Classification And Regression Tree

How Do We Take Decisions?

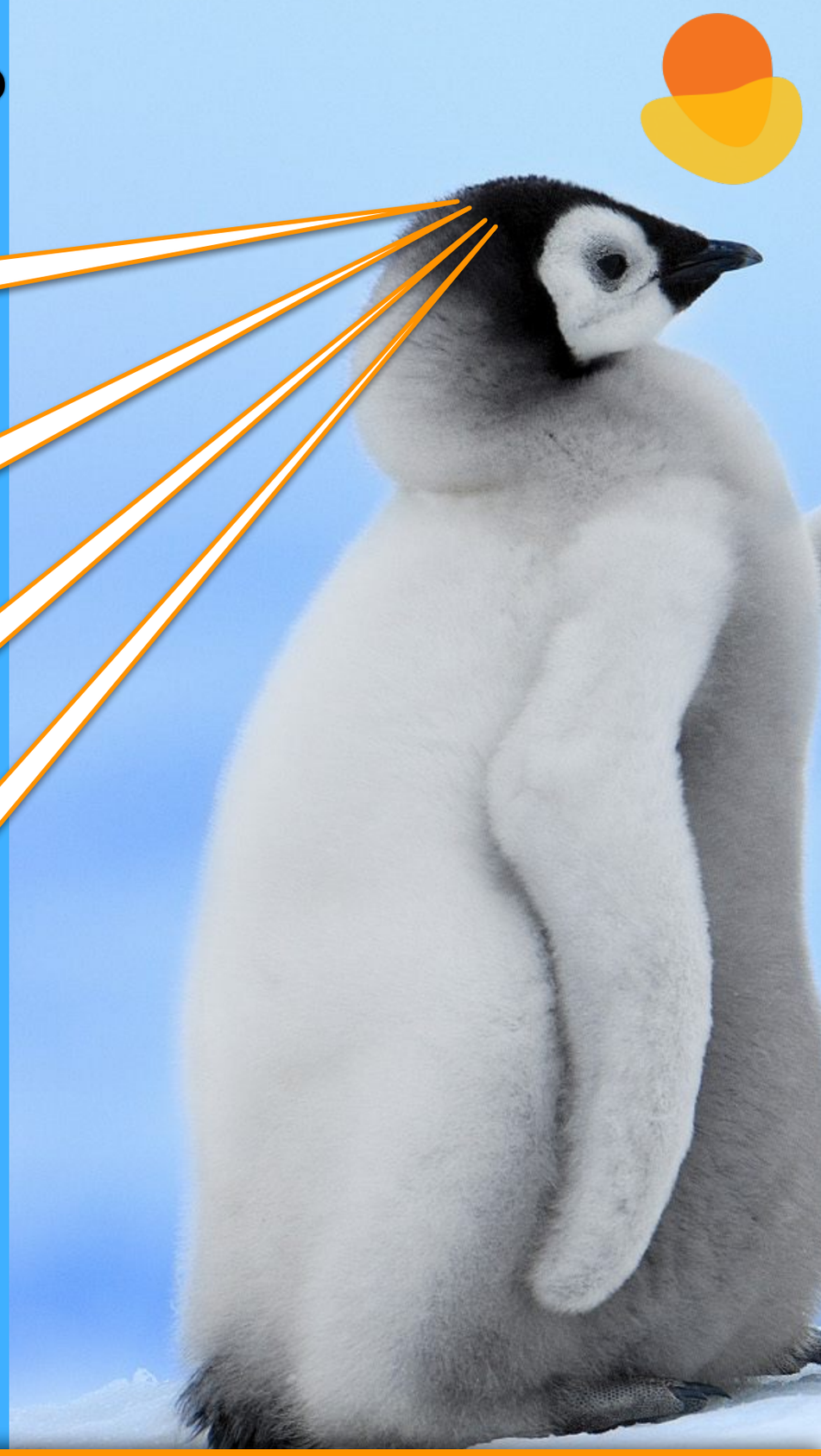
How Do We Take Decisions?

Do I want to watch a movie now?

