



# STA 235 - Prediction II: Classification and Regression Trees (CART)

Spring 2021

McCombs School of Business, UT Austin

# Some reminders

**Prediction Project has been posted**

Remember: Teams of two (max)

Two tasks: Binary outcome (classification) and continuous outcome (regression)

Need to use two (2) different methods for each

Make sure you read the instructions! (and ask questions)

# Where we've been...

- Talking about **bias vs variance** trade-off.
- **Model selection and regularization**: Stepwise selection, Ridge and Lasso regression.
- **K-nearest neighbors**



# ... and where we're going.



- Continue on our **prediction** journey:
  - **Decision Trees**: Classification and Regression Trees (CART)
- **Participation**: Activity in R.

Trees, trees everywhere!

# Let's start with a simple example

Remember our Disney+ example?

Predict who will cancel their subscription

We have some **information**:

- **city**: Whether the customer lives in a big city or not
- **female**: Whether the customer is female or not
- **age**: Customer's age (in years)
- **logins**: Number of logins to the platform in the past week.
- **mandalorian**: Whether the person has watched the Mandalorian or not.
- **unsubscribe**: Whether they canceled their subscription or not.

# The prediction task: Classification

- Our outcome is **binary**, so this is a **classification task**.
- Let's start looking at **two variables**:

**City & Mandalorian**

- Which one do you think is a better predictor?

**Let's look at the data!**

# City vs. Mandalorian

```
disney <- read.csv("https://raw.githubusercontent.com/maibennett/sta235/main/exampleSite/content/Cla
disney.train <- disney %>% dplyr::filter(train==1)

#Whole data
table(disney.train %>% dplyr::select(unsubscribe))

table(disney.train %>% dplyr::select(city, mandalorian))
```

Subscribers vs Unsubscribers		2x2 Frequency table		
Unsubscribe		Mandalorian		
Freq		No	Yes	City
0		212	500	0
1		1253	3035	1



# City vs. Mandalorian

```
#Subscribers
```

```
table(disney.train %>% dplyr::filter(unsubscribe==0) %>%  
      dplyr::select(city, mandalorian))
```

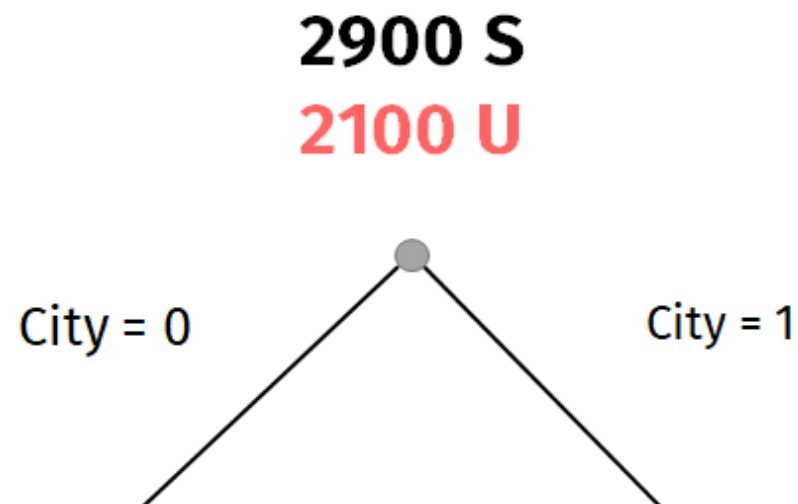
```
#Unsubscribers
```

```
table(disney.train %>% dplyr::filter(unsubscribe==1) %>%  
      dplyr::select(city, mandalorian))
```

Subscribers			Unsubscribers		
City	Mandalorian		City	Mandalorian	
	0	1		0	1
0	173	155	0	39	345
1	1067	1505	1	186	1530

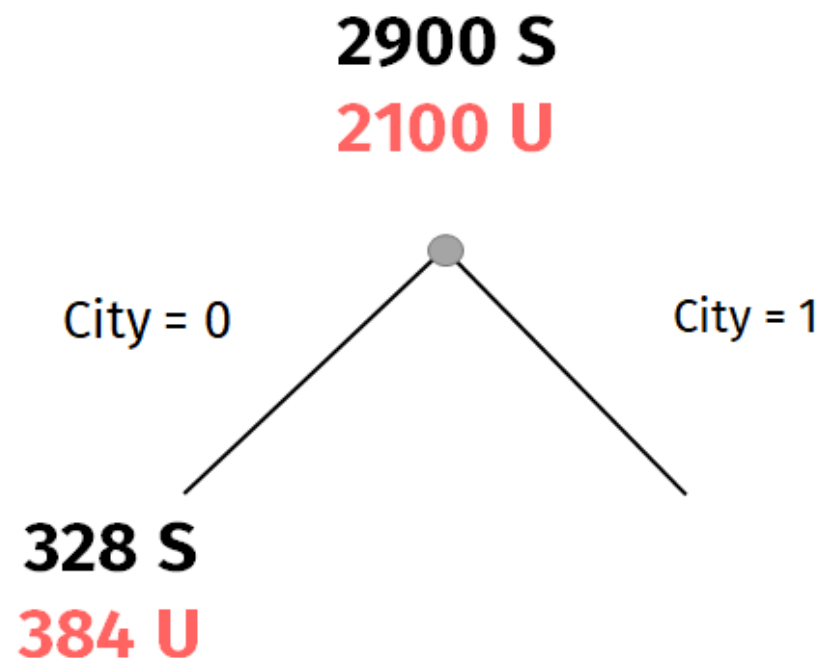
# Let's split by `city` first...

Subscribers			Unsubscribers		
	Mandalorian			Mandalorian	
City	0	1	City	0	1
0	173	155	0	39	345
1	1067	1505	1	186	1530



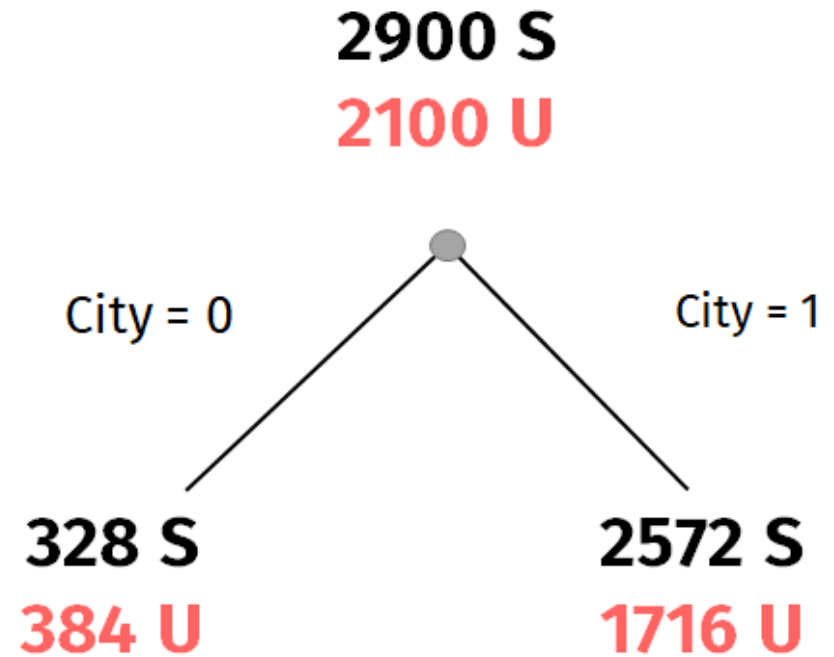
# Let's split by `city` first...

