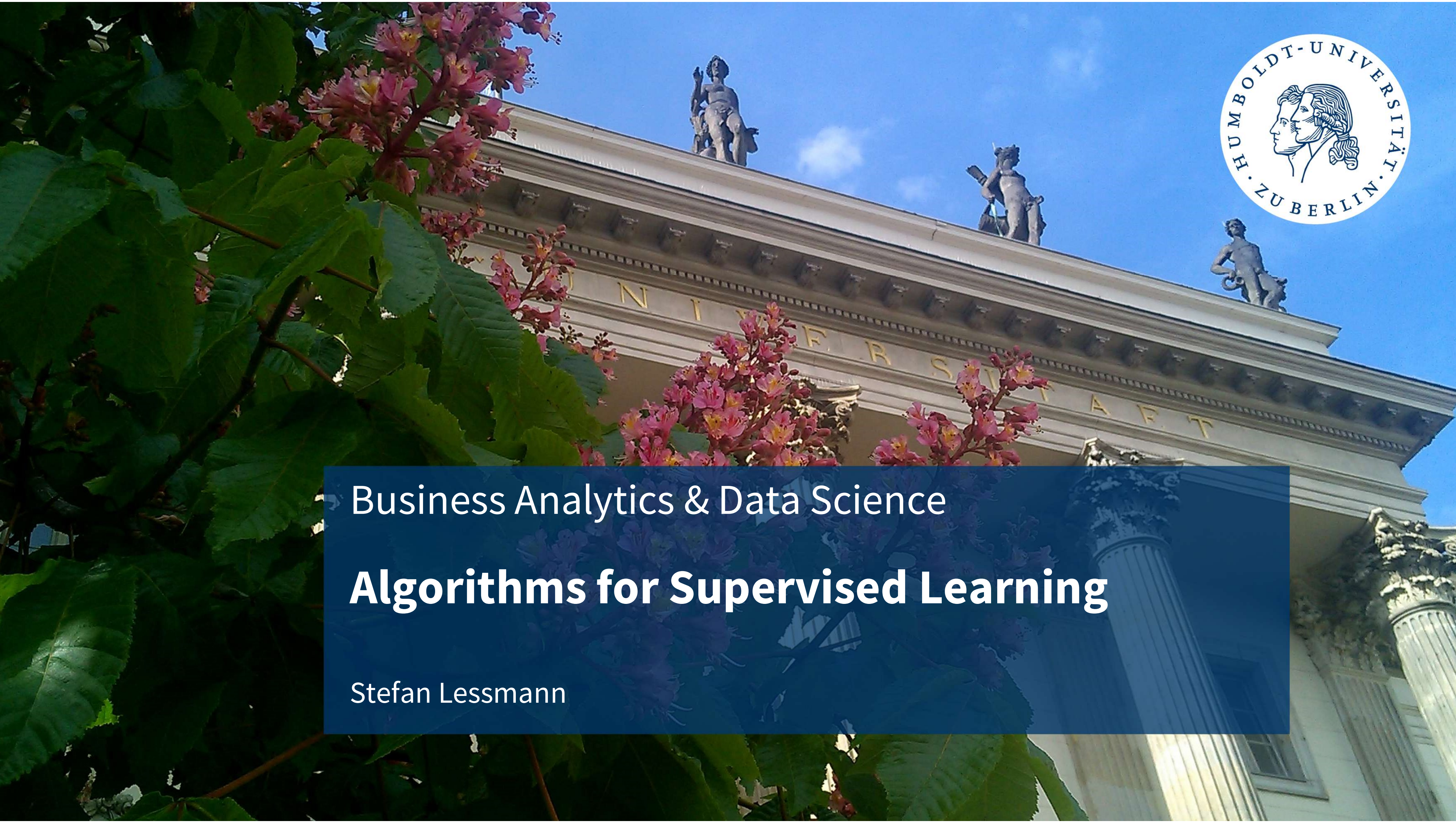


Business Analytics & Data Science

Algorithms for Supervised Learning

Stefan Lessmann

Agenda

Introduction
Supervised learning algorithms, commonalities and differences

Linear and Logistic Regression
Linear regression revisited, logistic link function, model estimation

Decision Tree Learning
Recap of regression, building blocks of a tree-growing algorithm

Summary

Introduction

Supervised learning algorithms, commonalities and differences

Two-Stage Paradigm

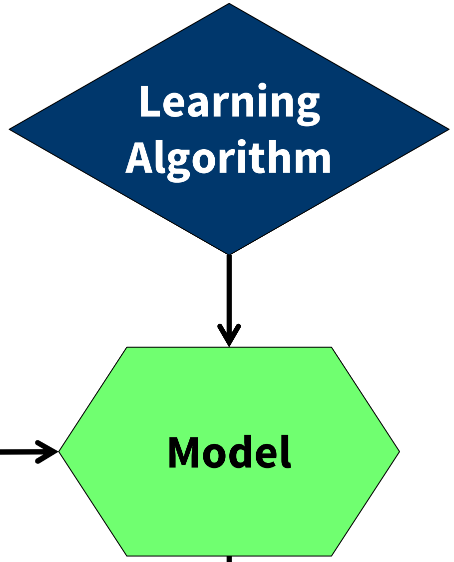
Characteristic of supervised (and other forms of) ML

Stage 1: Model Training

Data-driven development of a predictive model using labelled data $\mathcal{D} = \{Y_i, \boldsymbol{X}_i\}_{i=1}^{n}$

Training data incl. Y

i	Y	X_1	X_2	…	X_m
1	…	…	…	…	…
2	…	…	…	…	…
…	…	…	…	…	…
n	…	…	…	…	…



Stage 2: Model Testing & Use

Application of trained model to novel data yields output (e.g., forecasts)

