
## [166] "NOVOLOG.FLEXPEN"
## [167] "NYSTATIN"
## [168] "OLANZAPINE"
## [169] "OMEPRAZOLE"
## [170] "ONDANSETRON.HCL"
## [171] "ONDANSETRON.ODT"
## [172] "ONGLYZA"
## [173] "OXCARBAZEPINE"
## [174] "OXYBUTYNIN.CHLORIDE"
## [175] "OXYBUTYNIN.CHLORIDE.ER"
## [176] "PANTOPRAZOLE.SODIUM"
## [177] "PAROXETINE.HCL"
## [178] "PHENOBARBITAL"
## [179] "PHENYTOIN.SODIUM.EXTENDED"
## [180] "PIOGLITAZONE.HCL"
## [181] "POLYETHYLENE.GLYCOL.3350"
## [182] "POTASSIUM.CHLORIDE"
```

```
## [183] "PRADAXA"
## [184] "PRAMIPEXOLE.DIHYDROCHLORIDE"
## [185] "PRAVASTATIN.SODIUM"
## [186] "PREDNISONE"
## [187] "PREMARIN"
## [188] "PRIMIDONE"
## [189] "PROAIR.HFA"
## [190] "PROMETHAZINE.HCL"
## [191] "PROPRANOLOL.HCL"
## [192] "PROPRANOLOL.HCL.ER"
## [193] "QUETIAPINE.FUMARATE"
## [194] "QUINAPRIL.HCL"
## [195] "RALOXIFENE.HCL"
## [196] "RAMIPRIL"
## [197] "RANEXA"
## [198] "RANITIDINE.HCL"
## [199] "RESTASIS"
## [200] "RISPERIDONE"
## [201] "ROPINIROLE.HCL"
## [202] "SEROQUEL.XR"
## [203] "SERTRALINE.HCL"
## [204] "SIMVASTATIN"
## [205] "SOTALOL"
## [206] "SPIRIVA"
## [207] "SPIRONOLACTONE"
## [208] "SUCRALFATE"
## [209] "SULFAMETHOXAZOLE.TRIMETHOPRIM"
## [210] "SUMATRIPTAN.SUCCINATE"
## [211] "SYMBICORT"
## [212] "SYNTHROID"
## [213] "TAMSULOSIN.HCL"
## [214] "TEMAZEPAM"
## [215] "TERAZOSIN.HCL"
## [216] "TIMOLOL.MALEATE"
## [217] "TIZANIDINE.HCL"
## [218] "TOLTERODINE.TARTRATE.ER"
## [219] "TOPIRAMATE"
## [220] "TOPROL.XL"
## [221] "TORSEMIDE"
## [222] "TRAVATAN.Z"
## [223] "TRAZODONE.HCL"
## [224] "TRIAMCINOLONE.ACETONIDE"
## [225] "TRIAMTERENE.HYDROCHLOROTHIAZID"
## [226] "VALACYCLOVIR"
## [227] "VALSARTAN"
## [228] "VALSARTAN.HYDROCHLOROTHIAZIDE"
## [229] "VENLAFAXINE.HCL"
## [230] "VENLAFAXINE.HCL.ER"
## [231] "VENTOLIN.HFA"
## [232] "VERAPAMIL.ER"
## [233] "VESICARE"
## [234] "VOLTAREN"
## [235] "VYTORIN"
## [236] "WARFARIN.SODIUM"
```

```
## [237] "XARELTO"
## [238] "ZETIA"
## [239] "ZIPRASIDONE.HCL"
## [240] "ZOLPIDEM.TARTRATE"
## [241] "AL"
## [242] "AR"
## [243] "AZ"
## [244] "CA"
## [245] "CO"
## [246] "CT"
## [247] "DC"
## [248] "DE"
## [249] "FL"
## [250] "GA"
## [251] "HI"
## [252] "IA"
## [253] "ID"
## [254] "IL"
## [255] "IN"
## [256] "KS"
## [257] "KY"
## [258] "LA"
## [259] "MA"
## [260] "MD"
## [261] "ME"
## [262] "MI"
## [263] "MN"
## [264] "MO"
## [265] "MS"
## [266] "MT"
## [267] "NC"
## [268] "ND"
## [269] "NE"
## [270] "NH"
## [271] "NJ"
## [272] "NM"
## [273] "NV"
## [274] "NY"
## [275] "OH"
## [276] "OK"
## [277] "OR"
## [278] "PA"
## [279] "PR"
## [280] "RI"
## [281] "SC"
## [282] "SD"
## [283] "TN"
## [284] "TX"
## [285] "UT"
## [286] "VA"
## [287] "VT"
## [288] "WA"
## [289] "WI"
## [290] "WV"
```

```
## [291] "WY"
## [292] "other"
## [293] "Anesthesiology"
## [294] "Cardiology"
## [295] "Certified.Clinical.Nurse.Specialist"
## [296] "Dentist"
## [297] "Dermatology"
## [298] "Emergency.Medicine"
## [299] "Endocrinology"
## [300] "Family.Practice"
## [301] "Gastroenterology"
## [302] "General.Practice"
## [303] "Geriatric.Medicine"
## [304] "Hematology.Oncology"
## [305] "Infectious.Disease"
## [306] "Internal.Medicine"
## [307] "Medical.Oncology"
## [308] "Nephrology"
## [309] "Neurology"
## [310] "Neuropsychiatry"
## [311] "Nurse.Practitioner"
## [312] "Obstetrics.Gynecology"
## [313] "Ophthalmology"
## [314] "Optometry"
## [315] "Otolaryngology"
## [316] "Pediatric.Medicine"
## [317] "Physical.Medicine.and.Rehabilitation"
## [318] "Physician.Assistant"
## [319] "Podiatry"
## [320] "Psychiatry"
## [321] "Psychiatry...Neurology"
## [322] "Pulmonary.Disease"
## [323] "Radiation.Oncology"
## [324] "Rheumatology"
## [325] "Specialist"
## [326] "Student.in.an.Organized.Health.Care.Education.Training.Program"
## [327] "Urology"
## [328] "Surgeon"
## [329] "other.1"
## [330] "Pain.Management"
## [331] "Male"
```

Here is the breakdown of the 331 variables:

Column 1: target variable

Columns 2-240: number of prescriptions written for each non-opioid drug

Columns 241-291: state dummy variables

Columns 292-330: medical speciality dummy variables

Column 331: dummy variable for male (i.e. gender)

We need to do min-max normalization for columns 2-240 and then add that with our other colums (already on the 0-1 scale).