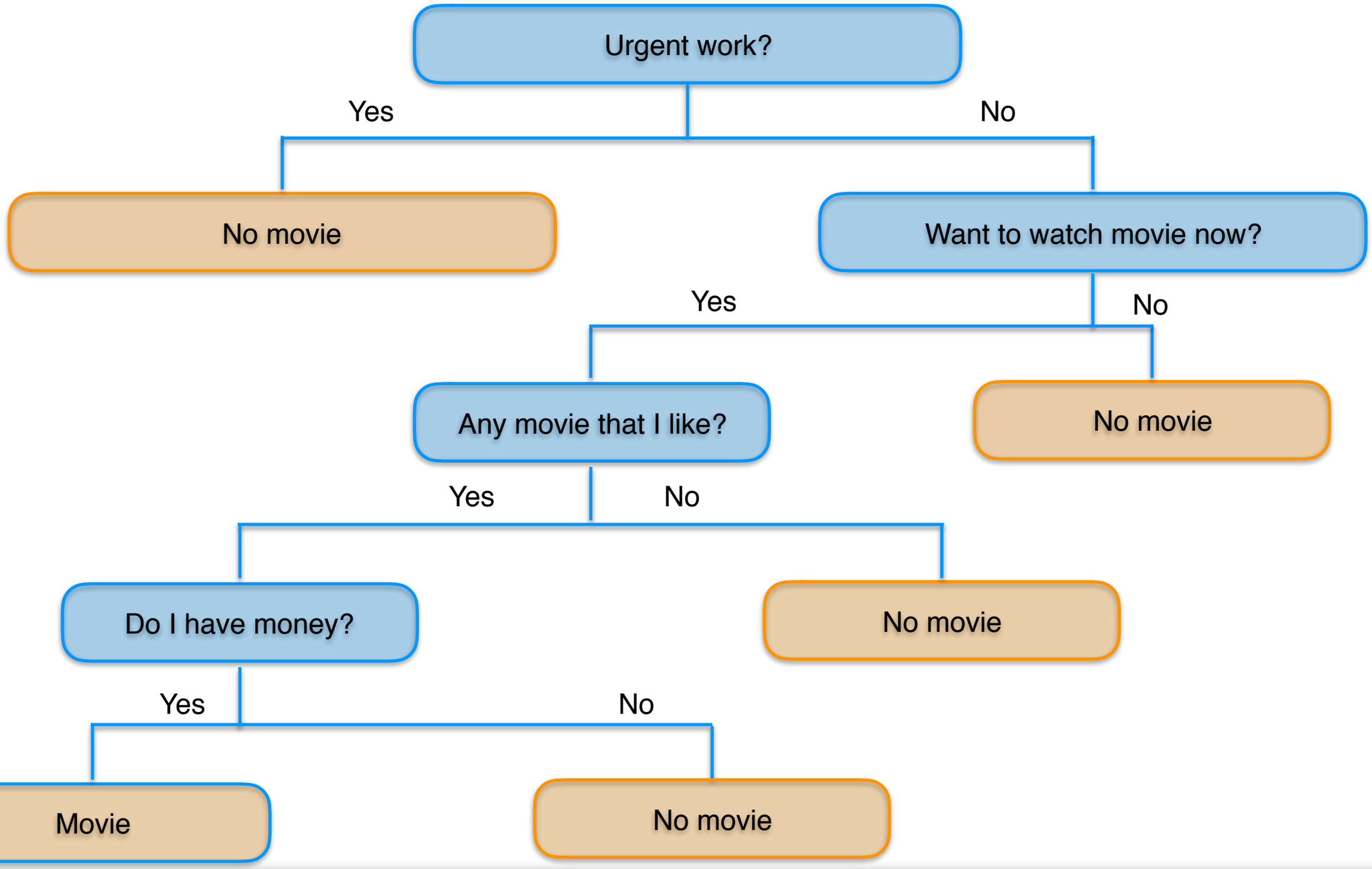
Is there any movie in the theatre that I like?

Do I have money to pay for the tickets?

Do I have any urgent work?



How Can We Arrive At A Decision? Let's Take A Structured Look



How Do We Calculate?

$$10 / 2 = ?$$

Q: What operation do I need to perform?
A: Division

Q: What is the dividend?
A: 10

Q: What is the divisor?
A: 2

Perform: Call divide function from our memory with parameters dividend = 10 and divisor = 2
A: The answer to the question is 5

How Do We Calculate?

Wow!

Now that we could solve this problem; did you realize there's a bit of training/ memorising involved in the process of solving this basic calculation?

We could invoke the right operation because we could classify the operation into one such class.