Subscribers			Unsubscribers		
	Mandalorian			Mandalorian	
City	0	1	City	0	1
0	173	155	0	39	345
1	1067	1505	1	186	1530

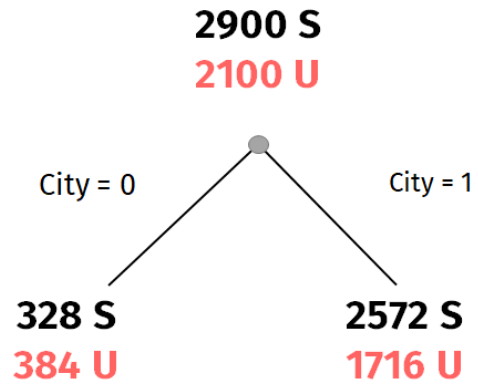


# Let's split by `city` first...

Subscribers			Unsubscribers		
	Mandalorian			Mandalorian	
City	0	1	City	0	1
0	173	155	0	39	345
1	1067	1505	1	186	1530

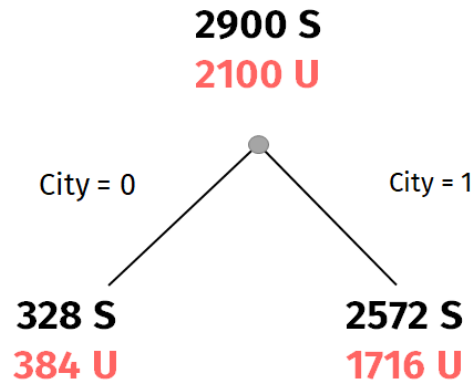


# Calculate probabilities for city



$$\Pr[\text{Correct} \mid \text{city} = 0] = ?$$

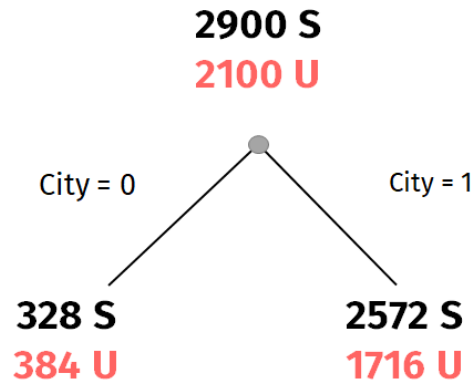
# Calculate probabilities for city



$$\Pr[\text{Correct} \mid \text{city} = 0] = \left(\frac{328}{328 + 384}\right)^2 + \left(\frac{384}{328 + 384}\right)^2$$

$$\Pr[\text{Correct} \mid \text{city} = 0] = 0.503$$

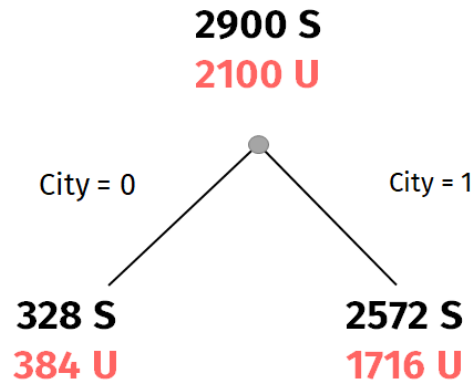
# Calculate probabilities for city



$$\Pr[\text{Correct} \mid \text{city} = 1] = \left(\frac{2572}{2572 + 1716}\right)^2 + \left(\frac{1716}{2572 + 1716}\right)^2$$

$$\Pr[\text{Correct} \mid \text{city} = 1] = 0.52$$

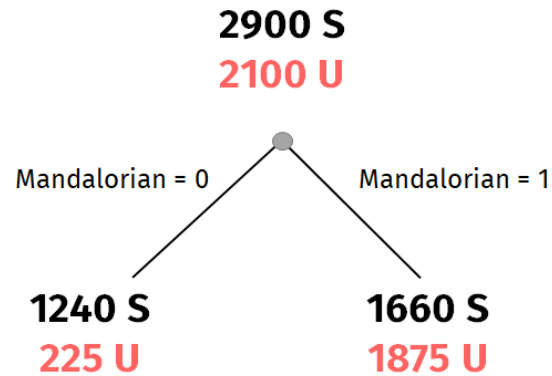
# Calculate probabilities for city



$$\begin{aligned}\Pr[\text{Correct}] &= \Pr[\text{Correct} | \text{city} = 0] \times \Pr[\text{city} = 0] + \\ &\quad \Pr[\text{Correct} | \text{city} = 1] \times \Pr[\text{city} = 1] = \\ &= 0.5 \times \frac{328 + 324}{5000} + 0.52 \times \frac{2572 + 1716}{5000} = \\ &= 0.518\end{aligned}$$



# And we can do the same for mandalorian



$$\begin{aligned}\Pr[\text{Correct}] &= \Pr[\text{Correct} | \text{mandlr} = 0] \times \Pr[\text{mandlr} = 0] + \\ &\quad \Pr[\text{Correct} | \text{mandlr} = 1] \times \Pr[\text{mandlr} = 1] = \\ &= 0.74 \times \frac{1240 + 225}{5000} + 0.502 \times \frac{1660 + 1875}{5000} = \\ &= 0.572\end{aligned}$$

**Poll Time!**

**Which variable would you choose  
for prediction?**

# Choosing predictors

- From the previous exercise, we can see that **using mandalorian yields a higher accuracy** (0.57 vs. 0.52)

**But we have more variables**

**How do we choose?**

# Decision Trees

- Main idea → **flowchart!**
- We will **stratify** (or segment) the predictor space into regions (ISLR, Ch. 8).
- Similar to KNN, we assign the **mean** or **mode** of the training obs in the region.