```
drugs<-prescribers[,2:240]
normalize<- function(x){return((x-min(x))/(max(x)-min(x)))}
drugs_n<-as.data.frame(lapply(drugs, normalize))
```

lapply wil apply as a list

Let's check our work to see if we did it correctly!

```
summary(drugs$ABILIFY) #Range was between 0 and 770
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   0.000   3.157   0.000 770.000
```

```
summary(drugs_n$ABILIFY) #Notice the range is now between 0 and 1
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.0041  0.0000  1.0000
```

Now we are going to combine the normalized variables with our dummy variables and the target variable.

```
prescribers_n<-cbind(prescribers[,c(1,241:331)], drugs_n[,])

prescribers_n<-prescribers_n[complete.cases(prescribers_n),]
```

Train and Test Sets

We divide our dataset into two subsets: training set and a test set. We use the training set to "train" our kNN model. We then use that model to predict the observations in our test set. This is how we gauge the performance of our prediction model. We will split our dataset into 80-20 (80% training and 20% test sets).

```
prescribers_n_train <- prescribers_n[1:20000,2:331]
prescribers_n_test <- prescribers_n[20001:25000,2:331]

prescribers_n_train_labels<-prescribers_n[1:20000,1]
prescribers_n_test_labels<-prescribers_n[20001:25000,1]
```

We will use the **class** package to perform kNN.

```
library(class)
prescribers_pred<-knn(train=prescribers_n_train, test=prescribers_n_test, cl=prescribers_n_train_labels
```

Evaluating Model Performance

You have to decide what is a "positive" versus a "negative" case in your dataset! We will label a positive case as someone who did prescribe opioids more than 10 times in 2014. A negative case is someone who did not.

True Positives = 2308

True Negatives = 1550

False Positives = 500

False Negatives = 642

```
library(gmodels)

CrossTable(x=prescribers_n_test_labels, y=prescribers_pred, prob.chisq=FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  5000
##
##
##                            | prescribers_pred
## prescribers_n_test_labels |        no |       yes | Row Total |
## --------------------------|-----------|-----------|-----------|
##                        no |      1555 |       496 |      2051 |
##                           |   431.331 |   353.478 |           |
##                           |     0.758 |     0.242 |     0.410 |
##                           |     0.690 |     0.180 |           |
##                           |     0.311 |     0.099 |           |
## --------------------------|-----------|-----------|-----------|
##                       yes |       697 |      2252 |      2949 |
##                           |   299.986 |   245.840 |           |
##                           |     0.236 |     0.764 |     0.590 |
##                           |     0.310 |     0.820 |           |
##                           |     0.139 |     0.450 |           |
## --------------------------|-----------|-----------|-----------|
##              Column Total |      2252 |      2748 |      5000 |
##                           |     0.450 |     0.550 |           |
## --------------------------|-----------|-----------|-----------|
##
##
```

```
Sensitivity = (2308/(2308+500))*100
Specificity = (1550/(1500+642))*100
Precision = (2308/(2308+500))*100
Accuracy = ((1550+2308)/5000)*100

print(Accuracy)
```

```
## [1] 77.16
```

Our lazy learner correctly classified **77.16%** of all the medical professionals as individuals who did or did not prescribe opioids more than 10 times in 2014. Not bad for a lazy learner!

Improving on the kNN Model

You should play around with the k value to see if you can improve the model performance. Try it!

Z score standardization

```
prescribers_z <- as.data.frame(scale(prescribers[-1]))

summary(prescribers$ABILIFY)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   0.000   3.157   0.000 770.000
```

```
summary(prescribers_z$ABILIFY) #notice that the max value is not compressed towards 1.
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.1521 -0.1521 -0.1521  0.0000 -0.1521 36.9459
```

```
prescribers_z_train<-prescribers_z[1:20000, ]
prescribers_z_test<-prescribers_z[20001:25000, ]

prescribers_z_train_labels<-prescribers[1:20000,1]
prescribers_z_test_labels<-prescribers[20001:25000,1]

prescribers_z_pred <- knn(train=prescribers_z_train, test=prescribers_z_test, cl=prescribers_z_train_lab

CrossTable(x=prescribers_z_test_labels, y=prescribers_z_pred, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  5000
##
##
##                            | prescribers_z_pred
## prescribers_z_test_labels |        no |       yes | Row Total |
## --------------------------|-----------|-----------|-----------|
```

```
##                          no |      1463 |       588 |      2051 |
##                             |     0.713 |     0.287 |     0.410 |
##                             |     0.658 |     0.212 |           |
##                             |     0.293 |     0.118 |           |
## ----------------------------|-----------|-----------|-----------|
##                         yes |       760 |      2189 |      2949 |
##                             |     0.258 |     0.742 |     0.590 |
##                             |     0.342 |     0.788 |           |
##                             |     0.152 |     0.438 |           |
## ----------------------------|-----------|-----------|-----------|
##               Column Total |      2223 |      2777 |      5000 |
##                             |     0.445 |     0.555 |           |
## ----------------------------|-----------|-----------|-----------|
##
##
```

```
Accuracy = ((1466+2191)/5000)*100

print(Accuracy)
```

```
## [1] 73.14
```

Z-transformation actually reduced the accuracy rate: **77.14%**.

Decision Trees: A More Sophisticated Classification Model

Decision trees follow recursive partitioning (top down greedy divide and conquer approach)

1. Choose the attribute that is most predictive of the target variable
2. Observations in the training data set are divided into groups of distinct values. This form the first set of branches.
3. Continue to divide and conquer the nodes, choosing the feature with the most prediction power each time until one of three conditions occur:

- all observations for a given node belong to the same class
- no more remaining attributes for further partitioning
- no observations are left

Splitting Criterion

Creating a Training and Test Set by Randomizing Observations

```
set.seed(12345) #set a seed to do draws from a random uniform distribution.
prescribers_rand <- prescribers[order(runif(25000)), ]
prescribers_train <- prescribers_rand[1:20000, ] #Training data set; 3000 observations
prescribers_test  <-prescribers_rand[20001:25000, ]
```

Using rpart to Build a Decision Tree

```
library(rpart)

prescribers_rpart <- rpart(prescribers_train$Opioid.Prescriber~., method="class", parms = list(split="g

#More on the method options:
# a. method="class" --> categorical (yes/no)
# b. method="anova" --> continuous
# c. method="poisson" --> count
# d. method="exp" --> survival analysis (in poverty/out of poverty)

#More on the parms option:
#a. The default splitting criterion is the Gini Index.
```

Root, Nodes, and Leaves