Basic operations are explicitly classified

Addition

Call *add* function from memory

Subtraction

Call *subtract* function from memory

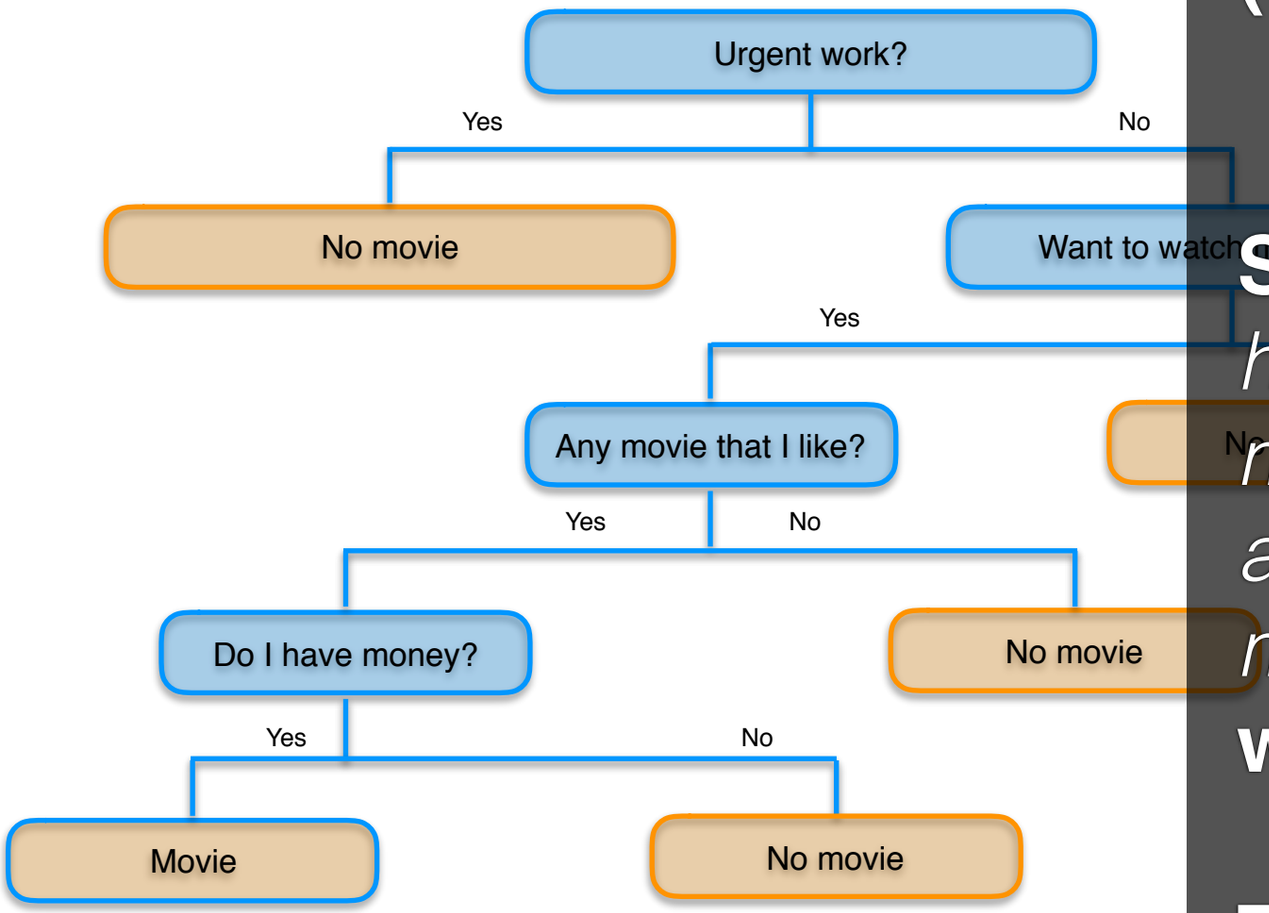
Multiplication

Call *multiply* function from memory

Division

Call *divide* function from memory

What Is This?



A tree; Well a decision tree. Because it helps us taking decision and it looks like a tree (up-side down).

So this tells me when *I do not have other urgent work, I am in the mood of watching a movie, There's a movie of my choice and I do have money to pay for the movie ticket* **- I will go for a movie.**

This tells me I can classify my response into *'Movie'* class and *'No movie'* class following the branches of the tree.

Is it so simple always?

Do I want to watch a movie now?

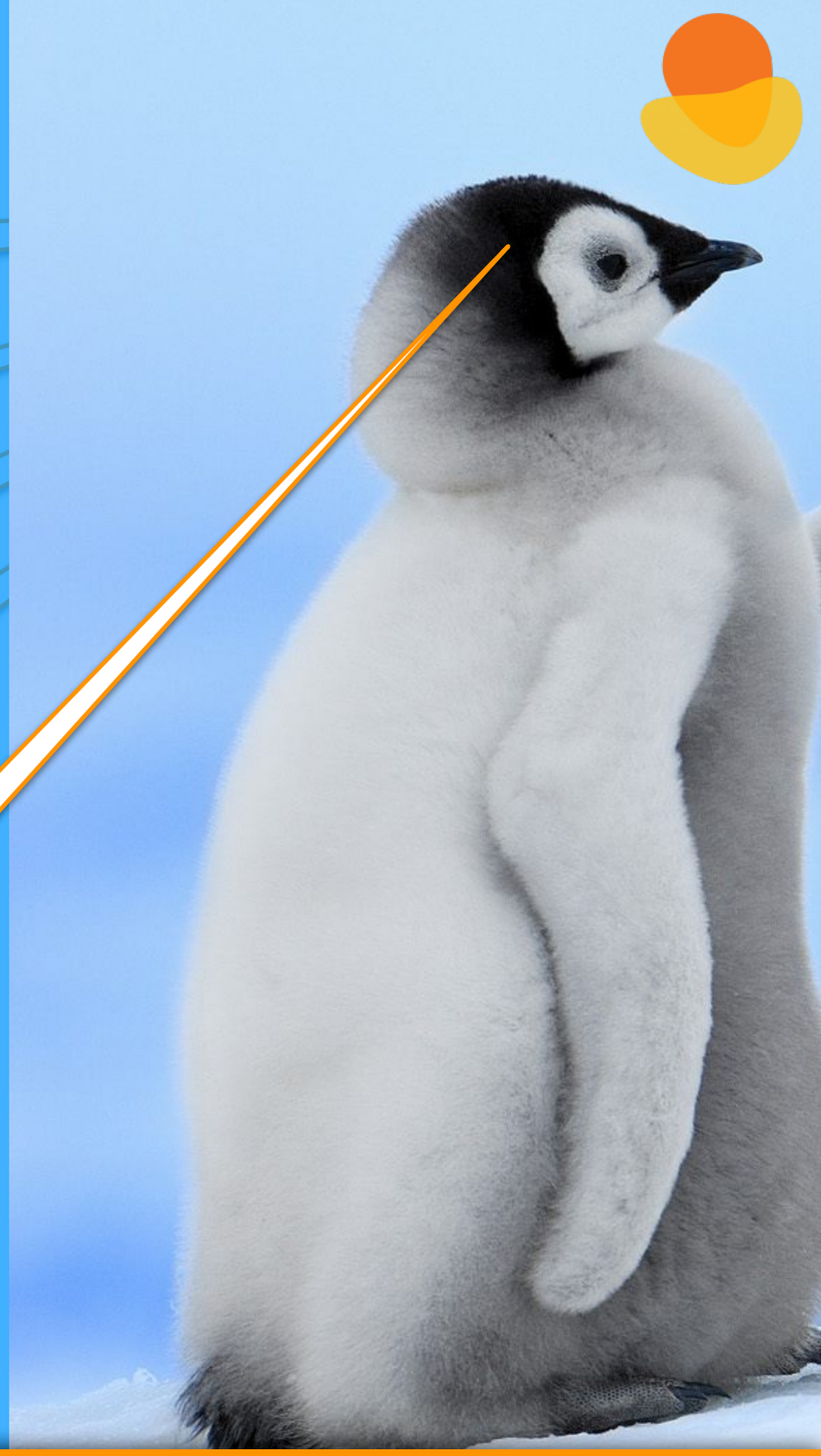
Is there any movie in the theatre that I like?