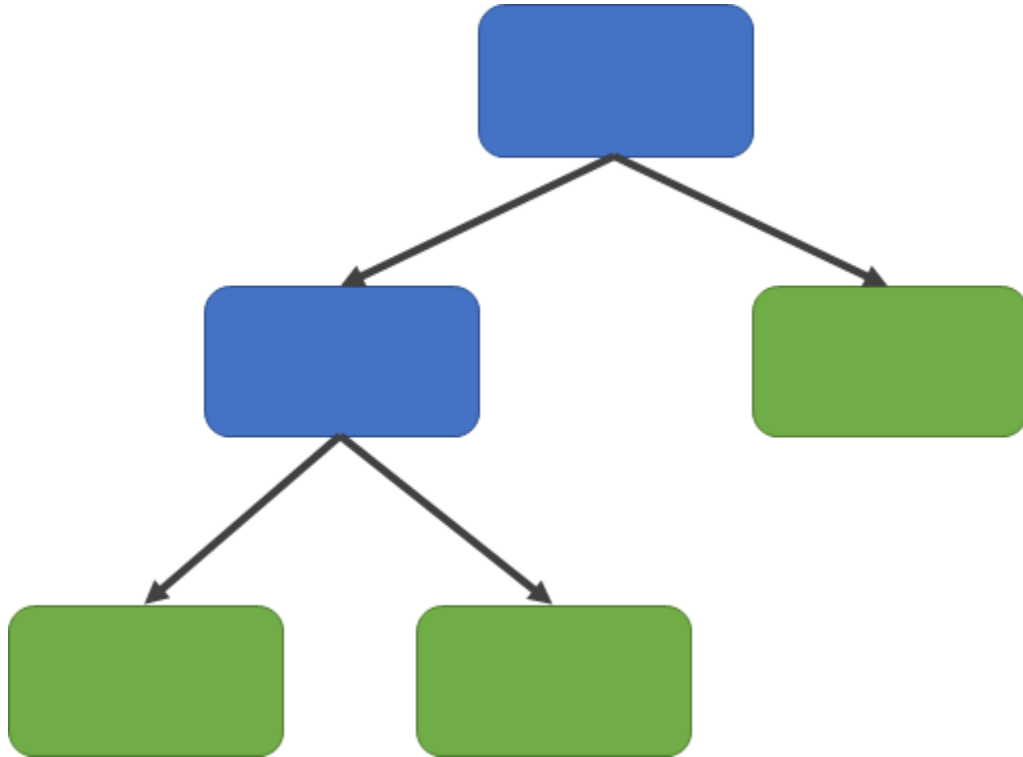
## Main advantages

Simple interpretation

## Main disadvantages

Overfitting

# Structure of Decision Trees



## Structure:

- Root node
- Internal nodes
- Leaves

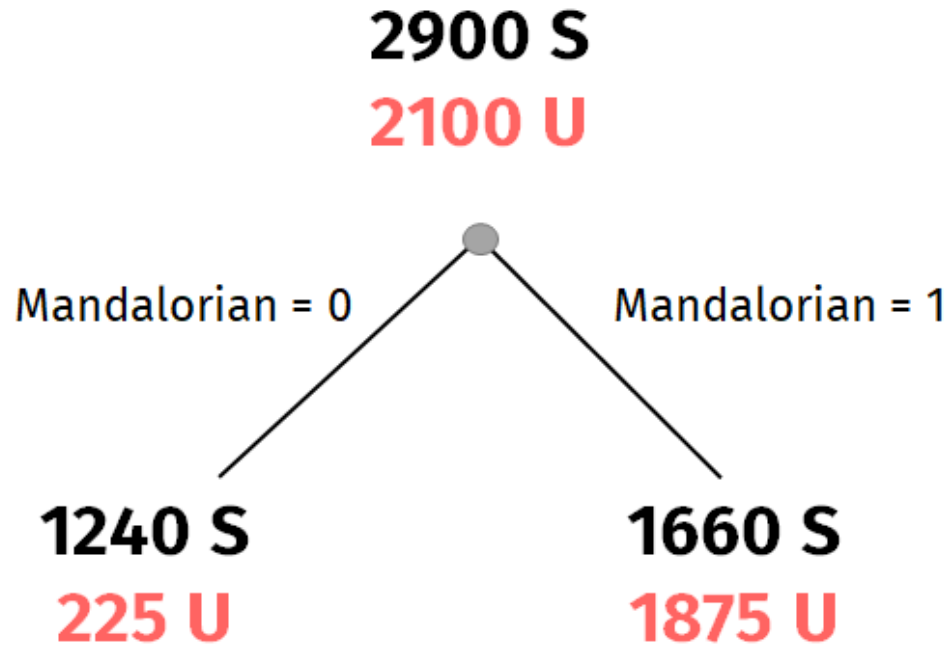
# Classification Trees

# Classification Tree

- Outcome is **categorical** (e.g. binary)
- Previous example: Chose splitting variable based on  $\text{Pr}(\text{Correct})$
- What if we just assigned **based on the proportion in each leave?** (i.e. similar to KNN)

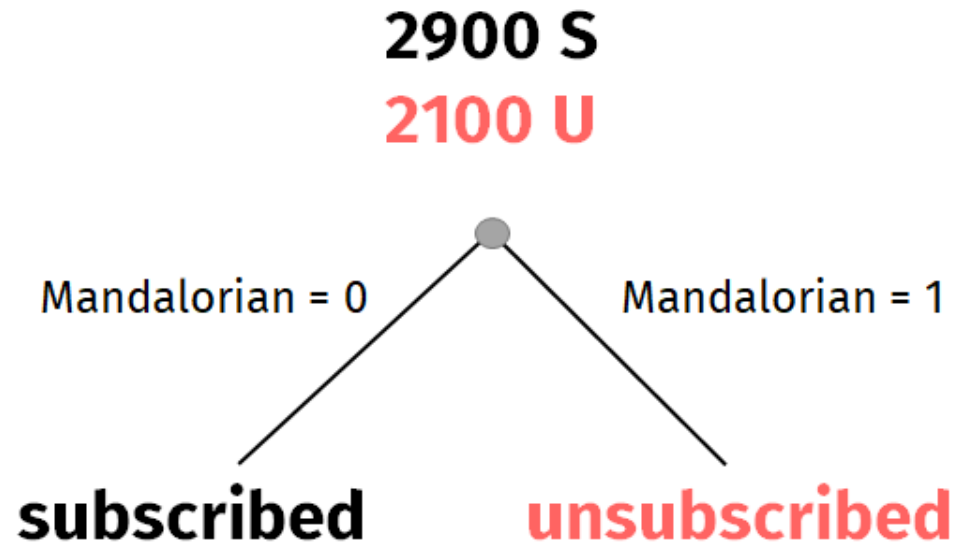


# Let's go back to our drawing



- $\Pr[U \mid \text{Leaf 1}] = 0.15$
- $\Pr[U \mid \text{Leaf 2}] = 0.53$

# Let's go back to our drawing



- $\Pr[U \mid \text{Leaf 1}] = 0.15$
- $\Pr[U \mid \text{Leaf 2}] = 0.53$

**Classification error: 42%**

# Measures for accuracy

- The classification error rate is **not very sensitive for tree-growing**.

**Poll time!**

**What is the main problem if our  
measure is not very sensitive for  
tree-growing?**

# Measures for accuracy

- The classification error rate is **not very sensitive for tree-growing**.
- Another measure is called **Gini index**:
  - Total variance across classes:

$$G = \sum_{k=1}^K \hat{p}_{mk}(1 - \hat{p}_{mk})$$

where  $\hat{p}_{mk}$  is the proportion of training obs in region  $m$  for class  $k$ .

- In our previous example: `

$$G_{mandalorian=0} = \frac{1240}{1240 + 225} \left(1 - \frac{1240}{1240 + 225}\right) + \frac{225}{1240 + 225} \left(1 - \frac{225}{1240 + 225}\right) = 0.26$$