```
summary(prescribers_rpart)
```

```
## Call:
## rpart(formula = prescribers_train$Opioid.Prescriber ~ ., data = prescribers_train,
##     method = "class", parms = list(split = "gini"))
##   n= 20000
##
##           CP nsplit rel error    xerror        xstd
## 1 0.09415190      0 1.0000000 1.0000000 0.008445696
## 2 0.09245329      2 0.8116962 0.8882553 0.008265709
## 3 0.05738898      3 0.7192429 0.7148750 0.007821976
## 4 0.05035186      4 0.6618539 0.6701043 0.007671481
## 5 0.02365931      5 0.6115021 0.6258190 0.007506531
## 6 0.01000000      6 0.5878428 0.6071342 0.007431881
##
## Variable importance
## LEVOTHYROXINE.SODIUM        METFORMIN.HCL          SIMVASTATIN
##                   16                   12                   12
##           LISINOPRIL  HYDROCHLOROTHIAZIDE ATORVASTATIN.CALCIUM
##                   11                   11                   11
##              Surgeon   Emergency.Medicine            MELOXICAM
##                    7                    6                    4
##    CIPROFLOXACIN.HCL               LYRICA       TIZANIDINE.HCL
##                    3                    2                    1
##  CYCLOBENZAPRINE.HCL       TAMSULOSIN.HCL             CELEBREX
##                    1                    1                    1
##          FINASTERIDE              Urology
##                    1                    1
##
## Node number 1: 20000 observations,    complexity param=0.0941519
##   predicted class=yes  expected loss=0.4121  P(node) =1
```

```
##     class counts:  8242 11758
##    probabilities: 0.412 0.588
##   left son=2 (15071 obs) right son=3 (4929 obs)
##   Primary splits:
##       LEVOTHYROXINE.SODIUM < 21.5  to the left,  improve=1270.802, (0 missing)
##       GABAPENTIN           < 12.5  to the left,  improve=1251.078, (0 missing)
##       METFORMIN.HCL        < 18.5  to the left,  improve=1209.829, (0 missing)
##       MELOXICAM            < 5.5   to the left,  improve=1150.233, (0 missing)
##       OMEPRAZOLE           < 20.5  to the left,  improve=1147.436, (0 missing)
##   Surrogate splits:
##       METFORMIN.HCL        < 15.5  to the left,  agree=0.938, adj=0.747, (0 split)
##       SIMVASTATIN          < 21.5  to the left,  agree=0.928, adj=0.706, (0 split)
##       LISINOPRIL           < 33.5  to the left,  agree=0.914, adj=0.650, (0 split)
##       HYDROCHLOROTHIAZIDE  < 16.5  to the left,  agree=0.913, adj=0.648, (0 split)
##       ATORVASTATIN.CALCIUM < 21.5  to the left,  agree=0.913, adj=0.647, (0 split)
##
## Node number 2: 15071 observations,    complexity param=0.0941519
##   predicted class=no   expected loss=0.4859664  P(node) =0.75355
##     class counts:  7747  7324
##    probabilities: 0.514 0.486
##   left son=4 (13730 obs) right son=5 (1341 obs)
##   Primary splits:
##       Surgeon            < 0.5   to the left,  improve=557.0383, (0 missing)
##       Emergency.Medicine < 0.5   to the left,  improve=393.9895, (0 missing)
##       MELOXICAM          < 5.5   to the left,  improve=320.6390, (0 missing)
##       CEPHALEXIN         < 5.5   to the left,  improve=284.4566, (0 missing)
##       PREDNISONE         < 5.5   to the left,  improve=274.9430, (0 missing)
##
## Node number 3: 4929 observations
##   predicted class=yes  expected loss=0.100426  P(node) =0.24645
##     class counts:   495  4434
##    probabilities: 0.100 0.900
##
## Node number 4: 13730 observations,    complexity param=0.09245329
##   predicted class=no   expected loss=0.4434814  P(node) =0.6865
##     class counts:  7641  6089
##    probabilities: 0.557 0.443
##   left son=8 (12902 obs) right son=9 (828 obs)
##   Primary splits:
##       Emergency.Medicine  < 0.5   to the left,  improve=470.4239, (0 missing)
##       PREDNISONE          < 5.5   to the left,  improve=375.7514, (0 missing)
##       CIPROFLOXACIN.HCL   < 5.5   to the left,  improve=332.8736, (0 missing)
##       CEPHALEXIN          < 5.5   to the left,  improve=283.7882, (0 missing)
##       CYCLOBENZAPRINE.HCL < 5.5   to the left,  improve=263.6881, (0 missing)
##   Surrogate splits:
##       ONDANSETRON.ODT < 11.5  to the left,  agree=0.943, adj=0.054, (0 split)
##
## Node number 5: 1341 observations
##   predicted class=yes  expected loss=0.07904549  P(node) =0.06705
##     class counts:   106  1235
##    probabilities: 0.079 0.921
##
## Node number 8: 12902 observations,    complexity param=0.05738898
##   predicted class=no   expected loss=0.410324  P(node) =0.6451
```

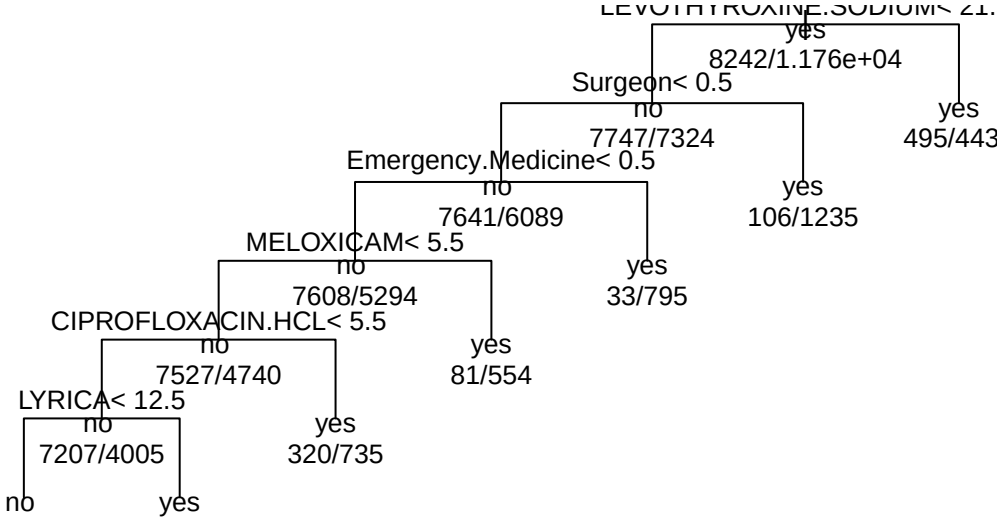
```
##     class counts:  7608  5294
##    probabilities: 0.590 0.410
##   left son=16 (12267 obs) right son=17 (635 obs)
##   Primary splits:
##       MELOXICAM           < 5.5   to the left,  improve=285.2504, (0 missing)
##       LYRICA              < 12.5  to the left,  improve=276.4450, (0 missing)
##       CYCLOBENZAPRINE.HCL < 5.5   to the left,  improve=273.5267, (0 missing)
##       GABAPENTIN          < 5.5   to the left,  improve=255.3779, (0 missing)
##       PREDNISONE          < 5.5   to the left,  improve=255.0635, (0 missing)
##   Surrogate splits:
##       CELEBREX            < 14.5  to the left,  agree=0.958, adj=0.148, (0 split)
##       CYCLOBENZAPRINE.HCL < 33.5  to the left,  agree=0.957, adj=0.123, (0 split)
##       TIZANIDINE.HCL      < 33.5  to the left,  agree=0.956, adj=0.107, (0 split)
##       VOLTAREN            < 17.5  to the left,  agree=0.956, adj=0.096, (0 split)
##       LIDOCAINE           < 20.5  to the left,  agree=0.955, adj=0.082, (0 split)
##
## Node number 9: 828 observations
##   predicted class=yes  expected loss=0.03985507  P(node) =0.0414
##     class counts:    33   795
##    probabilities: 0.040 0.960
##
## Node number 16: 12267 observations,    complexity param=0.05035186
##   predicted class=no   expected loss=0.3864025  P(node) =0.61335
##     class counts:  7527  4740
##    probabilities: 0.614 0.386
##   left son=32 (11212 obs) right son=33 (1055 obs)
##   Primary splits:
##       CIPROFLOXACIN.HCL             < 5.5   to the left,  improve=222.2517, (0 missing)
##       PREDNISONE                    < 5.5   to the left,  improve=214.1934, (0 missing)
##       SULFAMETHOXAZOLE.TRIMETHOPRIM < 5.5   to the left,  improve=199.1432, (0 missing)
##       CEPHALEXIN                    < 5.5   to the left,  improve=173.8988, (0 missing)
##       LATANOPROST                   < 5.5   to the right, improve=168.6610, (0 missing)
##   Surrogate splits:
##       TAMSULOSIN.HCL            < 59    to the left,  agree=0.931, adj=0.192, (0 split)
##       FINASTERIDE               < 16.5  to the left,  agree=0.930, adj=0.181, (0 split)
##       Urology                   < 0.5   to the left,  agree=0.929, adj=0.177, (0 split)
##       VESICARE                  < 17.5  to the left,  agree=0.927, adj=0.154, (0 split)
##       NITROFURANTOIN.MONO.MACRO < 5.5   to the left,  agree=0.927, adj=0.151, (0 split)
##
## Node number 17: 635 observations
##   predicted class=yes  expected loss=0.1275591  P(node) =0.03175
##     class counts:    81   554
##    probabilities: 0.128 0.872
##
## Node number 32: 11212 observations,    complexity param=0.02365931
##   predicted class=no   expected loss=0.3572066  P(node) =0.5606
##     class counts:  7207  4005
##    probabilities: 0.643 0.357
##   left son=64 (10927 obs) right son=65 (285 obs)
##   Primary splits:
##       LYRICA              < 12.5  to the left,  improve=137.5178, (0 missing)
##       LATANOPROST         < 5.5   to the right, improve=133.7560, (0 missing)
##       PREDNISONE          < 5.5   to the left,  improve=128.4142, (0 missing)
##       GABAPENTIN          < 5.5   to the left,  improve=122.1721, (0 missing)
```