Do I have money to pay for the tickets?

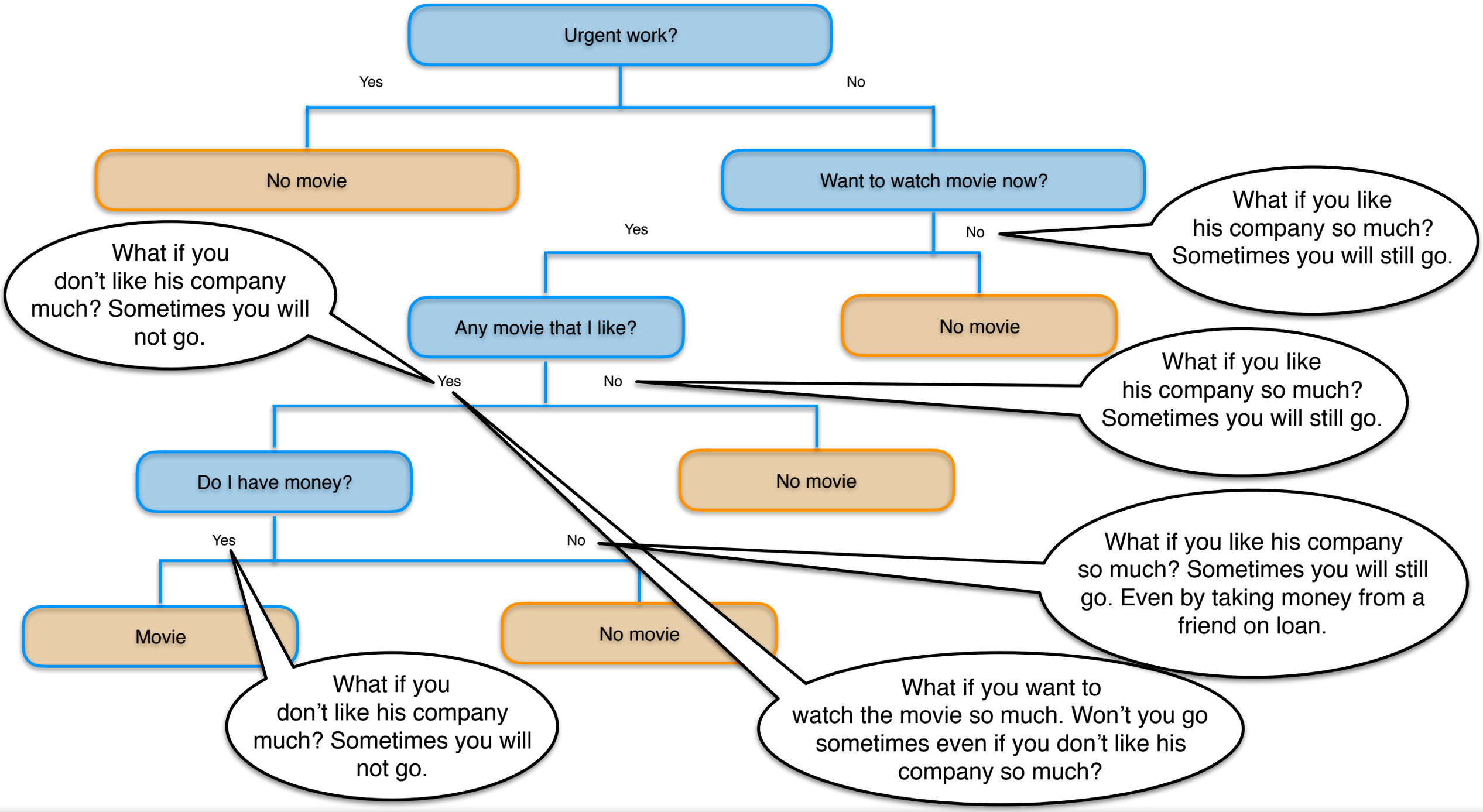
Do I have any urgent work?