New data w/o Y

i	X_1	X_2	…	X_m
$n+1$	…	…	…	…
$n+2$	…	…	…	…
…	…	…	…	…
N	…	…	…	…

Forecasts of Y

i	$\hat{Y}$
$n+1$	…
$n+2$	…
…	…
N	…

Supervised ML Algorithms Address the Same Problem Setting

Resale price forecasting example

- **We aim at forecasting resale prices (our target variable) denoted by Y**
- **We assume that resale prices Y depend on features X**
  - We do not know how exactly resale prices depend on feature values
  - But we have access to historical data $\mathcal{D} = \{y_i, \boldsymbol{x}_i\}_{i=1}^{n}$ that exemplifies the relationship
- **At decision time (e.g., when forecast is needed), we can observe X but not Y**
- **We use algorithms to learn a model f that maps from features to target $f(\boldsymbol{X}) \longrightarrow Y$**

Product	List price [$]	Age [month]	Industry	…
…	…	…	…	…

$$\boldsymbol{X} = (X_1, X_2, \dots, X_m) \in \mathbb{R}^m \xrightarrow{f(X)} Y \in \mathbb{R}$$

Observed resale price [$]
…

Algorithms for Supervised Learning (Selection)

A subjective view and a bit of guidance

Supervised learning algorithms

Tree- and prototype-based algorithms (non-parametric)

- CART
- CHAID
- C4.5

Individual trees

- Bagging / Random Forest
- Gradient Boosting / XGB

Ensembles (many trees)

- Nearest neighbors
- …

Regression-type algorithms (semi-/parametric)

- Linear regression
- Generalized linear models (GLM)
- Generalized additive models (GAM)
- Artificial neural networks (ANN)
- Support vector machines (SVM)
- …

Linear and Logistic Regression

Linear Regression Model

Postulates a linear, additive feature-to-target relationship

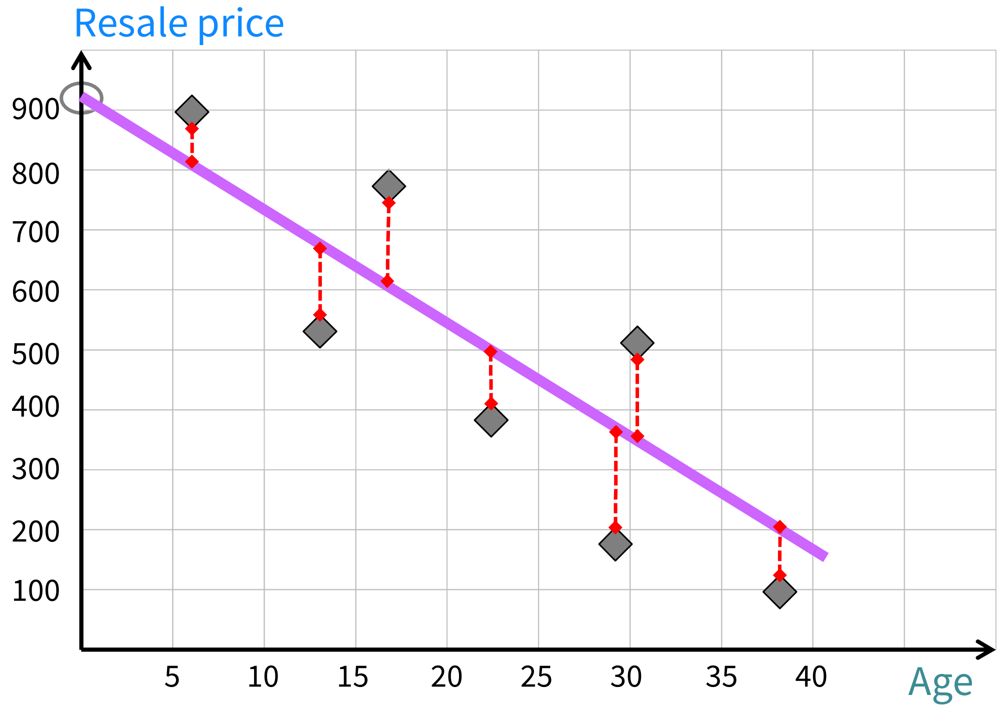
- **Famous regression equation**

$$\text{Resale Price} = \mathit{bias} + w_1 \text{List price} + w_2 \text{Age} + \cdots + w_m \text{Industry} + \mathit{residual}$$

- **Idea is to explain variation in resale prices by differences in observed feature values**
- **Simplification for plotting:**

$$\text{Resale Price} = \mathit{bias} + w_1 \text{Age} + \epsilon$$

…	Age [month]	…	Observed resale price [$]
…	6	…	900
…	13	…	515
…	17	…	890
…	23	…	395
…	29	…	180
…	31	…	501
…	38	…	100
…	…	…	…



Regression Model Estimation

Maximizing model fit through minimizing a loss function

- **A loss function J measures the degree to which the model output $\hat{Y}$ agrees with the true value of the target Y**
- **Squared-error loss**