```
##       CYCLOBENZAPRINE.HCL < 5.5   to the left,  improve=120.6095, (0 missing)
##   Surrogate splits:
##       GABAPENTIN          < 121.5 to the left,  agree=0.980, adj=0.200, (0 split)
##       TIZANIDINE.HCL      < 20.5  to the left,  agree=0.980, adj=0.196, (0 split)
##       BACLOFEN            < 47    to the left,  agree=0.978, adj=0.130, (0 split)
##       LEVETIRACETAM       < 93.5  to the left,  agree=0.977, adj=0.109, (0 split)
##       CYCLOBENZAPRINE.HCL < 18.5  to the left,  agree=0.977, adj=0.105, (0 split)
##
## Node number 33: 1055 observations
##   predicted class=yes  expected loss=0.3033175  P(node) =0.05275
##     class counts:   320   735
##    probabilities: 0.303 0.697
##
## Node number 64: 10927 observations
##   predicted class=no   expected loss=0.3445593  P(node) =0.54635
##     class counts:  7162  3765
##    probabilities: 0.655 0.345
##
## Node number 65: 285 observations
##   predicted class=yes  expected loss=0.1578947  P(node) =0.01425
##     class counts:    45   240
##    probabilities: 0.158 0.842
```

Visualizing the Decision Tree