Just add one more question

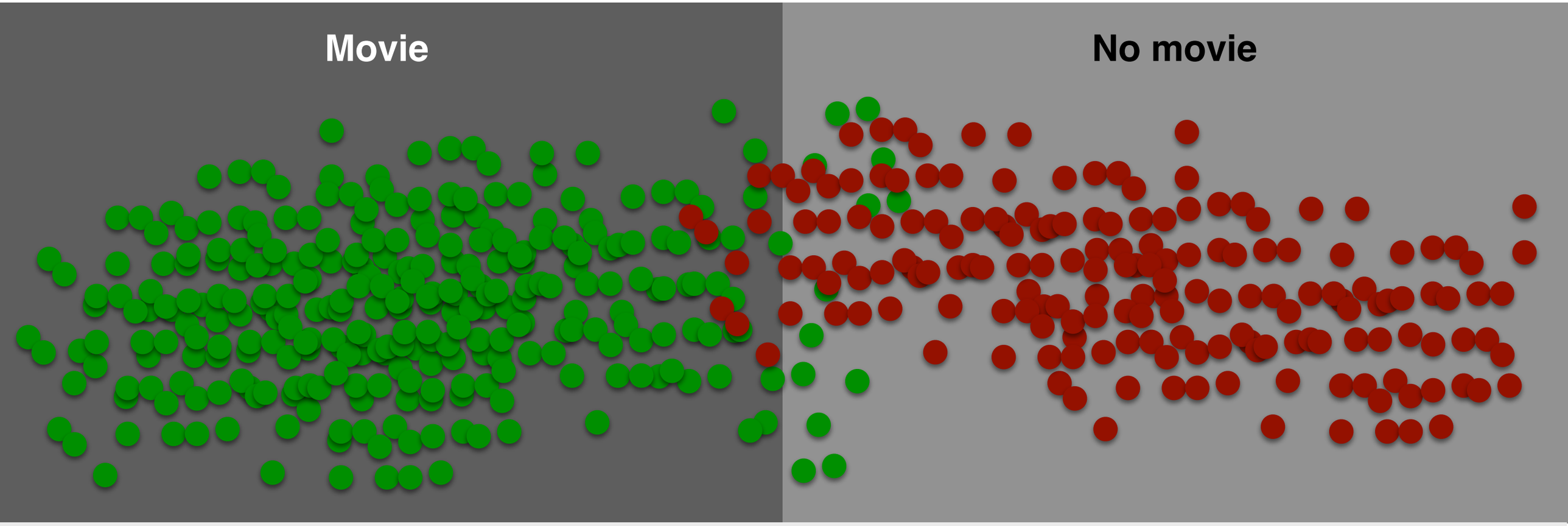
Do I like his company?



Question ‘Do I Like His Company’ Added To The Tree



Misclassification

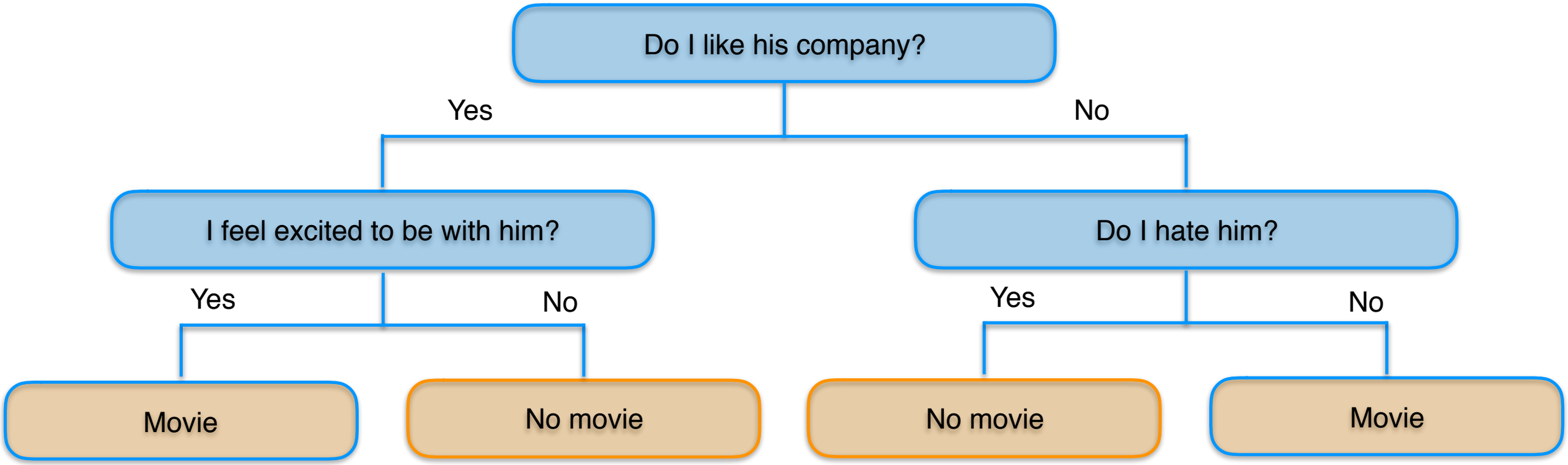


We wanted to classify our decisions where green balls would indicate our decision in favor of the movie and red balls would indicate our decisions in disapproval to the movie.
But post introduction of the question 'Do I like his company' made things difficult for us. We went to the movie (when our decision should have been 'no movie' based on the tree) just because we liked our friends company and we did not go to the movie (when our decision should have been 'movie' based on the tree) because we did not like the person's company so much.