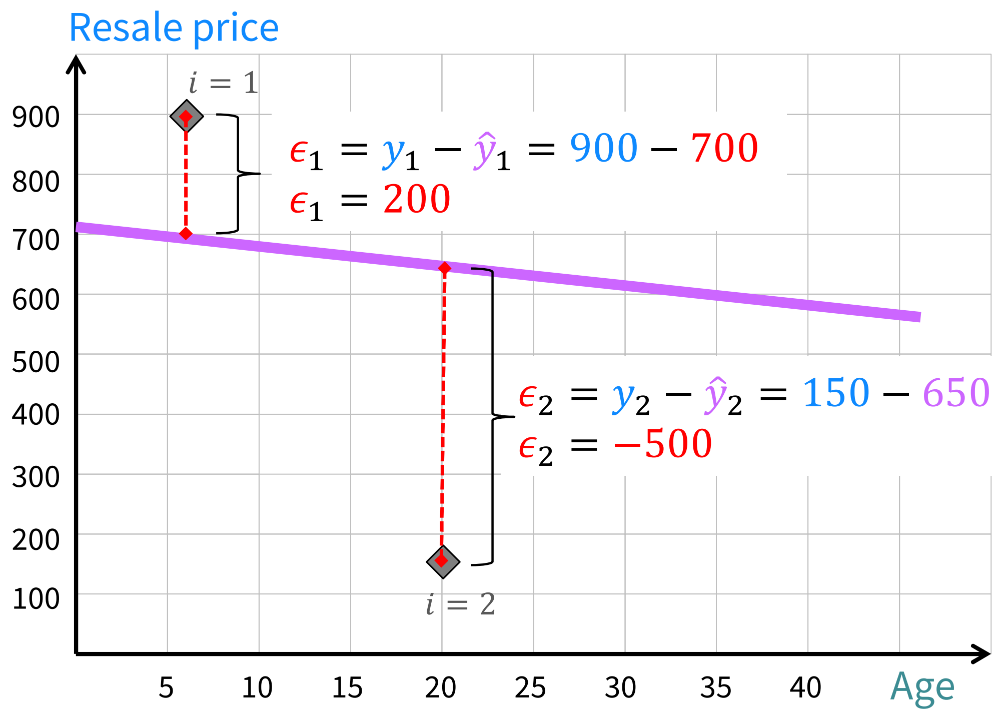
$$J = \sum_{i=1}^{n} (\epsilon_i)^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- **Whereby the model output depends on the free parameters w**

$$\hat{y}_i = b + w_1 x_{i1} + w_2 x_{i2} + \cdots + w_m x_{im}$$

$$\hat{y}_i = b + \sum_{j=1}^{m} w_j x_{ij}$$

1



Regression Model Estimation

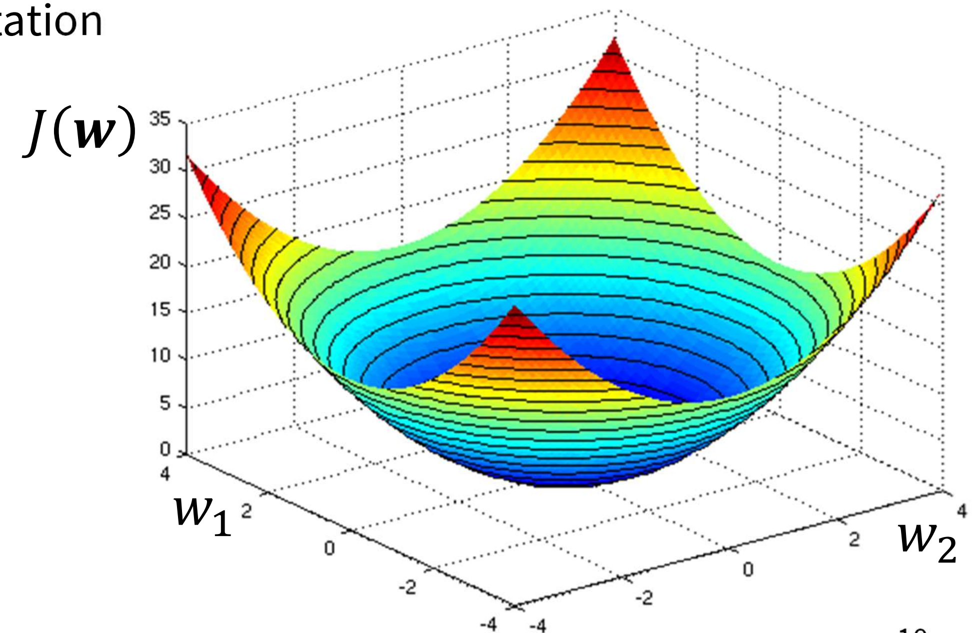
Minimizing empirical loss, that is the loss over our data

- **Squared-error loss**

$$J(\boldsymbol{w}) = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{m} w_j x_{ij}\right)^2 = \sum_{i=1}^{n}(y_i - \boldsymbol{w}\boldsymbol{X}_i)^2 = (\boldsymbol{y} - \boldsymbol{w}\boldsymbol{X})^\top(\boldsymbol{y} - \boldsymbol{w}\boldsymbol{X})$$

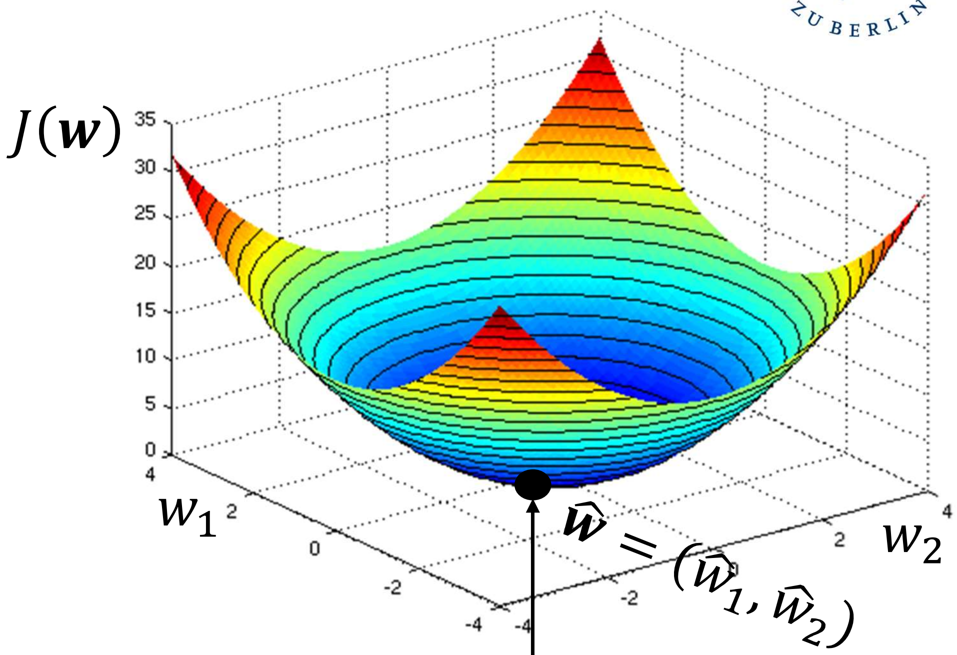
  - Note that we have dropped the bias to simplify the notation
  - Mathematically, we can add a constant to the data $\boldsymbol{X}$