```
plot(prescribers_rpart, uniform=TRUE, main="Classification Tree for Opioid Prescribers")
text(prescribers_rpart, use.n=TRUE, all=TRUE, cex=0.8)
```

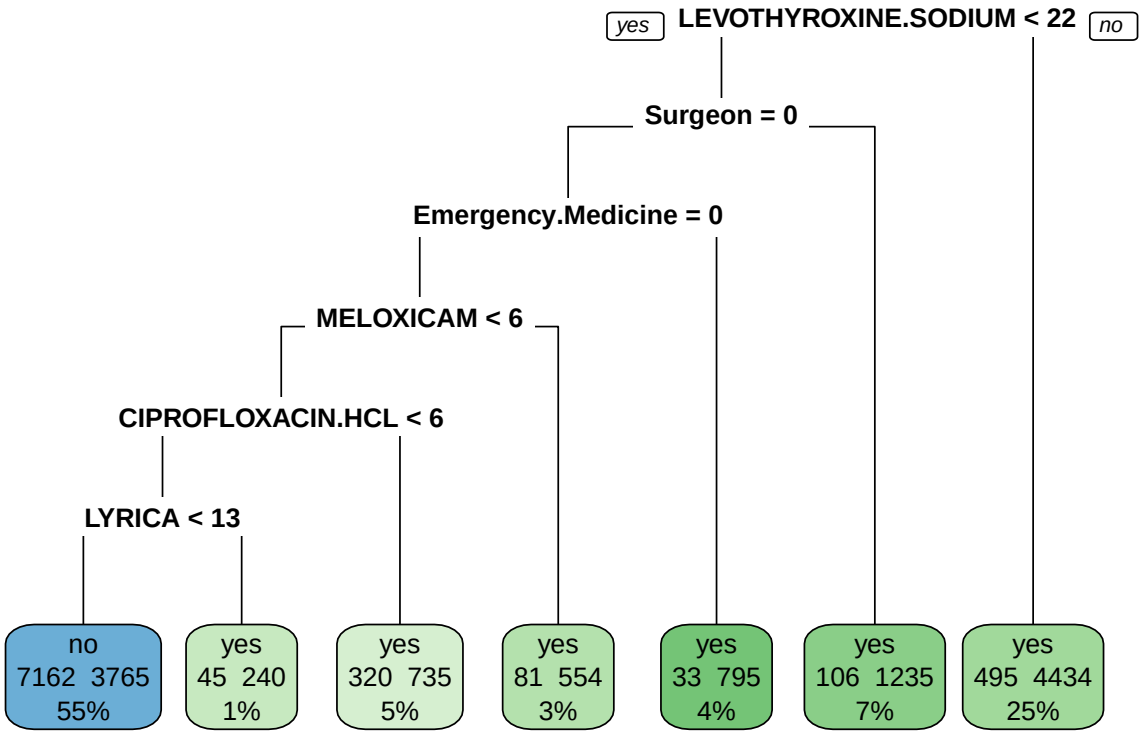
Classification Tree for Opioid Prescribers




```
# Something a bit fancier
library(rpart.plot)
rpart.plot(prescribers_rpart, type=0, extra=101)
#rpart.plot(prescribers_rpart, type=1, extra=101)



# Even fancier?
library(party)
```

```
## Loading required package: grid

## Loading required package: mvtnorm

## Loading required package: modeltools

## Loading required package: stats4

## Loading required package: strucchange

## Loading required package: zoo

##
## Attaching package: 'zoo'
```

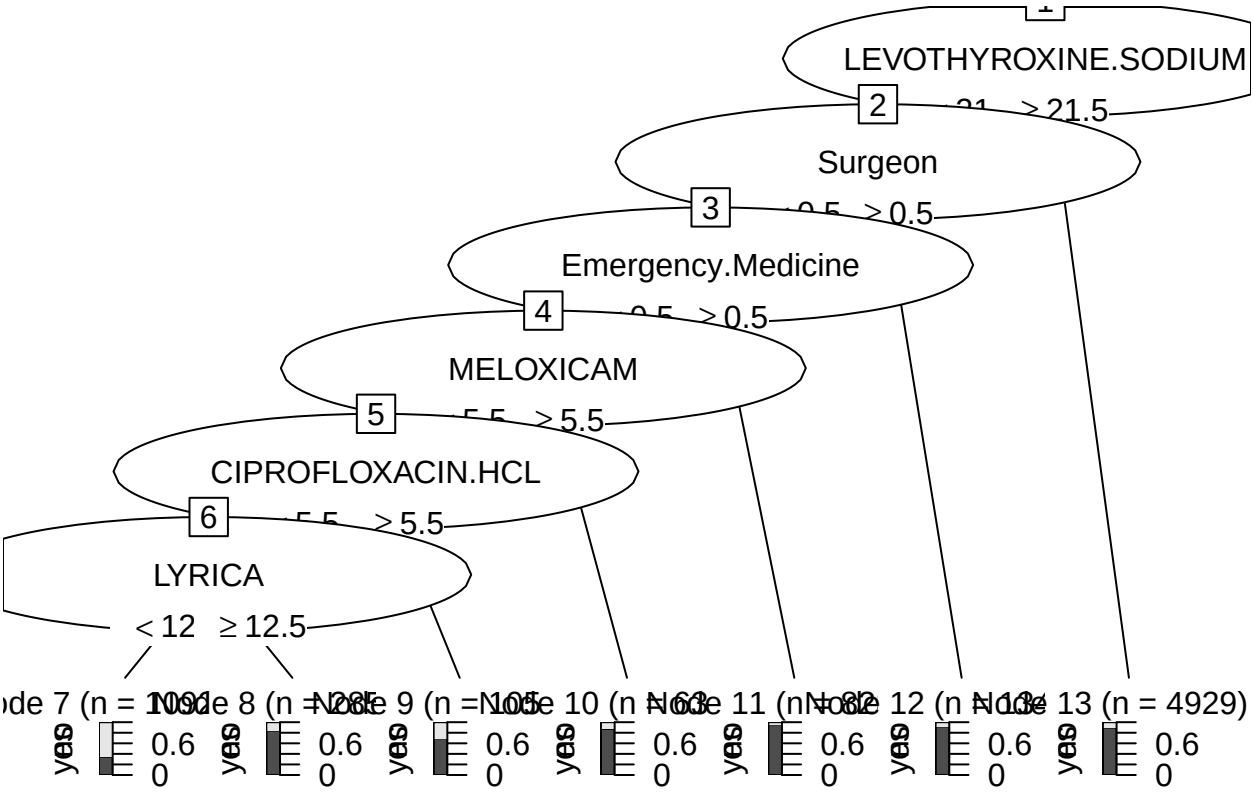
```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich
```




```
library(partykit)
```

```
## Loading required package: libcoin

##
## Attaching package: 'partykit'

## The following objects are masked from 'package:party':
##
##     cforest, ctree, ctree_control, edge_simple, mob, mob_control,
##     node_barplot, node_bivplot, node_boxplot, node_inner,
##     node_surv, node_terminal, varimp
```

```
prescribers_party<-as.party(prescribers_rpart)
plot(prescribers_party)
```



Evaluating Model Performance

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
actual <- prescribers_test$Opioid.Prescriber
predicted <- predict(prescribers_rpart, prescribers_test, type="class")
predicted <- predict(prescribers_rpart, prescribers_test, type="class")
results.matrix <- confusionMatrix(predicted, actual, positive="yes")
print(results.matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  1829  979
##        yes  241 1951
##
##               Accuracy : 0.756
```

```
##                  95% CI : (0.7438, 0.7679)
##     No Information Rate : 0.586
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5221
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6659
##             Specificity : 0.8836
##          Pos Pred Value : 0.8901
##          Neg Pred Value : 0.6514
##              Prevalence : 0.5860
##          Detection Rate : 0.3902
##    Detection Prevalence : 0.4384
##       Balanced Accuracy : 0.7747
##
##        'Positive' Class : yes
##
```

Notice that our accuracy rate is **75.6%**, which is less than **77.16%** in our kNN model (k=158 and using min-max transformation).