This situation is called *misclassification*. Where few observations have fallen in the wrong region making our classification rule prone to error.

How Can I Reduce Error?

Let us branch out the question: Do I like his company?



Now if we add this part into our decision tree we will surely reduce misclassification. But can we get rid of all misclassifications at this stage?

NO

How Can I Reduce More Error?

By branching out the decision tree more and more.

You mean more we branch out more reduction we get in misclassifications?

Yes

But where to stop? How much would we let our tree grow?

Where To Stop Our Tree From Growing?

Branching out

Less misclassification
Higher cost

Stop growing

Easy to manage
Prone to error

In real-life applications we need to balance.

Lets’s Summarize

We want to classify our observations into classes.

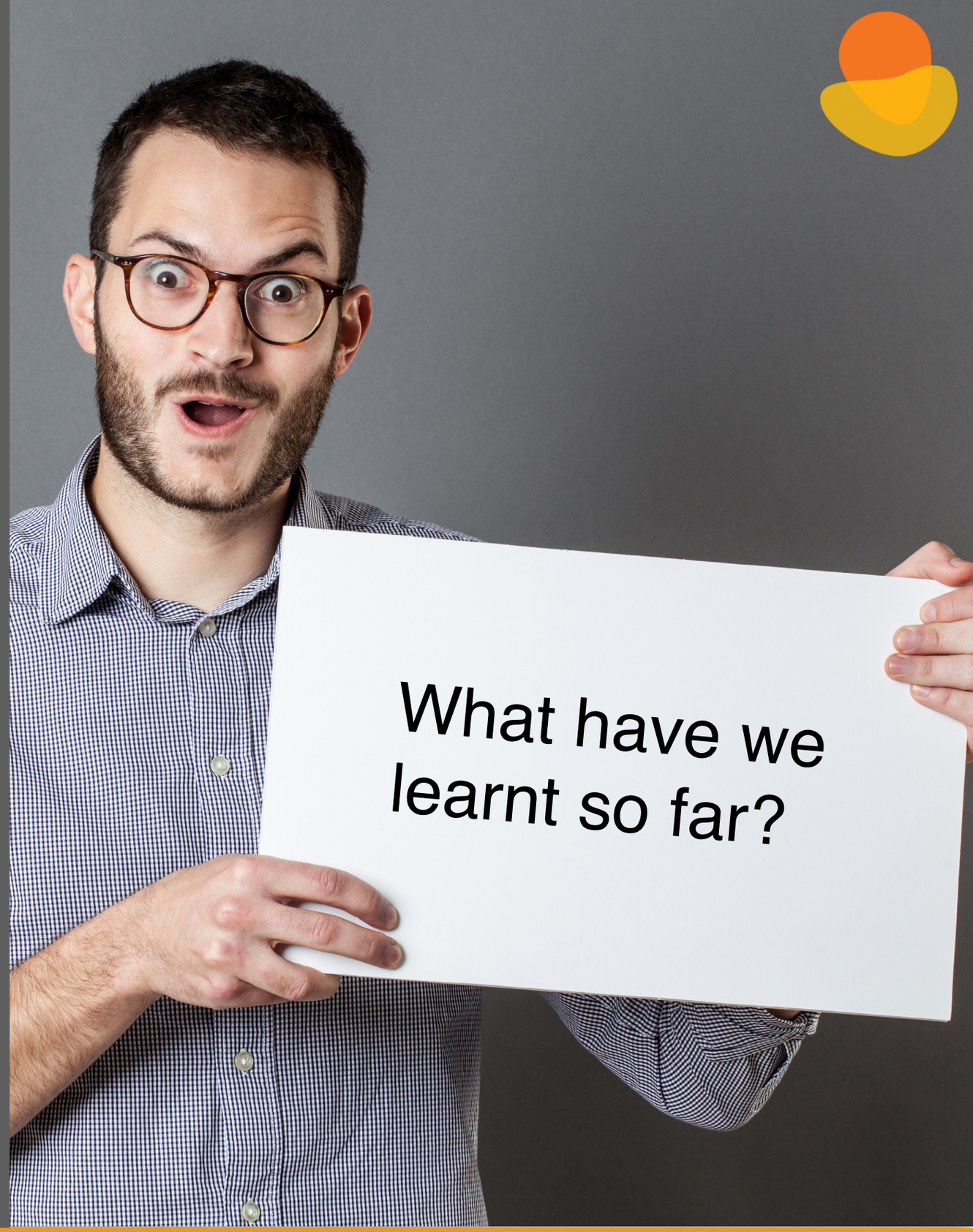
We will use training data to devise a classification rule.

The rule will have errors. There will be misclassifications.

We want to reduce the error from the rule.

We have to stop the tree from growing at a point.

We want to keep the cost minimum.