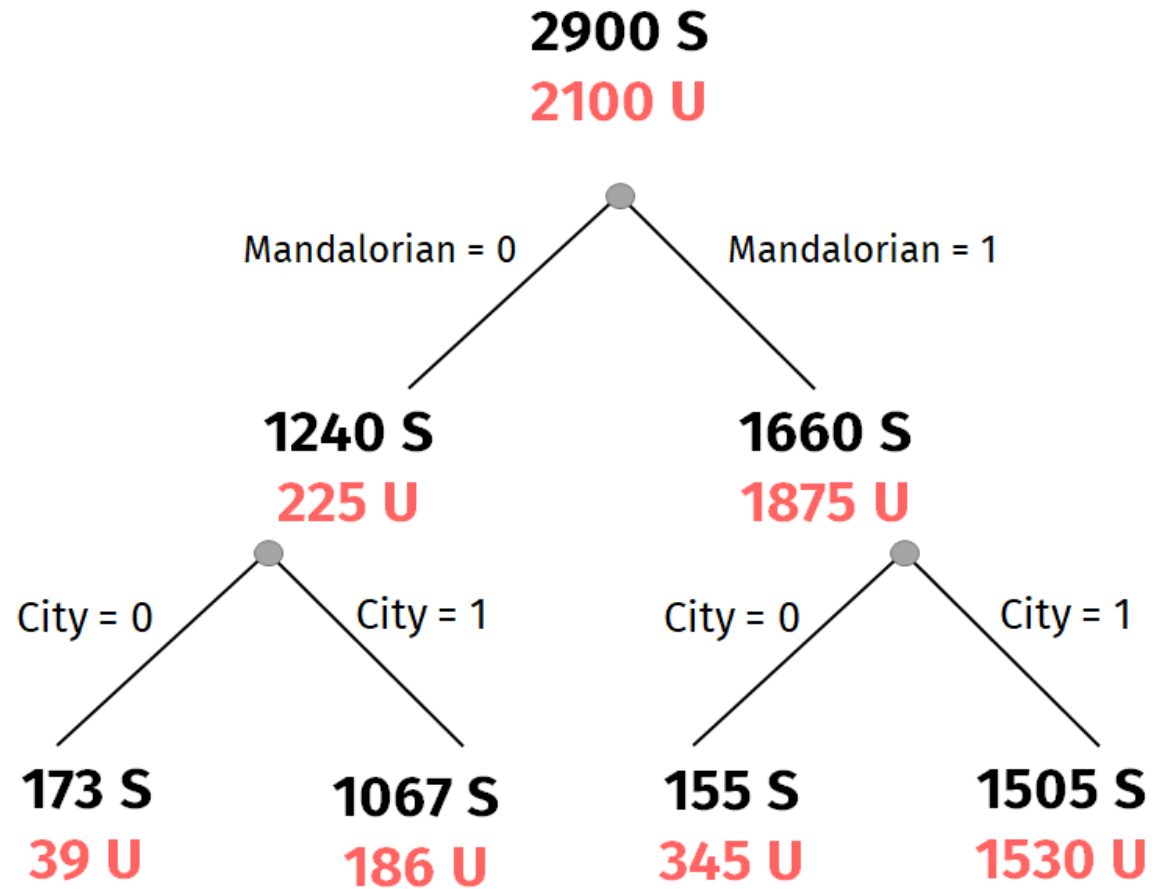
**Poll time!**

According to the Gini Index, is it better or worse to have a high  $p_{mk}$ ?

$$G = \sum_{k=1}^K \hat{p}_{mk}(1 - \hat{p}_{mk})$$



# Ok, but how about including other variables?



# Let's see how to do it in R!

```
library(rpart)
```

```
d.train <- disney.train %>% dplyr::select(mandalorian, city, unsubscribe)
```

```
set.seed(100)
```

```
m1 <- rpart(unsubscribe ~., data = d.train, method = "class", cp=-1)
```

# Let's see how to do it in R!

```
library(rpart)

d.train <- disney.train %>% dplyr::select(mandalorian, city, unsubscribe)

set.seed(100)

m1 <- rpart(unsubscribe ~., data = d.train, method = "class", cp=-1)
```

# Let's see how to do it in R!

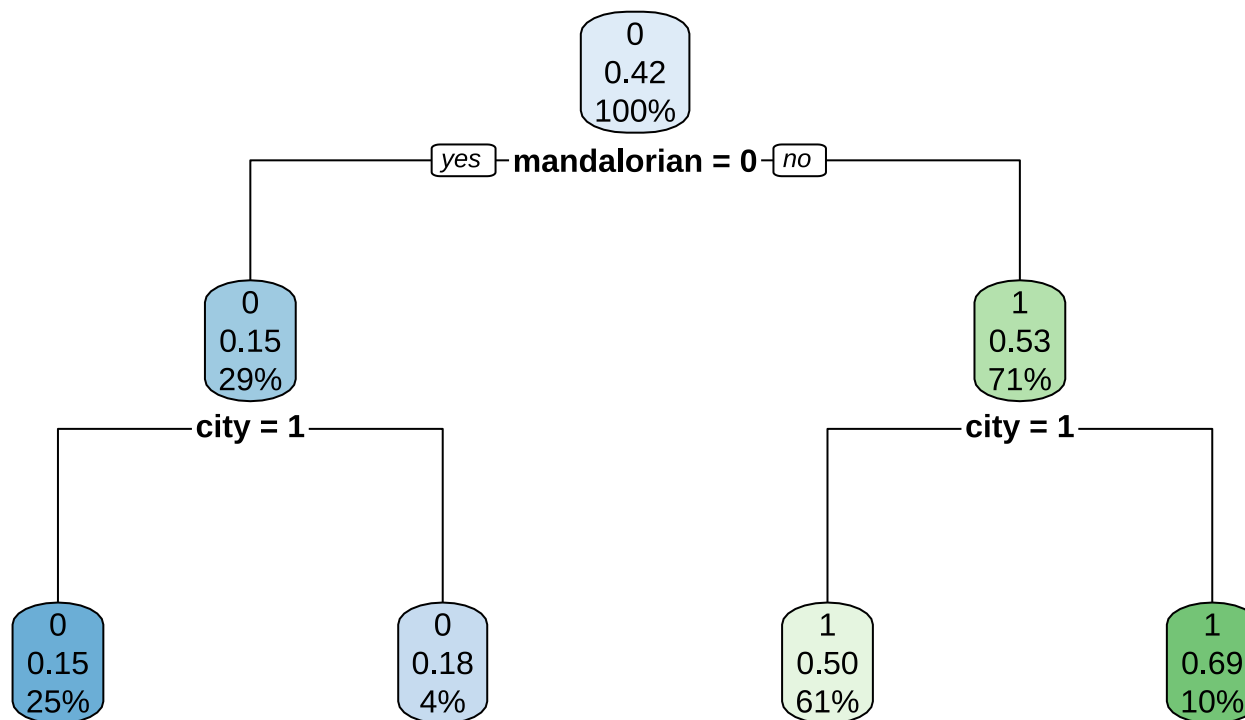
```
library(rpart)

d.train <- disney.train %>% dplyr::select(mandalorian, city, unsubscribe)

set.seed(100)

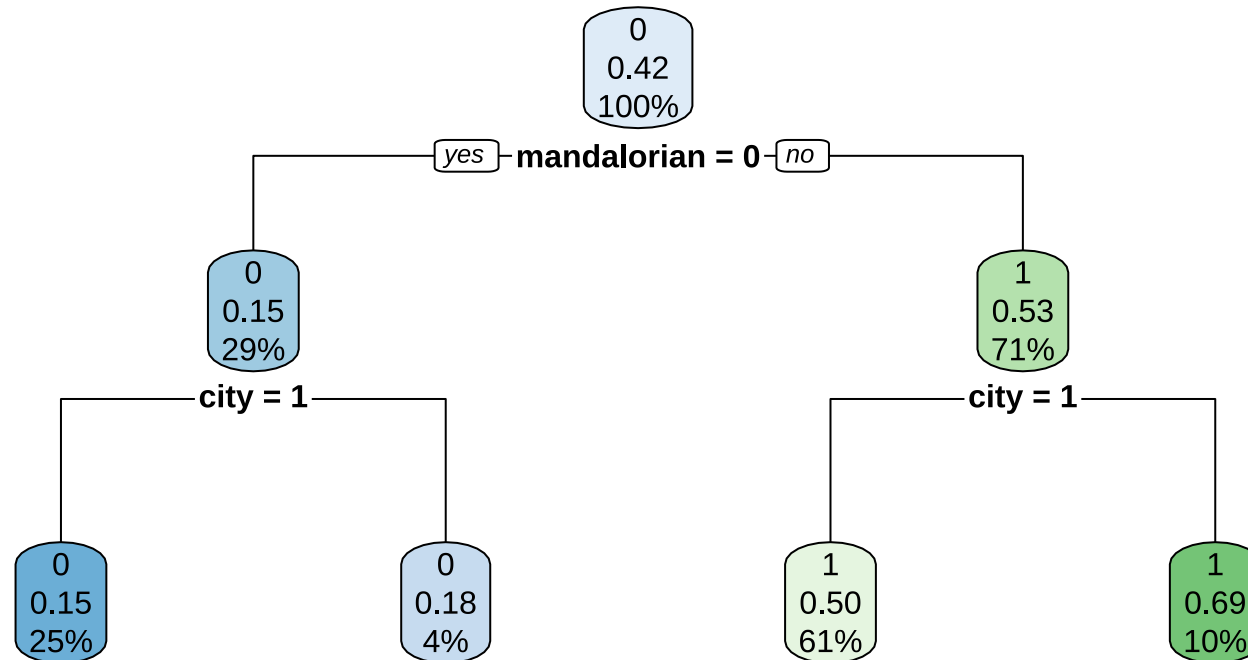
m1 <- rpart(unsubscribe ~., data = d.train, method = "class", cp=-1)
```

# Fully-grown tree



**Poll time!**

# What do you think the percentages in the leaves represent?



# Some parameters that might be important

```
m1 <- rpart(unsubscribe ~., data = d.train, method = "class",  
           control = rpart.control(cp=-1,  
                                   minsplit = 20))
```

- **cp**: Complexity parameter
  - Split must decrease the overall lack of fit by a factor of **cp**, or is not attempted.
  - Parameter for **pruning the tree**.
  - Higher **cp**, smaller the tree!

