Solution 1: Splitting criteria

a) • **Pen-and-paper solution**

1) Here, we have only one split variable x. We probe all splits of x in two groups, where thresholds are placed equidistant between the observed feature values (think about why this might help generalization):
   - (1) — $(2, 7, 10, 20)$ (split point 1.5)
   - $(1, 2)$ — $(7, 10, 20)$ (split point 4.5)
   - $(1, 2, 7)$ — $(10, 20)$ (split point 8.5)
   - $(1, 2, 7, 10)$ — (20) (split point 15)
2) For each split point, compute the sum of squares ($L2$ loss) in both groups.
3) Choose the point that splits both groups best w.r.t. empirical risk reduction.

A split point t leads to the following half-spaces:

$$\mathcal{N}_1(t) = \{(x, y) \in \mathcal{N} : x \leq t\} \quad \text{and} \quad \mathcal{N}_2(t) = \{(x, y) \in \mathcal{N} : x > t\}.$$

Calculate the risk $\mathcal{R}(\mathcal{N}, t)$ for each split point:

- $x \leq 1.5$

$$c_1 = y_1 = 1, c_2 = \tfrac{1}{4} \sum_{i=2}^{5} y_i = 5.625$$

$$\begin{aligned} \mathcal{R}(\mathcal{N}, 1.5) &= \tfrac{|\mathcal{N}_1|}{|\mathcal{N}|} \rho_{\text{MSE}}(\mathcal{N}_1) + \tfrac{|\mathcal{N}_2|}{|\mathcal{N}|} \rho_{\text{MSE}}(\mathcal{N}_2) = \\ &= \tfrac{1}{5} \cdot \left(\tfrac{1}{1}(1-1)^2\right) + \tfrac{4}{5} \cdot \left(\tfrac{1}{4}((1-5.625)^2 + (0.5-5.625)^2 + (10-5.625)^2 + (11-5.625)^2)\right) \\ &= 19.14 \end{aligned}$$

- $x \leq 4.5 \implies \mathcal{R}(\mathcal{N}, 4.5) = 13.43$
- $x \leq 8.5 \implies \mathcal{R}(\mathcal{N}, 8.5) = 0.13$ **optimal**
- $x \leq 15 \implies \mathcal{R}(\mathcal{N}, 15) = 12.64$

Proceeding accordingly for the monotonic, rank-preserving log transformation yields the same result:

- $\log x \leq 0.3 \implies \mathcal{R}(\mathcal{N}, 0.3) = 19.14$
- $\log x \leq 1.3 \implies \mathcal{R}(\mathcal{N}, 1.3) = 13.43$
- $\log x \leq 2.1 \implies \mathcal{R}(\mathcal{N}, 2.1) = 0.13$ **optimal**
- $\log x \leq 2.6 \implies \mathcal{R}(\mathcal{N}, 2.6) = 12.64$

• **R solution**

```r
x <- c(1, 2, 7, 10, 20)
y <- c(1, 1, 0.5, 10, 11)

compute_mse <- function (y) mean((y - mean(y))**2)
```

```r
compute_total_mse <- function (yleft, yright) {
  num_left <- length(yleft)
  num_right <- length(yright)
  w_mse_left <- num_left / (num_left + num_right) * compute_mse(yleft)
  w_mse_right <- num_right / (num_left + num_right) * compute_mse(yright)
  w_mse_left + w_mse_right
}

split <- function(x, y) {

  # try out all unique points as potential split points and ...
  unique_sorted_x <- sort(unique(x))
  split_points <- head(unique_sorted_x, length(unique_sorted_x) - 1) +
    0.5 * diff(unique_sorted_x)

  node_mses <- lapply(
    split_points,
    function(i) {
      y_left <- y[x <= i]
      y_right <- y[x > i]
      # ... compute SS in both groups
      mse_split <- compute_total_mse(y_left, y_right)
      print(sprintf("split at %.6f: empirical risk = %.2f", i, mse_split))
      mse_split
    }
  )

  # select the split point yielding the maximum impurity reduction
  split_points[which.min(node_mses)]

}

split(x, y) # 3rd obs is best split point
```

```
## [1] "split at 1.500000: empirical risk = 19.14"
## [1] "split at 4.500000: empirical risk = 13.43"
## [1] "split at 8.500000: empirical risk = 0.13"
## [1] "split at 15.000000: empirical risk = 12.64"
## [1] 8.5
```

```r
split(log(x), y) # again, 3rd obs wins
```

```
## [1] "split at 0.346574: empirical risk = 19.14"
## [1] "split at 1.319529: empirical risk = 13.43"
## [1] "split at 2.124248: empirical risk = 0.13"
## [1] "split at 2.649159: empirical risk = 12.64"
## [1] 2.124248
```

b) For regression trees, we usually identify *impurity* with *variance*. Here is why:

- It is reasonable to define impurity via the deviation between actual target values and the predicted constant – either using absolute or square distances to enforce symmetry of positive and negative residuals.
- Recall the constant $L2$ risk minimizer for a node $\mathcal{N}$:

$$\bar{y} = \arg\min_c \frac{1}{|\mathcal{N}|} \sum_{i=1}^{|\mathcal{N}|} (y^{(i)} - c)^2,$$

because

$$\begin{aligned}
\min_c \frac{1}{|\mathcal{N}|} \sum_{i=1}^{|\mathcal{N}|} (y^{(i)} - c)^2 &\iff \frac{\partial}{\partial c} \left(\frac{1}{|\mathcal{N}|} \sum_{i=1}^{|\mathcal{N}|} (y^{(i)} - c)^2 \right) = 0 \\
&\iff \frac{1}{|\mathcal{N}|} \frac{\partial}{\partial c} \left(\sum_{i=1}^{|\mathcal{N}|} \left({y^{(i)}}^2 - 2y^{(i)}c + c^2 \right) \right) = 0 \\
&\iff \left(\sum_{i=1}^{|\mathcal{N}|} \left(-2y^{(i)} + 2c \right) \right) = 0 \\
&\iff |\mathcal{N}| \cdot c = \sum_{i=1}^{|\mathcal{N}|} y^{(i)} \\
&\implies \hat{c} = \frac{1}{|\mathcal{N}|} \sum_{i=1}^{|\mathcal{N}|} y^{(i)} = \bar{y}.
\end{aligned}$$

- Consequently, we have

$$\bar{y} = \arg\min_c \frac{1}{|\mathcal{N}|} \sum_{i=1}^{|\mathcal{N}|} (y^{(i)} - c)^2,$$

where the right hand side is the (biased) sample variance for sample mean c.

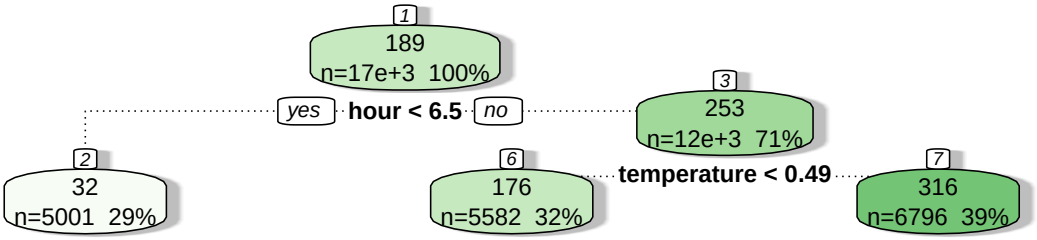
- Therefore, predicting the sample mean both minimizes risk under $L2$ loss and variance impurity.
- Since constant mean prediction is equivalent to an intercept LM (minimizing the sum of squared residuals!), regression trees with $L2$ loss perform piecewise constant linear regression.
- The same correspondence holds between impurity via absolute distances and $L1$ regression.

Solution 2: CART hyperparameters

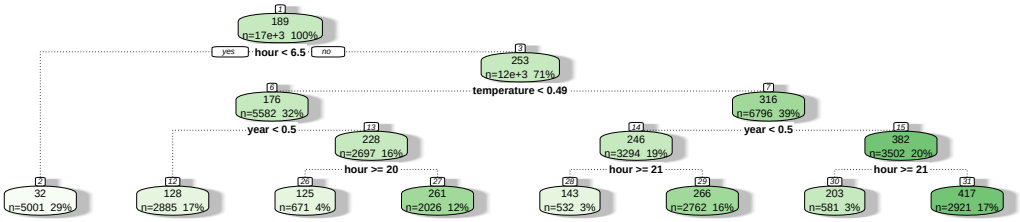
a) Allowing for more splits will make the model more complex, thus – all else being equal – achieving a better data fit but also increasing the risk of overfitting.

b) Code:

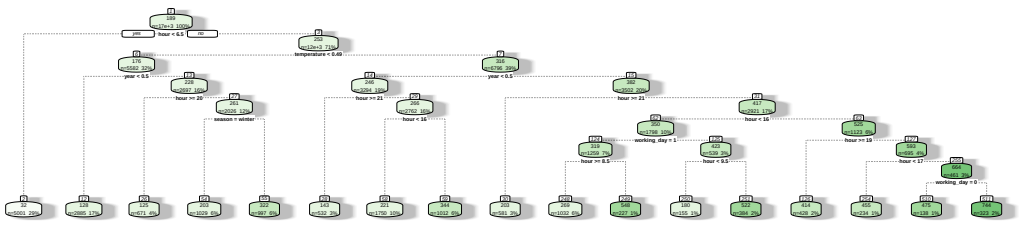
```
library(mlr3verse)
library(rattle)
task <- tsk("bike_sharing")
task$select(task$feature_names[task$feature_names != "date"])

for (i in c(2, 4, 8)) {
  learner <- lrn("regr.rpart", minsplit = 2, maxdepth = i, minbucket = 1)
  learner$train(task)
  fancyRpartPlot(learner$model, caption = sprintf("maxdepth: %i", i))
}
```



maxdepth: 2



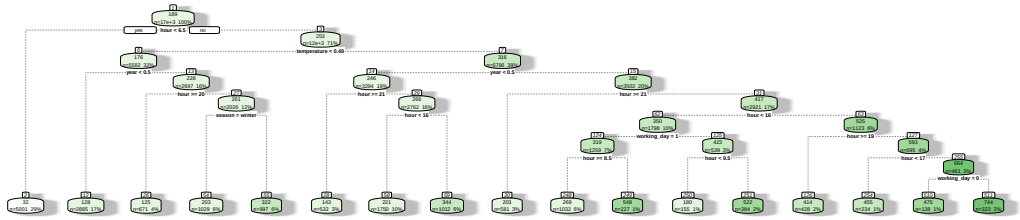
maxdepth: 4



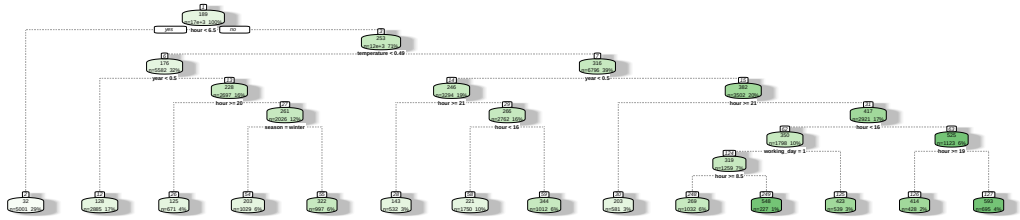
maxdepth: 8

```
for (i in c(5, 1000, 10000)) {
  learner <- lrn("regr.rpart", minsplit = i, maxdepth = 20, minbucket = 1)
  learner$train(task)
```

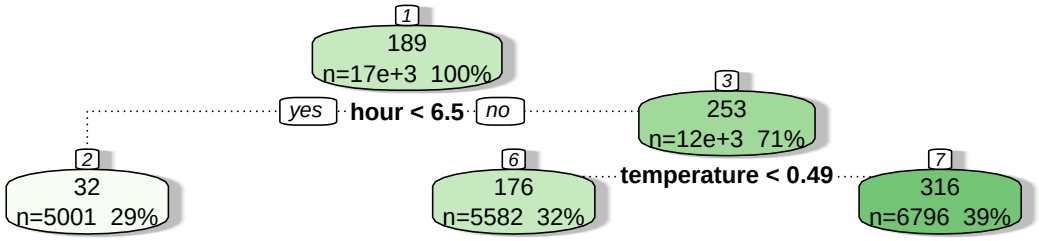
```
  fancyRpartPlot(learner$model, caption = sprintf("minsplit: %i", i))
}
```



minsplit: 5



minsplit: 1000



minsplit: 10000

Higher values of `maxdepth` and lower values of `minsplit`, respectively, produce more complex trees.

c) Both hyperparameters can be used to control tree depth, and their effect depends on the properties of the data-generating process (e.g., at least 100 observations to split further can mean very pure or very impure nodes). Sometimes requirements like interpretability might steer our decision (e.g., a tree of depth 15 is probably hard to interpret). Usually, however, we will employ *hyperparameter tuning* to determine the values for both (and other) hyperparameters, deferring the responsibility from the practitioner to the data.