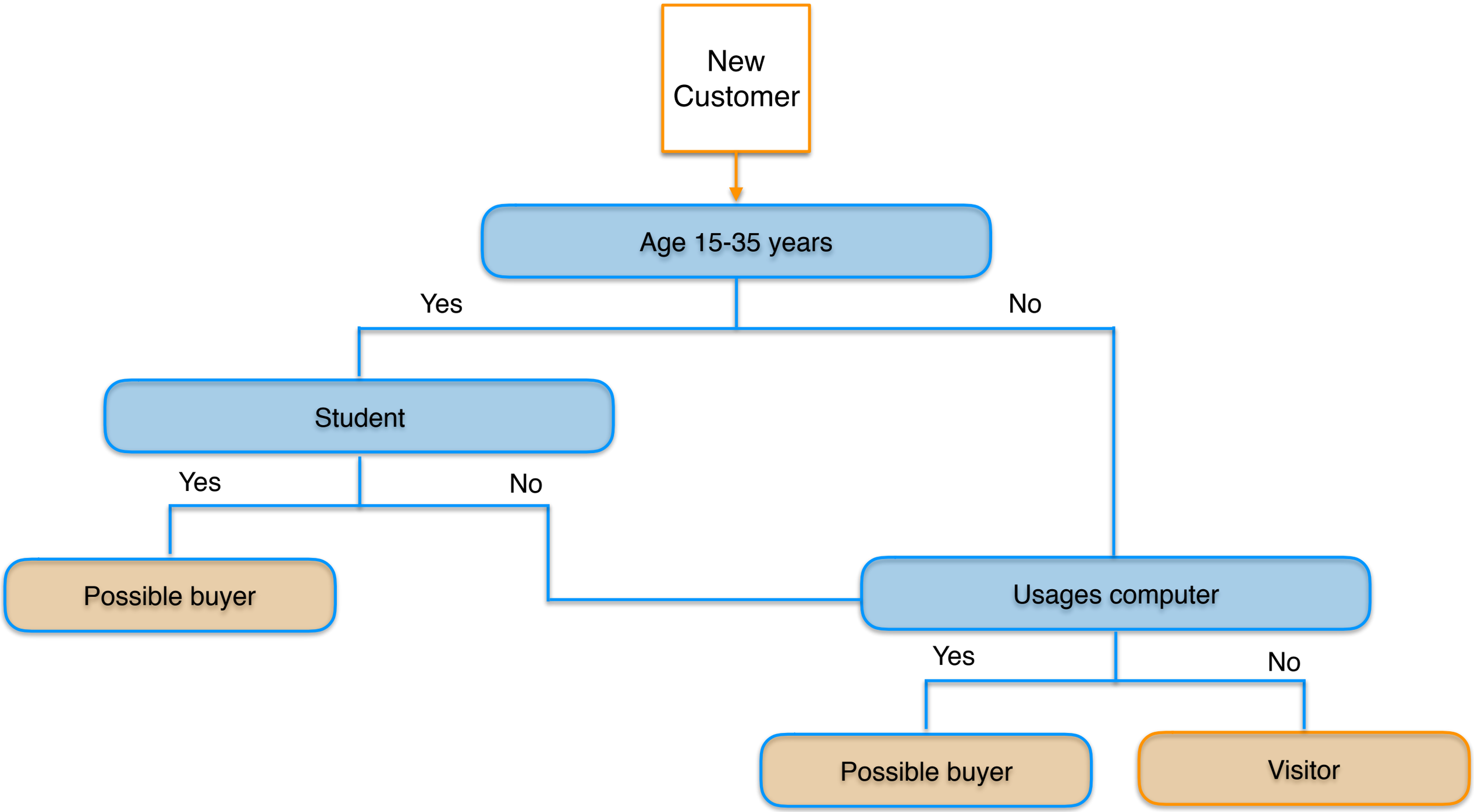
A Shop-owner's Problem

Let us now talk about a problem of classifying a new customer entering a computer shop as a 'possible buyer' or 'visitor'. Here the classification done by the shop owner is based only on the visitor's age and usage of internet.

Let us say the preset rule for the shop owner is if the 'Age' of the visitor is between 15 - 35 years and he/she is a student then the visitor may be classified as 'possible buyer'.

Another rule he made is, if the visitor uses internet then he/she would be a 'possible buyer'. If both the questions have the answer 'No' then the owner classifies the visitor as a 'visitor' and do not pay attention to him/her. This is a simple example and there can be many like this with several questions.
Let us now create our decision tree to better understand this.

The Decision Tree - Shop-owner's Problem



Get Ready For Theory.

We have followed standards from Harvard University lecture in the following slides.

Building A Regression Tree

Divide the predictor space into J distinct not overlapping regions $R_1, R_2, R_3, \ldots, R_J$

We make the same prediction for all observations in the same region. We use the mean of responses for all training observations that are in the region.

Minimizing Error

Our aim is to find $R_1, R_2, R_3, \ldots, R_J$ so that RSS, residual sum of squares, is minimized.

So RSS = $\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$

where $\hat{y}_{R_j}$ is the mean response value of all training observations in the R_j region

This is computationally very expensive.

What is the solution?

Recursive Binary Splitting Algorithm

A. Consider all predictor X_p and all possible values of the cut-points s for each predictor. Choose the predictor and cut-point so that it minimizes the RSS

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

This will be quicker based on the assumption that the number of predictors is very large.

B. Apply **Step A** in the sub regions.

C. Stop when the node contains only one class or node contains less than ***n*** data points or maximum depth is reached.