- **To fit the regression model, we set w such that the error is minimal**
  - Mathematically, we seek the minimum of the loss function over the model parameters
  - $\hat{\boldsymbol{w}} \leftarrow \operatorname{argmin}_{\boldsymbol{w}} J(\boldsymbol{w})$



Regression Model Estimation
Finding the optimal solution

- **Formalization of estimation (training) task**
  - $J(\boldsymbol{w}) = \sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{m} w_j x_{ij}\right)^2$
  - $\hat{\boldsymbol{w}} \leftarrow \operatorname{argmin}_{\boldsymbol{w}} J(\boldsymbol{w})$
- **Solution**
  - Apply calculus principle
  - Calculate partial derivatives of $J(\boldsymbol{w})$ with respect to each w_j and set to zero



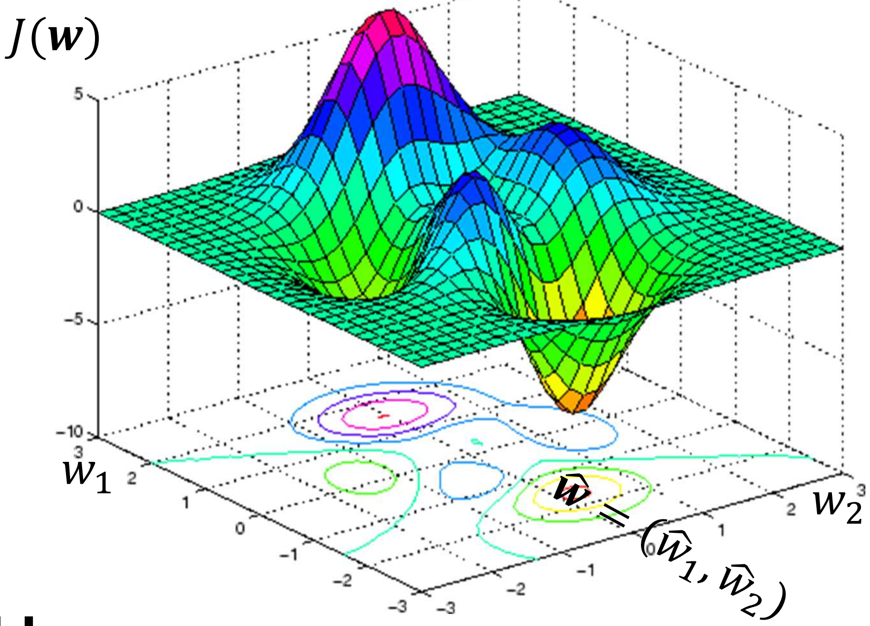
$$\frac{\partial J(\boldsymbol{w})}{\partial \boldsymbol{w}} = 0 \quad \left\{ \begin{array}{l} \dfrac{\partial J(\boldsymbol{w})}{\partial w_1} = 0 \\[2em] \dfrac{\partial J(\boldsymbol{w})}{\partial w_2} = 0 \end{array} \right. \Rightarrow \hat{\boldsymbol{w}} = \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{y}$$

For the special case of linear regression with squared error loss, we obtain an analytical solution.

Regression Model Estimation
Generalization

- **Minimization problem remains (largely) unchanged**
  - ☐ $J(\boldsymbol{w}) = \sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{m} w_j x_{ij}\right)^2$
  - ☐ $\widehat{\boldsymbol{w}} \leftarrow \operatorname{argmin}_{\boldsymbol{w}} J(\boldsymbol{w})$

$$\frac{\partial J(\boldsymbol{w})}{\partial \boldsymbol{w}} = 0 \quad \begin{cases} \dfrac{\partial J(\boldsymbol{w})}{\partial w_1} = 0 \\ \dfrac{\partial J(\boldsymbol{w})}{\partial w_2} = 0 \end{cases} \Rightarrow \widehat{\boldsymbol{w}} = ?$$



- **Computing analytical solution typically impossible**
- **Use iterative numerical algorithms to find $\widehat{w}$**

Classification Setting Revisited
Credit risk modeling example

- **Target variable is categorical**
  - Binary classification
  - Multi-class classification
- **Why not use linear regression?**

Bureau score	Collateral	Debt/Income	Years at address	Age	…	Default
650	Yes	20%	2	<21	…	No (0)
280	No	43%	0	21-29	…	Yes
750	Yes	27%	8	30-39	…	No (0)
600	Yes	18%	4	40-50	…	No (0)
575	No	33%	12	>50	…	No (0)
715	Yes	24%	1	21-29	…	No (0)
580	No	18%	6	40-50	…	Yes (1)
410	Yes	29%	4	21-29	…	No (0)
800	Yes	14%	10	40-50	…	Yes (1)