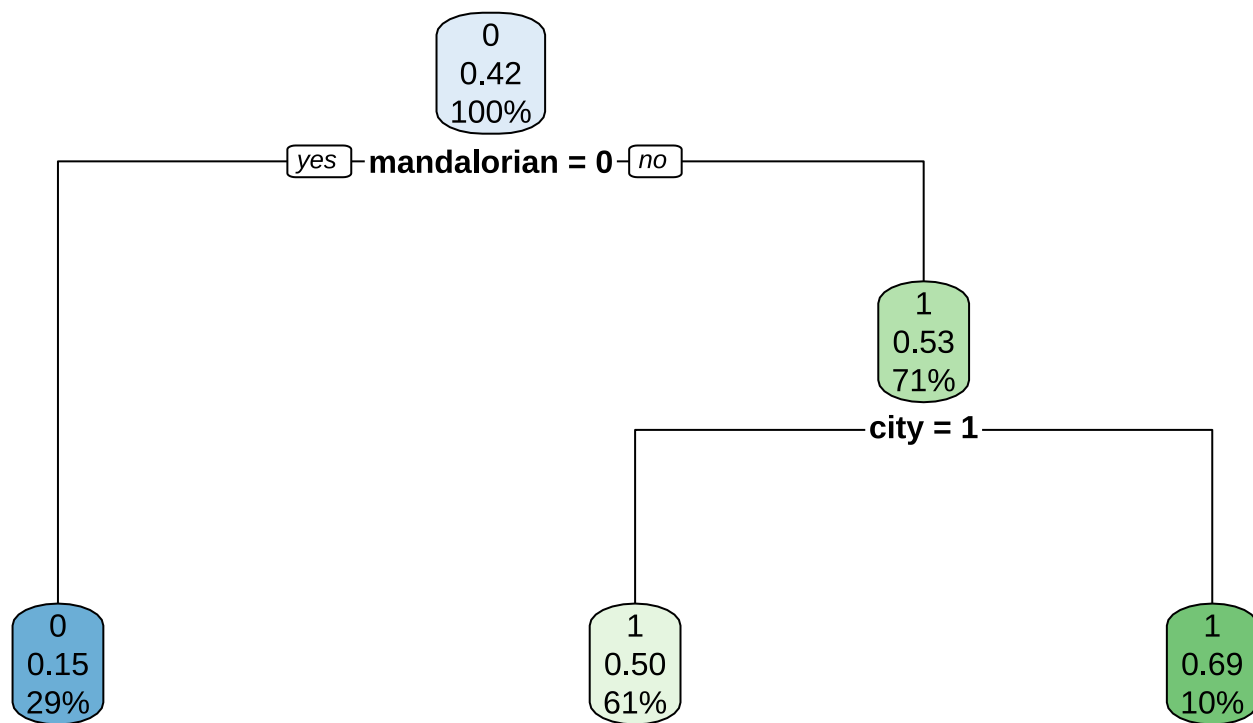


# Some parameters that might be important

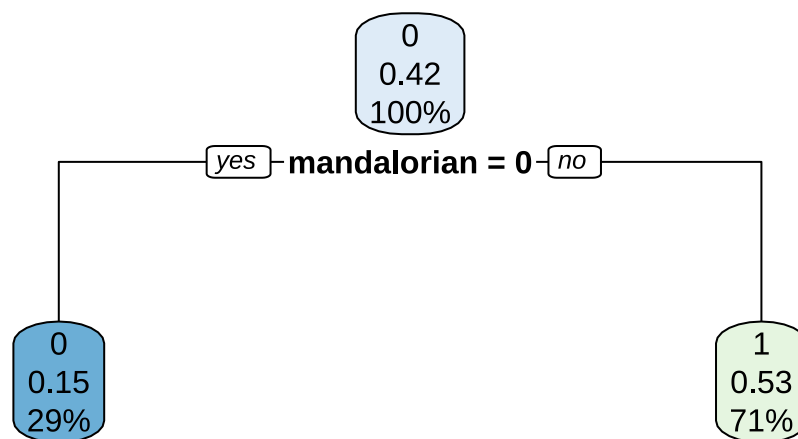
```
m1 <- rpart(unsubscribe ~., data = d.train, method = "class",  
           control = rpart.control(cp=-1,  
                                   minsplit = 20))
```

- `minsplit`: Min. number of obs in a node to attempt a split.

# If we set `minsplit` to 1500...



# If we don't set $C_p$ ...



# If we don't set `cp`...

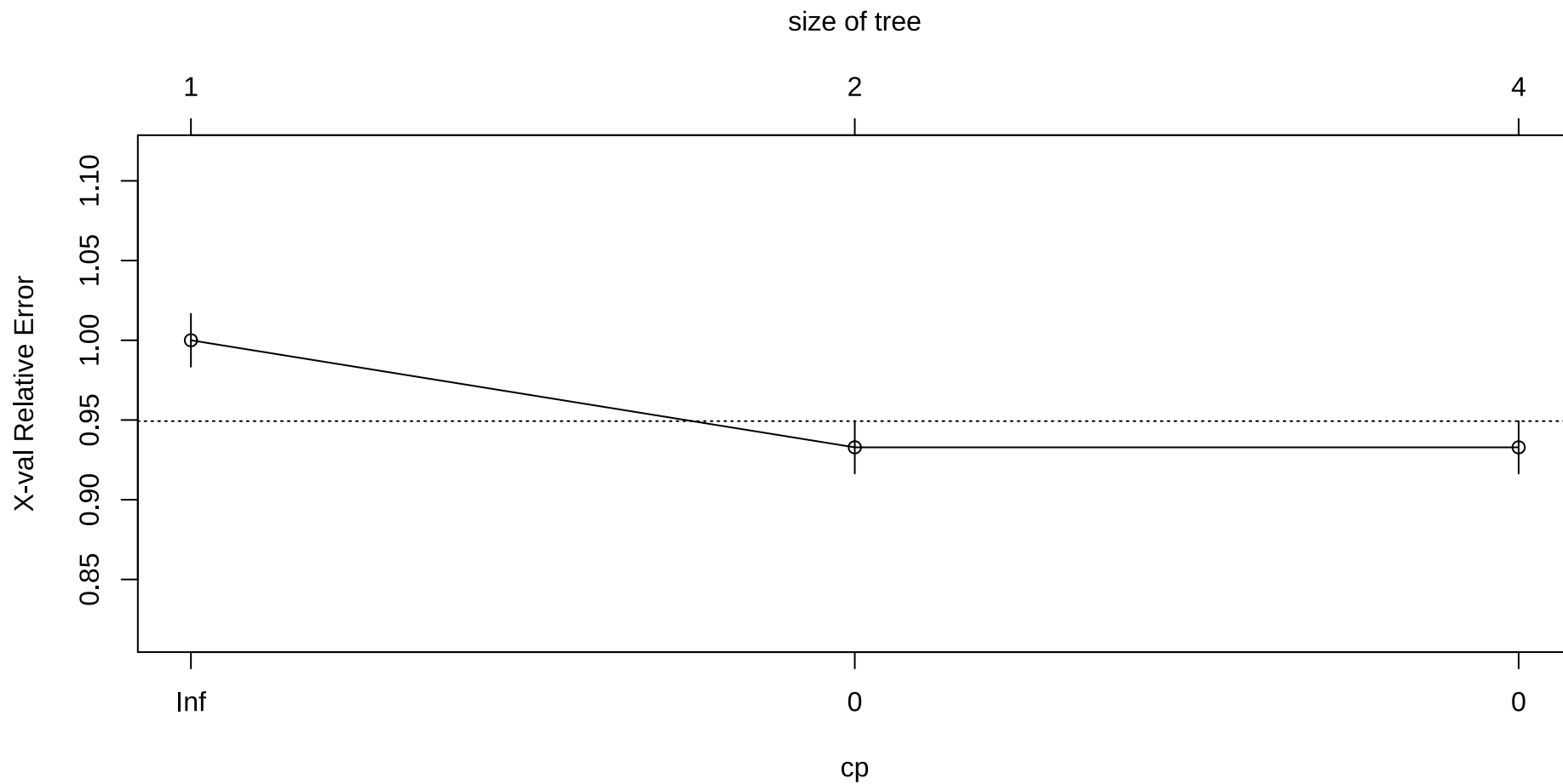
```
m1$cpstable
```

```
##           CP nsplit rel error      xerror      xstd
## 1  0.102381      0  1.000000  1.0000000  0.01661898
## 2  0.000000      1  0.897619  0.9328571  0.01643695
## 3 -1.000000      3  0.897619  0.9328571  0.01643695
```

```
m3$cpstable
```

```
##           CP nsplit rel error      xerror      xstd
## 1 0.102381      0  1.000000  1.0000000  0.01661898
## 2 0.010000      1  0.897619  0.897619  0.01631851
```