Using the Complexity Parameter (CP Value) to Prune the Decision Tree

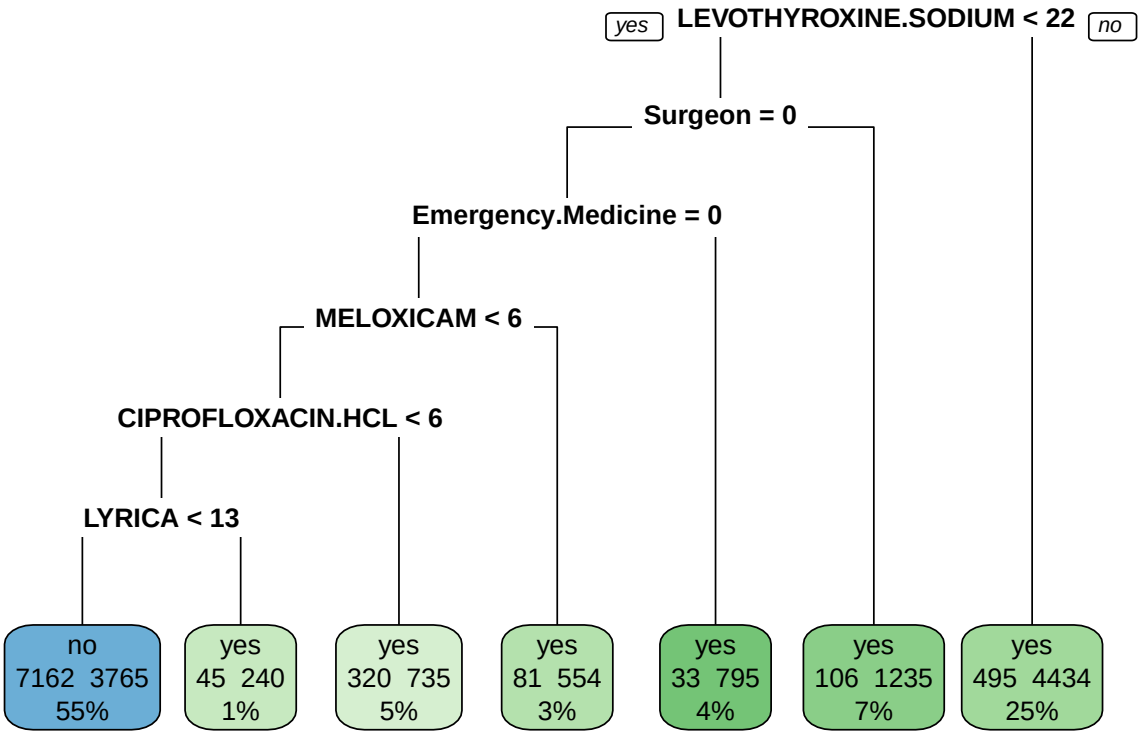
```
cptable<-printcp(prescribers_rpart)
```

```
##
## Classification tree:
## rpart(formula = prescribers_train$Opioid.Prescriber ~ ., data = prescribers_train,
##     method = "class", parms = list(split = "gini"))
##
## Variables actually used in tree construction:
## [1] CIPROFLOXACIN.HCL    Emergency.Medicine   LEVOTHYROXINE.SODIUM
## [4] LYRICA               MELOXICAM            Surgeon
##
## Root node error: 8242/20000 = 0.4121
##
## n= 20000
##
##         CP nsplit rel error  xerror      xstd
## 1 0.094152      0   1.00000 1.00000 0.0084457
## 2 0.092453      2   0.81170 0.88826 0.0082657
## 3 0.057389      3   0.71924 0.71488 0.0078220
## 4 0.050352      4   0.66185 0.67010 0.0076715
## 5 0.023659      5   0.61150 0.62582 0.0075065
## 6 0.010000      6   0.58784 0.60713 0.0074319
```

```
cptable
```

```
##           CP nsplit rel error    xerror        xstd
## 1 0.09415190      0 1.0000000 1.0000000 0.008445696
## 2 0.09245329      2 0.8116962 0.8882553 0.008265709
## 3 0.05738898      3 0.7192429 0.7148750 0.007821976
## 4 0.05035186      4 0.6618539 0.6701043 0.007671481
## 5 0.02365931      5 0.6115021 0.6258190 0.007506531
## 6 0.01000000      6 0.5878428 0.6071342 0.007431881
```

```
set.cp.value<-cptable[which.min(cptable[,"xerror"]),"CP"]

Pruned_prescribers_rpart <- prune(prescribers_rpart, cp=set.cp.value)

rpart.plot(Pruned_prescribers_rpart, type=0, extra=101)
```



Well, that did not do anything! The tree is the same as before. This is due to the fact that the CP value continues to decrease with more splits. Time to try something else!

Visualizing Cross Validation Results

This plots the size of tree (nsplit+1) on top and the complexity parameter at the bottom (x-axis). The red line is the minimum cross validated error (or xerror) + one standard deviation (or xstd).