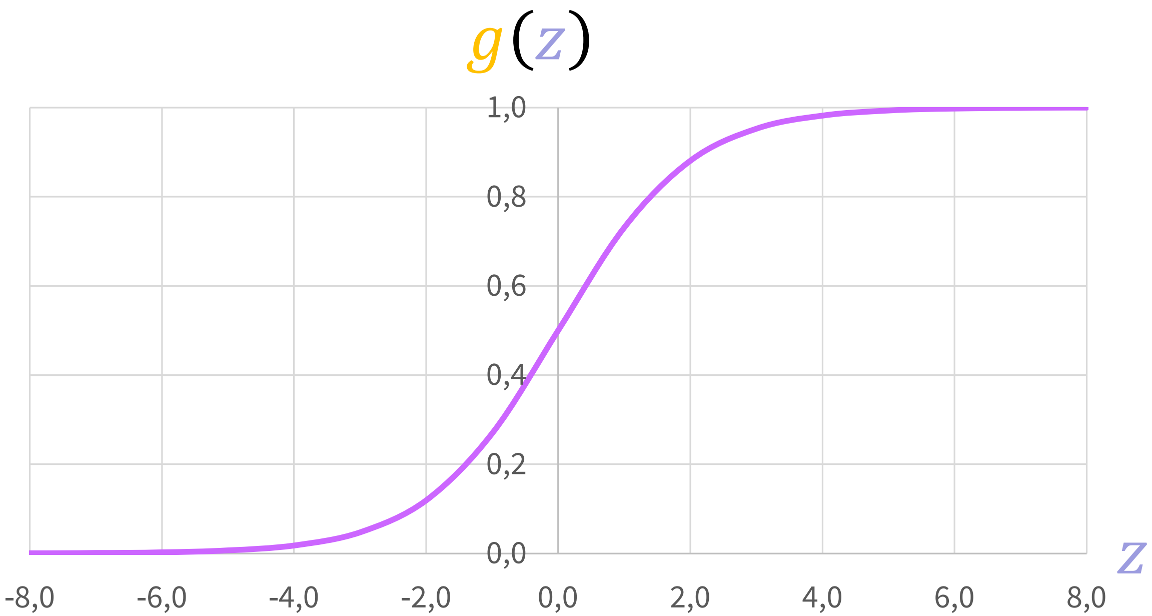
Logistic Regression Model

- **Focuses on binary classification while extensions address multi-class case**
- **Belongs to the family of generalized linear models (GLM)**
  - Outcome Y assumed be generated from a particular distribution (e.g., binomial for logistic regression)
  - Mean of that distribution assumed to depend on features X through $E(Y|X) = g^{-1}(\boldsymbol{w}\boldsymbol{X})$
  - With g denoting a link function
- **Logistic function**

$$g(z) = \frac{1}{1 + e^{-z}}$$

- **Linear predictor**

$$z = \boldsymbol{w}\boldsymbol{X}$$



Logistic Regression Model

Estimate the entry probability of each state

- **Credit Scoring example**
  - Classify credit applicants with feature values $\boldsymbol{X}$ into good ($Y = 1$) and bad ($Y = 0$) risks
  - Feature vector $\boldsymbol{X}$ captures data from the application form, bureau score, etc.
- **Model class membership probability using the logistic function**
  - $0 \leq p(Y = 1|\boldsymbol{X}), p(Y = 0|\boldsymbol{X}) \leq 1$
  - Solves problem with predictions outside the zero-one interval

$$p(Y = 1|\boldsymbol{X}) = \frac{1}{1 + e^{-z}}$$

$$p(Y = 0|\boldsymbol{X}) = 1 - p(Y = 1|\boldsymbol{X}) = 1 - \frac{1}{1 + e^{-z}}$$

with $z = \boldsymbol{wX}$

$$= b + w_1 x_1 + w_2 x_2 + \cdots + w_m x_m$$

Logistic Regression Model

Reformulation of the modeling task

$$\frac{p(Y=1|\boldsymbol{X})}{p(Y=0|\boldsymbol{X})} = e^{z} = e^{b+w_1x_1+w_2x_2+\cdots+w_mx_m}$$

$$\log\left(\frac{p(Y=1|\boldsymbol{X})}{p(Y=0|\boldsymbol{X})}\right) = b + w_1x_1 + w_2x_2 + \cdots + w_mx_m$$

$$\frac{p(Y=1|\boldsymbol{X})}{1-p(Y=1|\boldsymbol{X})}$$

is the odds in favor of $y = 1$, which implies good risk

$$\log\left(\frac{(Y=1|\boldsymbol{X})}{1-(Y=1|\boldsymbol{X})}\right)$$

is called the **logit** or log-odds

Can think of logistic regression as running linear regression using the log-odds as (transformed) target variable.

Logistic Regression Model

Variable importance

- **Interpretation of model coefficients**
  - Interpretability of a model
  - Major strength of (linear) regression
- **Say x_j increases by 1 unit:**

$$logit|_{x_j+1} = logit|_{x_j} + w_j$$

$$odds|_{x_j+1} = odds|_{x_j} + e^{w_j}$$

e^{w_j} is called the odds-ratio: the multiplicative increase in the odds when x_j increases by one unit (everything else remaining constant)

- $w_j > 0 \Leftrightarrow e^{w_j} > 1$ **odds and probability increase with** x_j
- $w_j < 0 \Leftrightarrow e^{w_j} < 1$ **odds and probability decrease with** x_j