# How can we use this for selecting the size of our tree?



# Basic Algorithm

**1) Start at the root node**

**2) Split the parent node at covariate  $x_i$  to minimize the sum of child node impurities**

**3) Assign training samples to new child nodes**

**4) Stop if leaves are pure or early stopping criteria is satisfied, else repeat step (1) and (2) for each new child nodes**

**Now it's your turn!**

# Instructions

- Using the code **provided on the course's website**:
  - Fit a classification tree using **all the covariates**

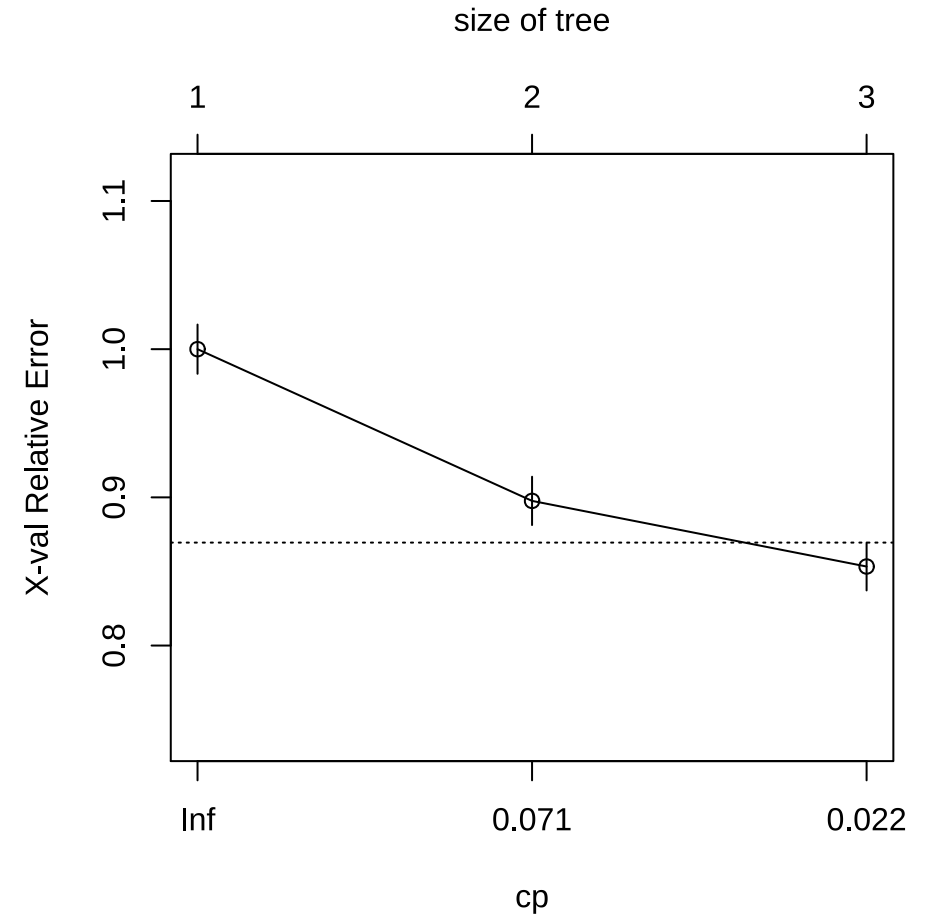
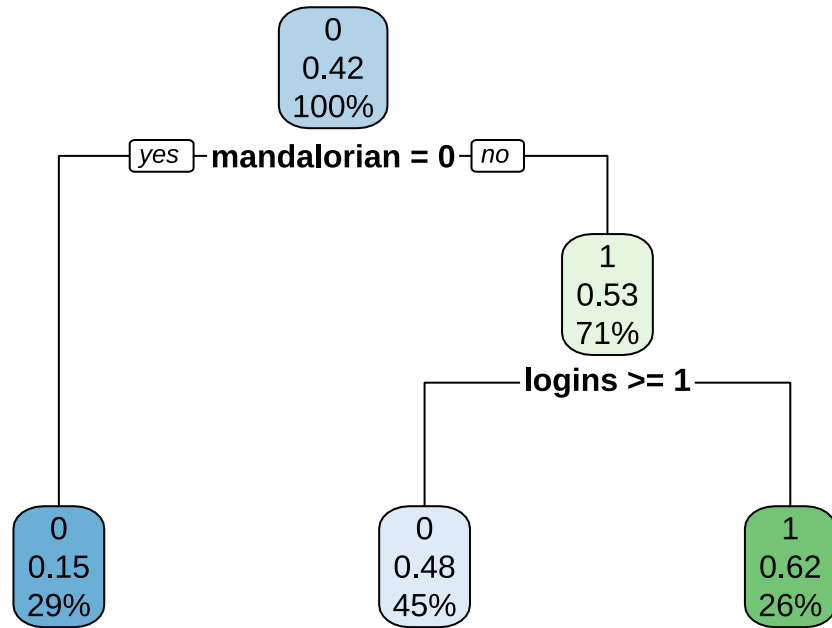
Play around with `cp`. What happens?

Interpret your tree!



# Results

```
set.seed(100)
mex <- rpart(unsubscribe ~., data = disney.trai
              method = "class", cp = 0.01)
rpart.plot(mex)
```