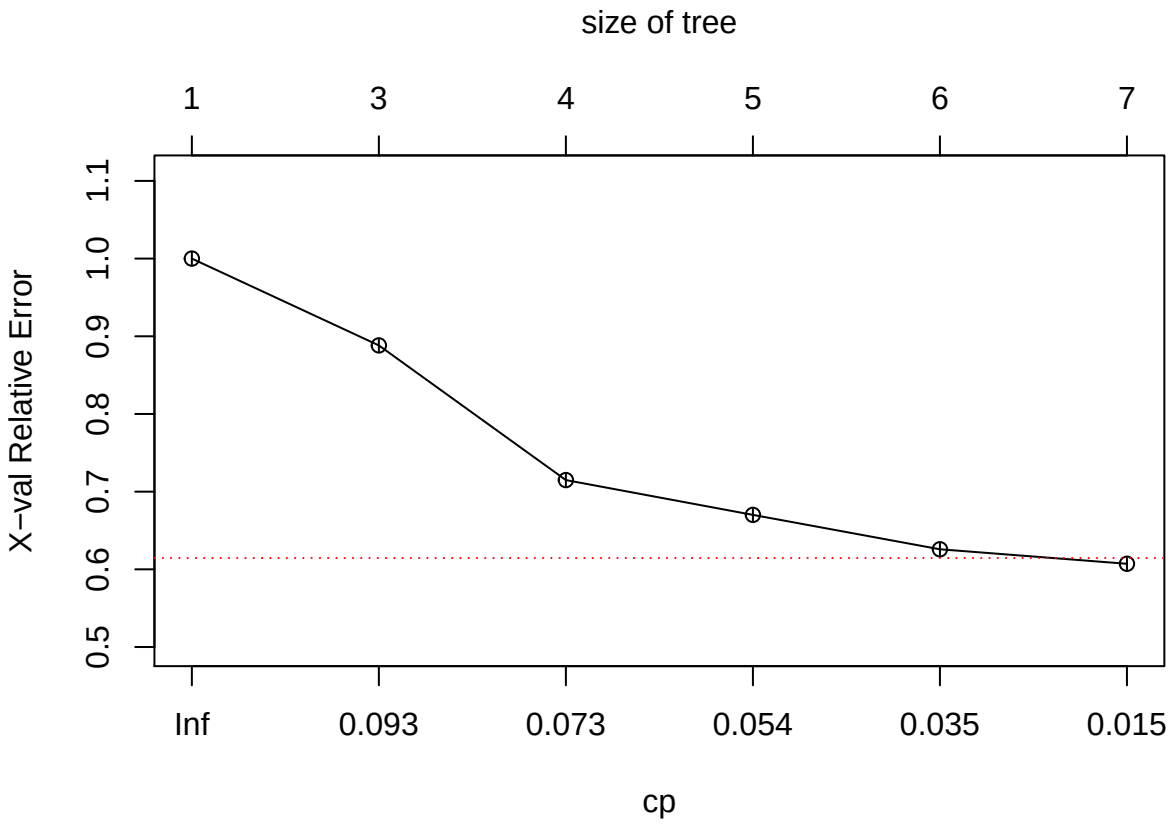
```
cptable<-printcp(prescribers_rpart)
```

```
##
## Classification tree:
## rpart(formula = prescribers_train$Opioid.Prescriber ~ ., data = prescribers_train,
##     method = "class", parms = list(split = "gini"))
##
## Variables actually used in tree construction:
## [1] CIPROFLOXACIN.HCL    Emergency.Medicine   LEVOTHYROXINE.SODIUM
## [4] LYRICA               MELOXICAM            Surgeon
##
## Root node error: 8242/20000 = 0.4121
##
## n= 20000
##
##         CP nsplit rel error  xerror      xstd
## 1 0.094152      0   1.00000 1.00000 0.0084457
## 2 0.092453      2   0.81170 0.88826 0.0082657
## 3 0.057389      3   0.71924 0.71488 0.0078220
## 4 0.050352      4   0.66185 0.67010 0.0076715
## 5 0.023659      5   0.61150 0.62582 0.0075065
## 6 0.010000      6   0.58784 0.60713 0.0074319
```

```
cptable
```

```
##           CP nsplit rel error    xerror        xstd
## 1 0.09415190      0 1.0000000 1.0000000 0.008445696
## 2 0.09245329      2 0.8116962 0.8882553 0.008265709
## 3 0.05738898      3 0.7192429 0.7148750 0.007821976
## 4 0.05035186      4 0.6618539 0.6701043 0.007671481
## 5 0.02365931      5 0.6115021 0.6258190 0.007506531
## 6 0.01000000      6 0.5878428 0.6071342 0.007431881
```

```
plotcp(prescribers_rpart, minline=TRUE, col="red")
```



Picking a Tree Size

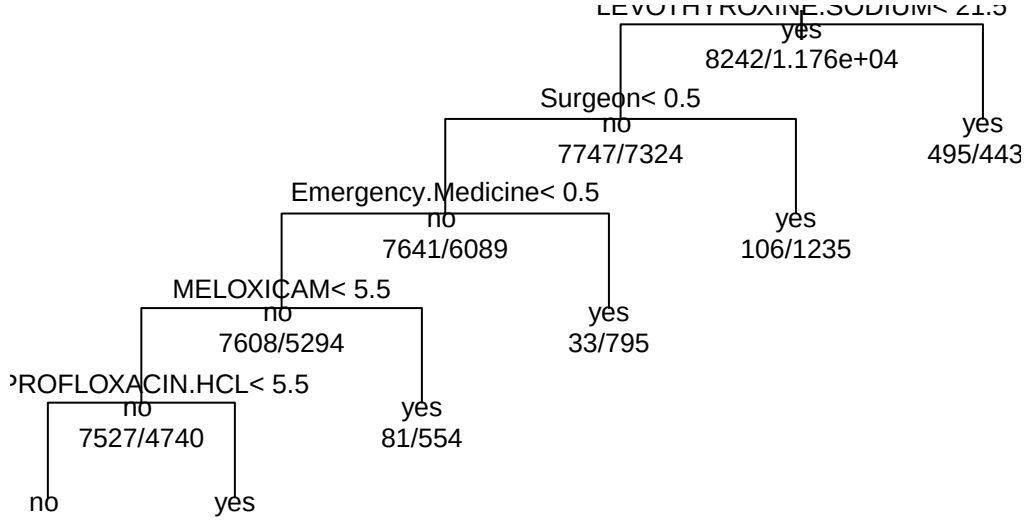
Method 1: Look for the "elbow" in the CP plot. Set the tree size at the cp value where the "elbow" occurs.

Method 2: Pick the cp value that is within one standard deviation of the minimum xerror (the red line).