Logistic Regression Model Estimation
Maximum Likelihood

- **Maximize the probability of observing the sample under the model**
  - Special case of empirical risk minimization
  - Assume a statistical model of the DGP and maximize free parameters of the likelihood function
- **Binary classification problem ($Y = 0$ or $Y = 1$)**
  - Assume the data is IID so for each observation, the class label originates from a Bernoulli trial
  - Probability density $Y = p^Y \cdot (1 - \mathrm{p})^{1-Y}$ with Y being our random variable and p the *success* probability
  - Rewritten for our classification problem we obtain $p(Y = 1|\boldsymbol{X})^Y \cdot (1 - p(Y = 1|\boldsymbol{X}))^{1-Y}$
- **Due to the IDD assumption, we can write the likelihood of observing a sample of size *n***

$$\prod_{i=1}^{n} p(Y_i = 1|\boldsymbol{X}_i)^{Y_i} \cdot (1 - p(Y_i = 1|\boldsymbol{X}_i))^{1-Y_i}$$

- **Log-Likelihood function**

$$\sum_{i=1}^{n} Y_i \log\big(p(Y_i = 1|\boldsymbol{X}_i)\big) + (1 - Y_i)\log(1 - p(Y_i = 1|\boldsymbol{X}_i))$$

  - Taking logs does not change the maximum
  - Mathematically more convenient to work with sums instead of products

Logistic Regression Model Estimation (cont.)

Minimize negative log-likelihood function

- **Find w through minimizing the negative log-likelihood function (aka log-loss)**

$$\hat{\boldsymbol{w}} \leftarrow \operatorname{argmin}_{\boldsymbol{w}} J(\boldsymbol{w}) = -\sum_{i=1}^{n} Y_i \log\big(p(Y_i = 1|\boldsymbol{X}_i)\big) + (1 - Y_i)\log(1 - p(Y_i = 1|\boldsymbol{X}_i))$$

- **Logistic regression models class probabilities using the logistic function**

$$p(Y = 1|\boldsymbol{X}) = \frac{1}{1 + e^{-z}}$$

$$z = \boldsymbol{w}\boldsymbol{X} = w_1 x_1 + w_2 x_2 + \cdots + w_m x_m$$

- **Putting everything together**

$$\hat{\boldsymbol{w}} \leftarrow \operatorname{argmin}_{\boldsymbol{w}} J(\boldsymbol{w}) = -\sum_{i=1}^{n} Y_i \log\left(\frac{1}{1 + e^{-\boldsymbol{w}\boldsymbol{X}_i}}\right) + (1 - Y_i)\log\left(1 - \frac{1}{1 + e^{-\boldsymbol{w}\boldsymbol{X}_i}}\right)$$

- **Minimize $J(w)$ using standard purpose solvers**

Regression Modeling Summarized

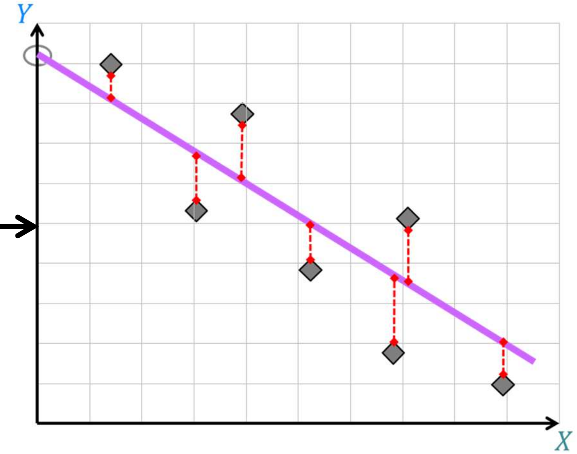
Stage 1: Model Training

Estimate regression coefficients through minimizing loss function.

- Least square loss function in linear regression
- Negative log-likelihood loss in logit model

Training data incl. Y

i	Y	X_1	X_2	…	X_m
1	…	…	…	…	…
2	…	…	…	…	…
…	…	…	…	…	…
n	…	…	…	…	…



$$\hat{Y} = b + \hat{w}_1 X_1 + \hat{w}_2 X_2 + \cdots + \hat{w}_m X_m$$

Stage 2: Model Testing & Use

Compute model output by regression equation.

- Feature values of the (new) observation
- Estimated coefficients (i.e., parameters of the linear predictor function)