Pruning

We can stop branching out the tree after a certain RSS improvement is achieved.

Another approach is to let the tree grow large and then look at the sub-trees that minimize test error. This can be done by cross validation of all possible subtrees. But this process is surely too expensive.

Pruning Algorithm

A. Use recursive binary splitting to grow a large tree on the training data. Stop only when each terminal node has fewer than some minimum number of observations
B. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α, where is the tuning parameter
C. Use K-fold cross-validation to choose α
    a. Repeat A and B on the k^{th} fold
    b. Estimate the MSE as a function of α average all and pick α
D. Return the subtree from Step B that corresponds to the chosen value of α

Classification Tree

Classification tree is very similar to regression tree except that fact that it is used to predict a qualitative response rather than a quantitative one.

So here RSS cannot be used as a criterion for making the binary splits.

Classification Error Rate And Gini Index

$$E = 1 - \max_k \hat{p}_{mk}$$

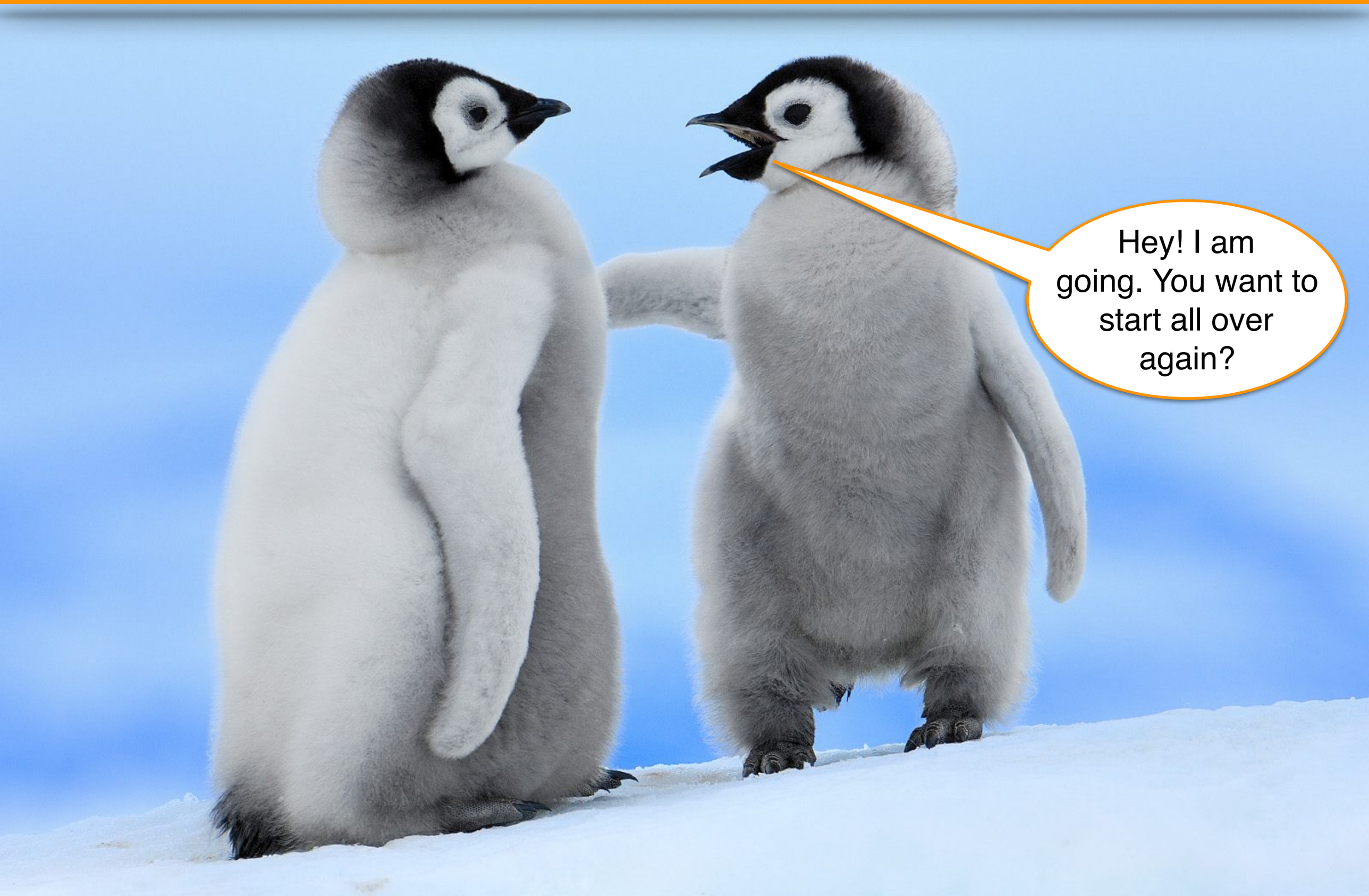
$\hat{p}_{mk}$ represents the proportion of training observations in the m^{th} region that are from the k^{th} class.

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

G takes small values when p_{mk} is small or close to 1, therefore is a measure of purity of the nodes.

Pruning Classification Tree

Use the same algorithm as for regression tree but instead of RSS use Gini index.


Hey! I am going. You want to start all over again?