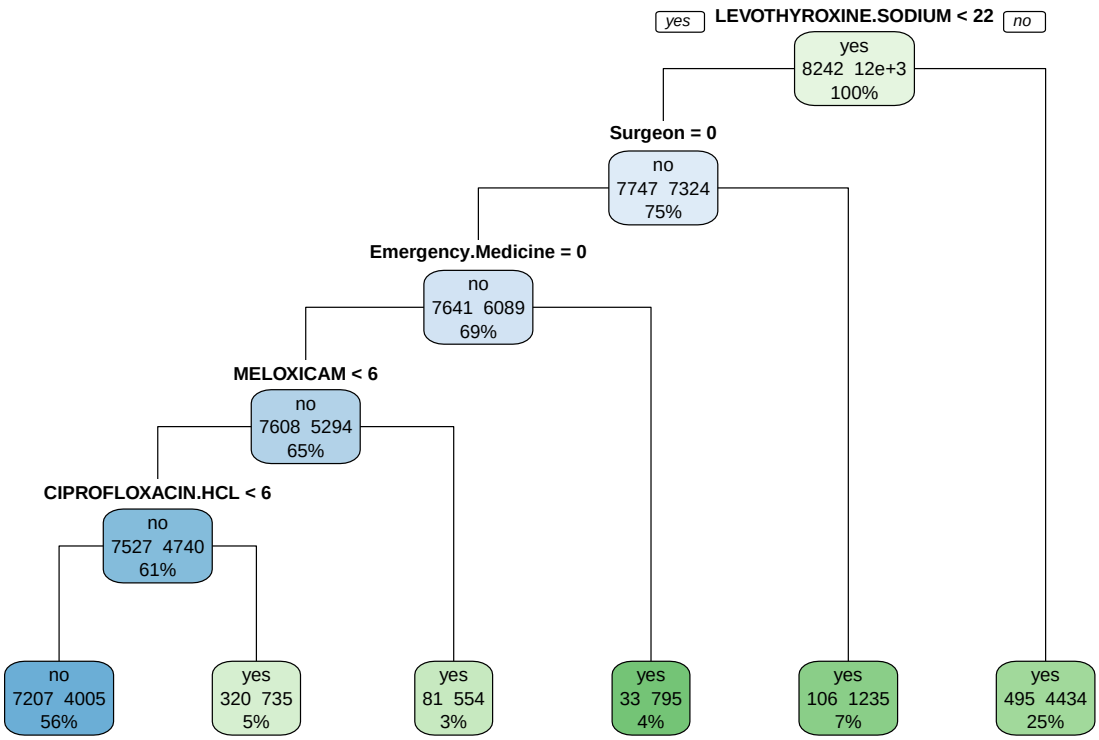
Method 3: Manually prune the tree until desired result is achieved.

```
Pruned_prescribers_rpart <-prune(prescribers_rpart,cp=.05, minsplit=10, minbucket=round(minsplit/3)) # 

plot(Pruned_prescribers_rpart, uniform=TRUE,main="Classification Tree for Opioid Prescribers")
text(Pruned_prescribers_rpart, use.n=TRUE, all=TRUE, cex=.8)
```

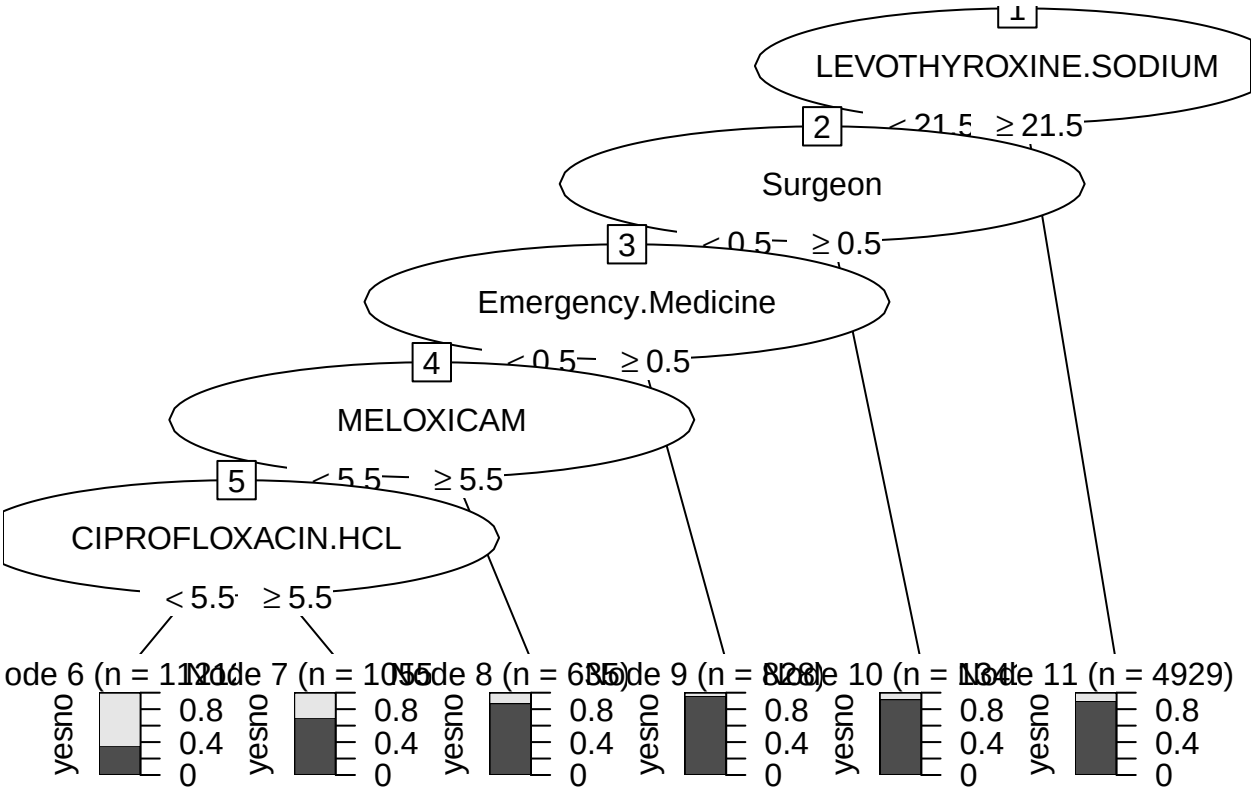
Classification Tree for Opioid Prescribers




```
rpart.plot(Pruned_prescribers_rpart, type=1, extra=101)
```




```
Pruned_prescribers_party<-as.party(Pruned_prescribers_rpart)
plot(Pruned_prescribers_party)
```




```
actual <- prescribers_test$Opioid.Prescriber
predicted <- predict(Pruned_prescribers_rpart, prescribers_test, type="class")
results.matrix <- confusionMatrix(predicted, actual, positive="yes")
print(results.matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  1844 1040
##        yes  226 1890
##
##                Accuracy : 0.7468
##                  95% CI : (0.7345, 0.7588)
##     No Information Rate : 0.586
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5066
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6451
##             Specificity : 0.8908
##          Pos Pred Value : 0.8932
##          Neg Pred Value : 0.6394
##              Prevalence : 0.5860
```

```
##          Detection Rate : 0.3780
##    Detection Prevalence : 0.4232
##       Balanced Accuracy : 0.7679
##
##        'Positive' Class : yes
##
```

Our pruned tree perform nearly as well as our fully grown tree. Accuracy rate is **74.68%** whereas it was previously **75.6%**.

Exploring rpart.control

You should play around with rpart.control to do additional tree pruning.

```
#help(rpart.control)
```