New data w/o Y

i	X_1	X_2	…	X_m
$n+1$	…	…	…	…
$n+2$	…	…	…	…
…	…	…	…	…
N	…	…	…	…

Forecasts of Y

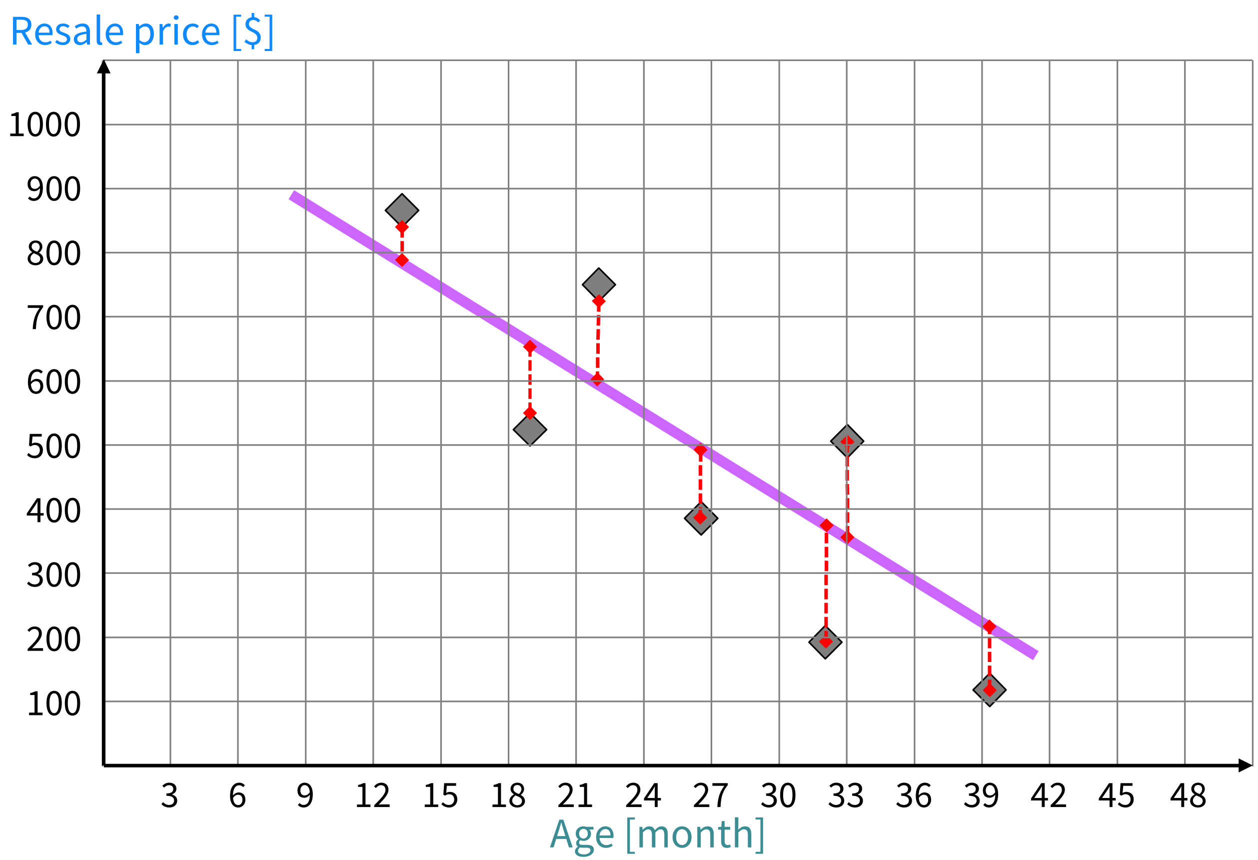
i	$\hat{Y}$
$n+1$	…
$n+2$	…
…	…
N	…

Decision Tree Learning

Recap of regression, building blocks of a tree-growing algorithm

Linear Regression Example Revisited

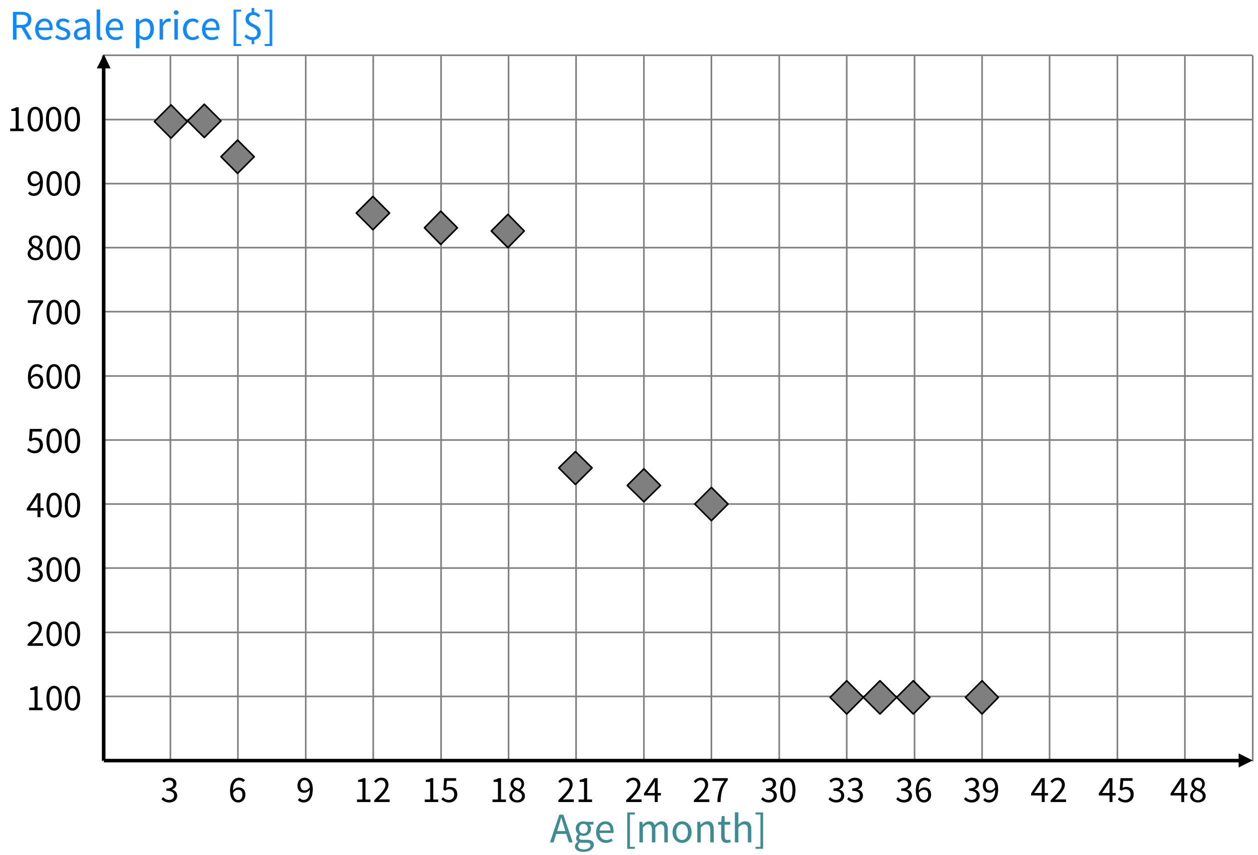
- **Data follows an overall trend**
  - Resale price decreases with age
  - Increasing age by one month roughly decreases resale prices by the same amount
- **Linear regression captures the trend and is a suitable model**
- **But what if the relationship between the target variable and a feature is not linear?**