Solution 3: Impurity reduction

a) The deterministic rule tells us to pick the class with the highest empirical frequency. With the randomizing rule, we draw from a categorical distribution that is parameterized with these same empirical frequencies, meaning we draw the most frequent class with probability $\gamma = \max_k \pi_k^{(\mathcal{N})}$. If we repeat this process many times, we will predict this class γ % of the time. Therefore, in expectation, we will also predict the most frequent class in most cases, but the rule is more nuanced as the magnitude of γ makes a difference (the closer to 1, the more similar both rules).

b) In order to compute the expected MCE, we need some random variables (RV) because we want to make a statement that holds for arbitrary training data drawn from the data-generating process. More precisely, we define $n \in \mathbb{N}$ i.i.d. RV $Y^{(1)}, \ldots, Y^{(n)}$ that are distributed according to the categorical distribution induced by the observed class frequencies:

$$\mathbb{P}(Y^{(i)} = k|\mathcal{N}) = \pi_k^{(\mathcal{N})} \quad \forall i \in \{1, \ldots, n\}, \quad k \in \mathcal{Y}.$$

The label $y^{(i)}$ of the i-th training observation is thus a realization of the corresponding RV $Y^{(i)}$.

Since our new randomization rule is stochastic, the predictions for the training observations will also be realizations of RV that we denote by $\hat{Y}^{(1)}, \ldots, \hat{Y}^{(n)}$. By design, they follow the same categorical distribution:

$$\mathbb{P}(\hat{Y}^{(i)} = k|\mathcal{N}) = \pi_k^{(\mathcal{N})} \quad \forall i \in \{1, \ldots, n\}, \quad k \in \mathcal{Y}.$$

Then, we can define the MCE for a node with $n = |\mathcal{N}|$ when the randomizing rule is used:

$$\rho_{\text{MCE}}(\mathcal{N}) = \frac{1}{n} \sum_{i=1}^{n} \left[Y^{(i)} \neq \hat{Y}^{(i)} \right].$$

Taking the expectation of this MCE leads to a statement about a node with arbitrary training data:

$$
\begin{aligned}
\mathbb{E}_{Y^{(1)},\ldots,Y^{(n)},\hat{Y}^{(1)},\ldots,\hat{Y}^{(n)}}\left(\rho_{\mathrm{MCE}}(\mathcal{N})\right) &= \mathbb{E}_{Y^{(1)},\ldots,Y^{(n)},\hat{Y}^{(1)},\ldots,\hat{Y}^{(n)}}\left(\frac{1}{n}\sum_{i=1}^{n}\left[Y^{(i)} \neq \hat{Y}^{(i)}\right]\right) \\
&= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{Y^{(i)},\hat{Y}^{(i)}}\left(\left[Y^{(i)} \neq \hat{Y}^{(i)}\right]\right) \quad \text{i.i.d. assumption + linearity} \\
&= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{Y^{(i)}}\left(\mathbb{E}_{\hat{Y}^{(i)}}\left(\left[Y^{(i)} \neq \hat{Y}^{(i)}\right]\right)\right) \quad \text{Fubini's theorem} \\
&= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{Y^{(i)}}\left(\sum_{k\in\mathcal{Y}}\pi_k^{(\mathcal{N})}\cdot\left[Y^{(i)} \neq k\right]\right) \\
&= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{Y^{(i)}}\left(1-\pi_{k=Y^{(i)}}^{(\mathcal{N})}\right) \\
&= \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{g}\pi_k^{(\mathcal{N})}\cdot\left(1-\pi_k^{(\mathcal{N})}\right) \\
&= n\cdot\frac{1}{n}\sum_{k=1}^{g}\pi_k^{(\mathcal{N})}\cdot\left(1-\pi_k^{(\mathcal{N})}\right) \\
&= \sum_{k=1}^{g}\pi_k^{(\mathcal{N})}\cdot\left(1-\pi_k^{(\mathcal{N})}\right).
\end{aligned}
$$

This is precisely the Gini index CART use for splitting with Brier score. Gini impurity can thus be viewed as the expected frequency with which the training samples will be misclassified in a given node $\mathcal{N}$.

In other words, in constructing CART with minimal Gini impurity, we minimize the expected rate of misclassification across the training data.