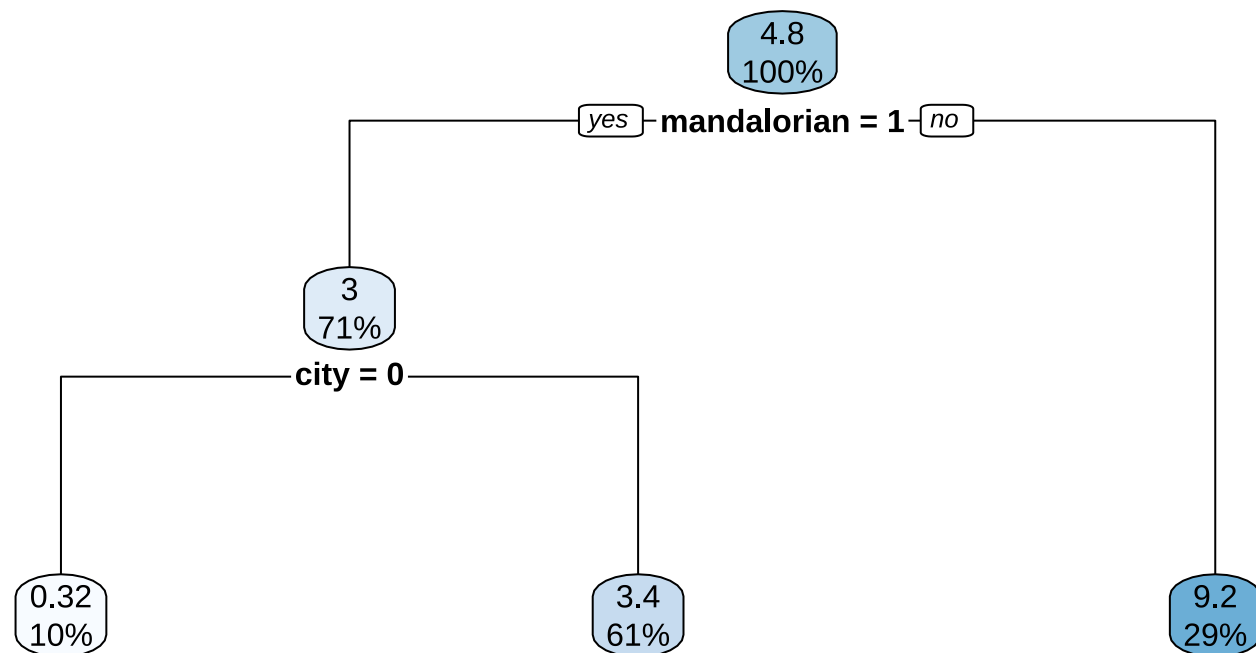
# Regression Trees

# Regression Trees

- Outcome is **continuous**
- Very similar to what we have seen with **classification trees**:
  - Predicted outcome is the **mean outcome for the leaf/region**.

# In R is basically the same

```
set.seed(100)
r1 <- rpart(logins ~. - unsubscribe, data = disney.train,
            method = "anova")
rpart.plot(r1)
```

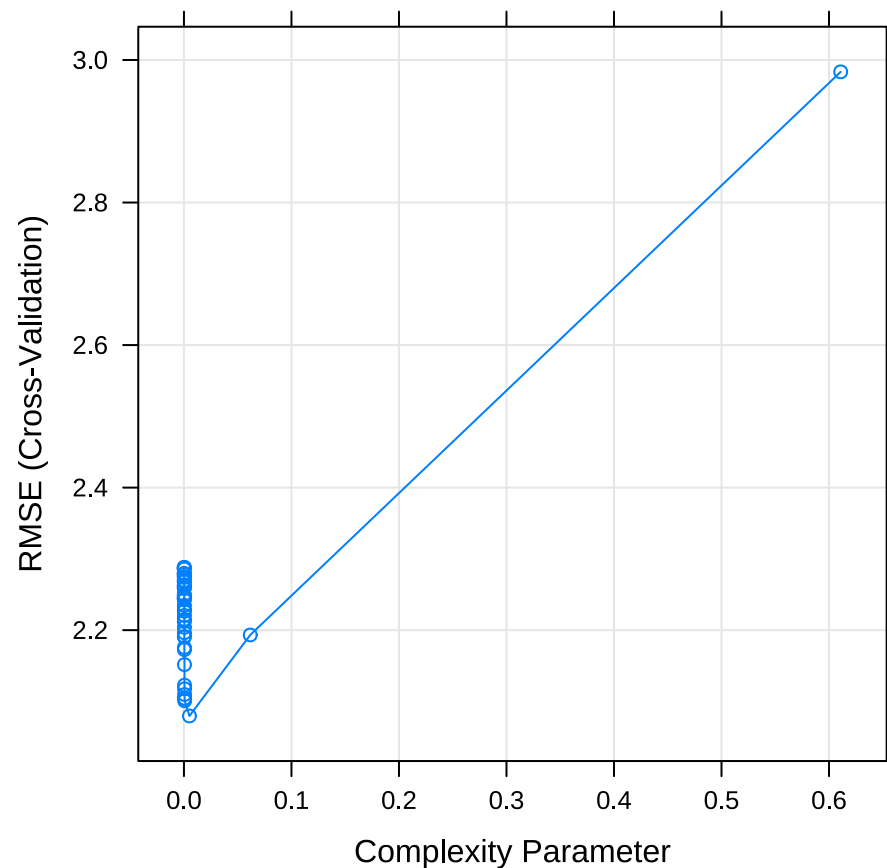


# Let's incorporate cross-validation!

```
library(caret)
set.seed(100)

mcv <- train(
  logins ~. - unsubscribe, data = disney.train,
  method = "rpart",
  trControl = trainControl("cv", number = 10),
  tuneLength = 50
)

plot(mcv)
```



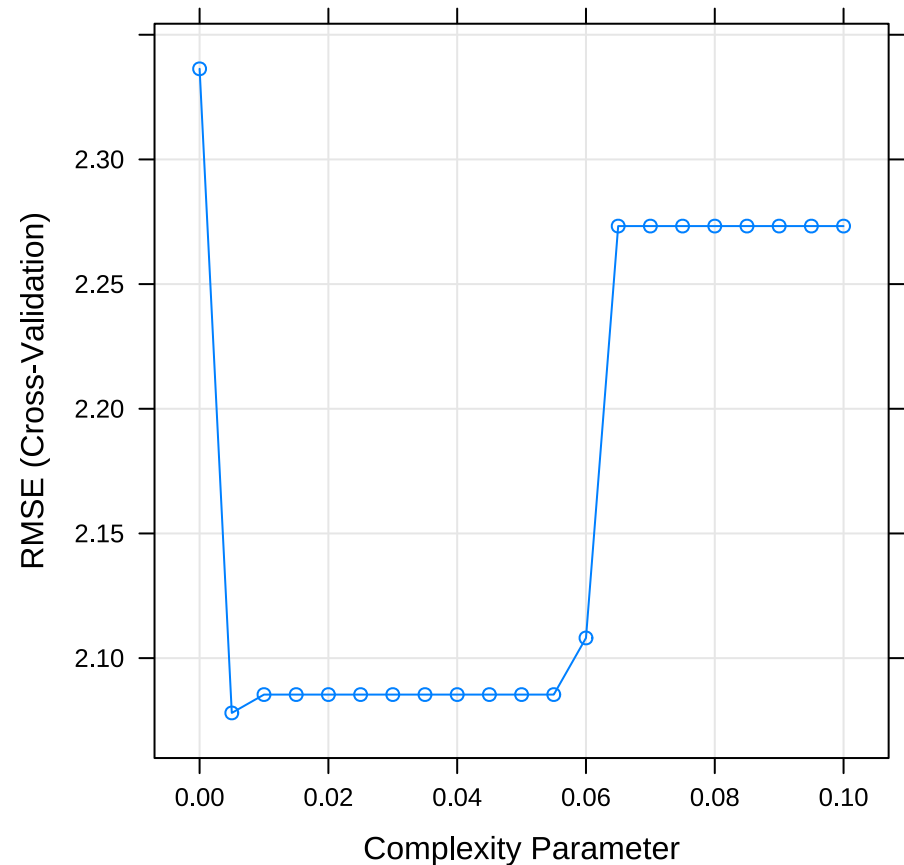
# Let's incorporate cross-validation!

```
library(caret)
set.seed(100)

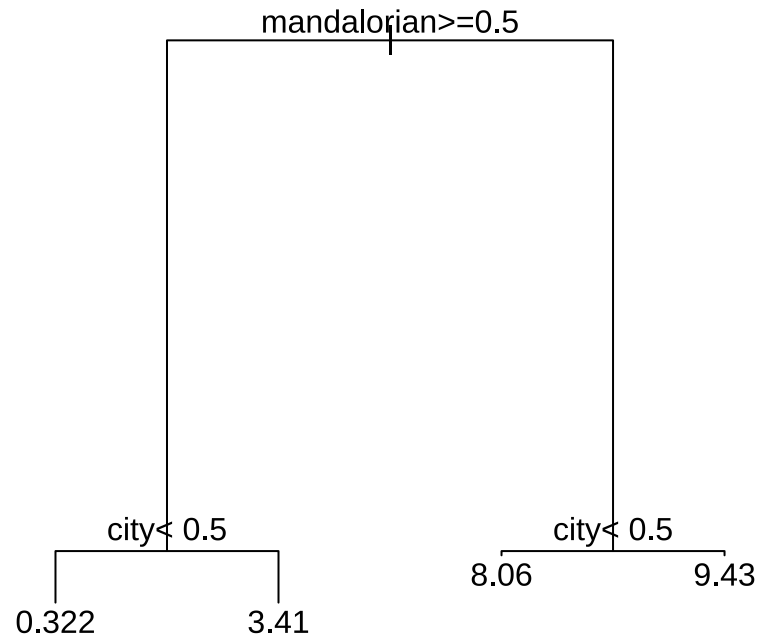
tuneGrid <- expand.grid(cp = seq(0, 0.1, 0.005)

mcv <- train(
  logins ~. - unsubscribe, data = disney.train
  method = "rpart",
  trControl = trainControl("cv", number = 10),
  tuneGrid = tuneGrid
)

plot(mcv)
```



# Plot the tree



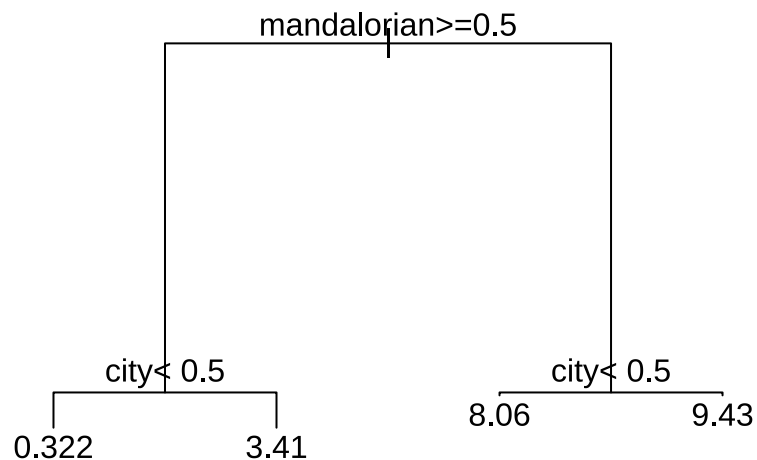
```
mcv$finalModel
```

```
## n= 5000
##
## node), split, n, deviance, yval
##      * denotes terminal node
##
## 1) root 5000 66387.3700 4.806800
##    2) mandalorian >= 0.5 3535 24633.5000 2.973409
##      4) city < 0.5 500 517.1580 0.322000 *
##      5) city >= 0.5 3035 20022.2800 3.410214 *
##    3) mandalorian < 0.5 1465 1200.0180 9.230717
##      6) city < 0.5 212 132.2028 8.061321 *
##      7) city >= 0.5 1253 728.8571 9.428571 *
```

**Poll time!**



# What would the predicted value be for a customer who hasn't watched The Mandalorian and lives in a city?



```
mcv$finalModel
```

```
## n= 5000
##
## node), split, n, deviance, yval
##      * denotes terminal node
##
## 1) root 5000 66387.3700 4.806800
##    2) mandalorian >= 0.5 3535 24633.5000 2.973409
##      4) city < 0.5 500 517.1580 0.322000 *
##      5) city >= 0.5 3035 20022.2800 3.410214 *
##    3) mandalorian < 0.5 1465 1200.0180 9.230717
##      6) city < 0.5 212 132.2028 8.061321 *
##      7) city >= 0.5 1253 728.8571 9.428571 *
```

# Remember that we care about predictions outside our training sample

- For predicting **unsubscribe**

```
disney.test <- disney %>% dplyr::filter(train=0)

mclass <- train(
  factor(unsubscribe) ~., data = disney.train,
  method = "rpart",
  trControl = trainControl("cv", number = 10),
  tuneLength = 50
)

pred.class <- mclass %>% predict(disney.test)

mean(pred.class==disney.test$unsubscribe)
```

```
## [1] 0.639
```

- For predicting **logins**:

```
pred.reg <- mcv %>% predict(disney.test)

RMSE(pred.reg, disney.test$logins)
```

```
## [1] 2.099631
```

# Main takeaways of decision trees



## Main advantages:

- Easy to interpret and explain (you can plot them!)
- Mirrors human decision-making.
- Can handle qualitative predictors (without need for dummies).

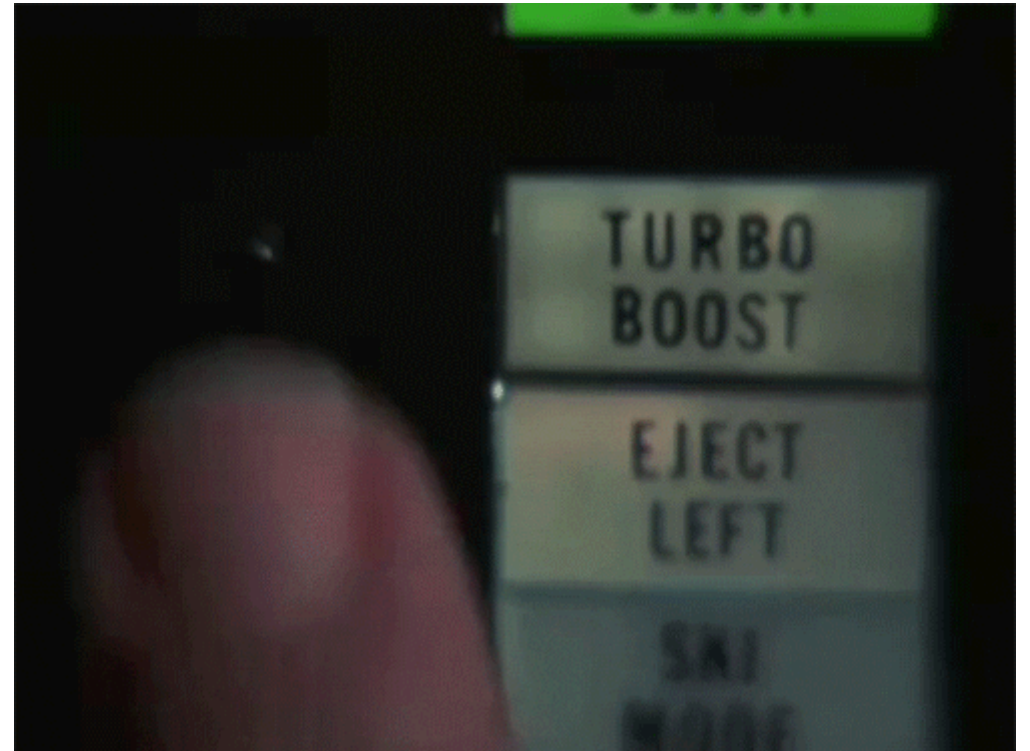
## Main disadvantages:

- Accuracy not as high as other methods
- Very sensitive to training data (e.g. overfitting)

# Next class

Use of decision trees as building blocks for **more powerful prediction methods!**

- Bagging
- Random Forests
- Boosting



# References

- James, G. et al. (2013). "Introduction to Statistical Learning with Applications in R". *Springer. Chapter 8*.
- Ritvik Kharkar. (2019). "Decision Trees". *Video materials from ritvikmath (YouTube)*.
- Starmer, J.. (2018). "Decision Trees". *Video materials from StatQuest (YouTube)*.
- STDHA. (2018). "CART Model: Decision Tree Essentials"