Linear Regression Example Revisited

Considering a different data set

Age [months]	Resale price [$]
3	1000
4,5	1000
6	950
12	850
15	825
18	825
21	450
24	425
27	400
33	100
34,5	100
36	100
39	100

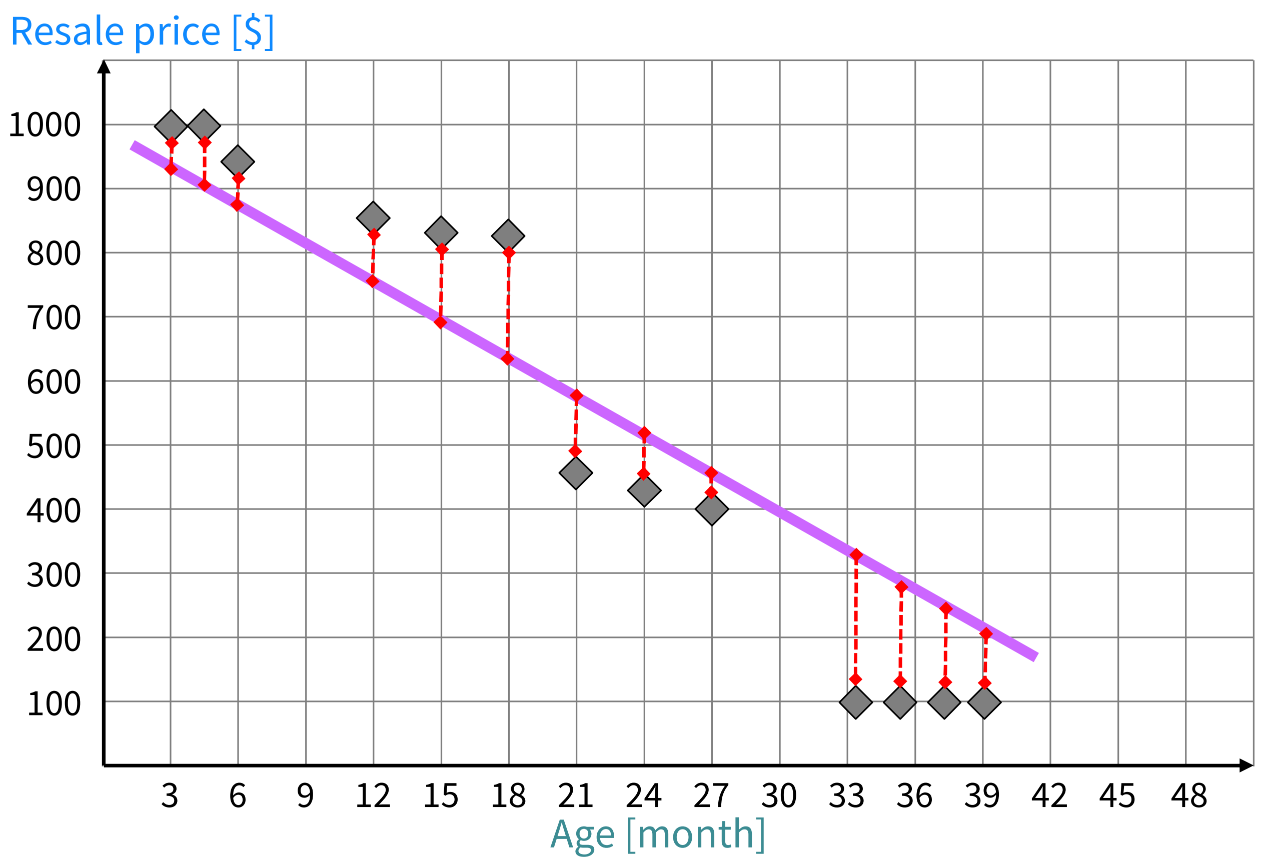


Example of a Nonlinear Relationship
Discontinuities in the feature-to-target relationship impede regression

- **Resale prices decrease with age**
  - But the relationship exhibits jumps
  - The amount by which a one month increase in age decreases resale prices changes across different values of age
  - Hypothetical explanation:

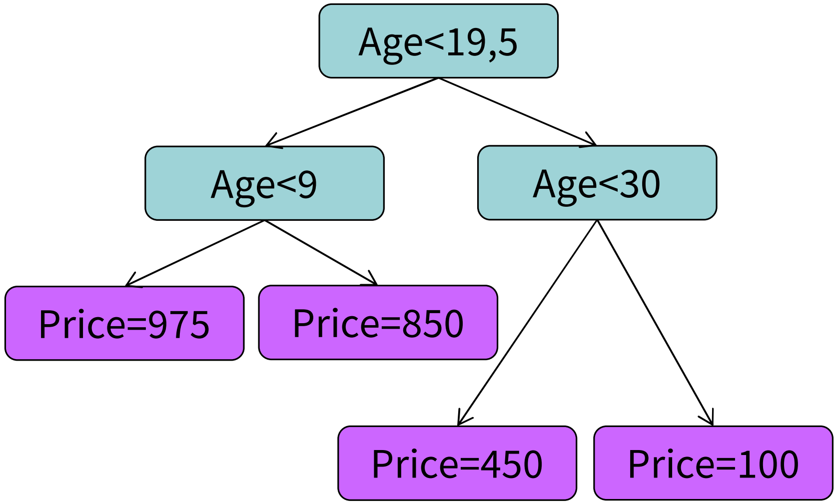
    Say PCs older than, e.g., 30 month are not resold as PC.

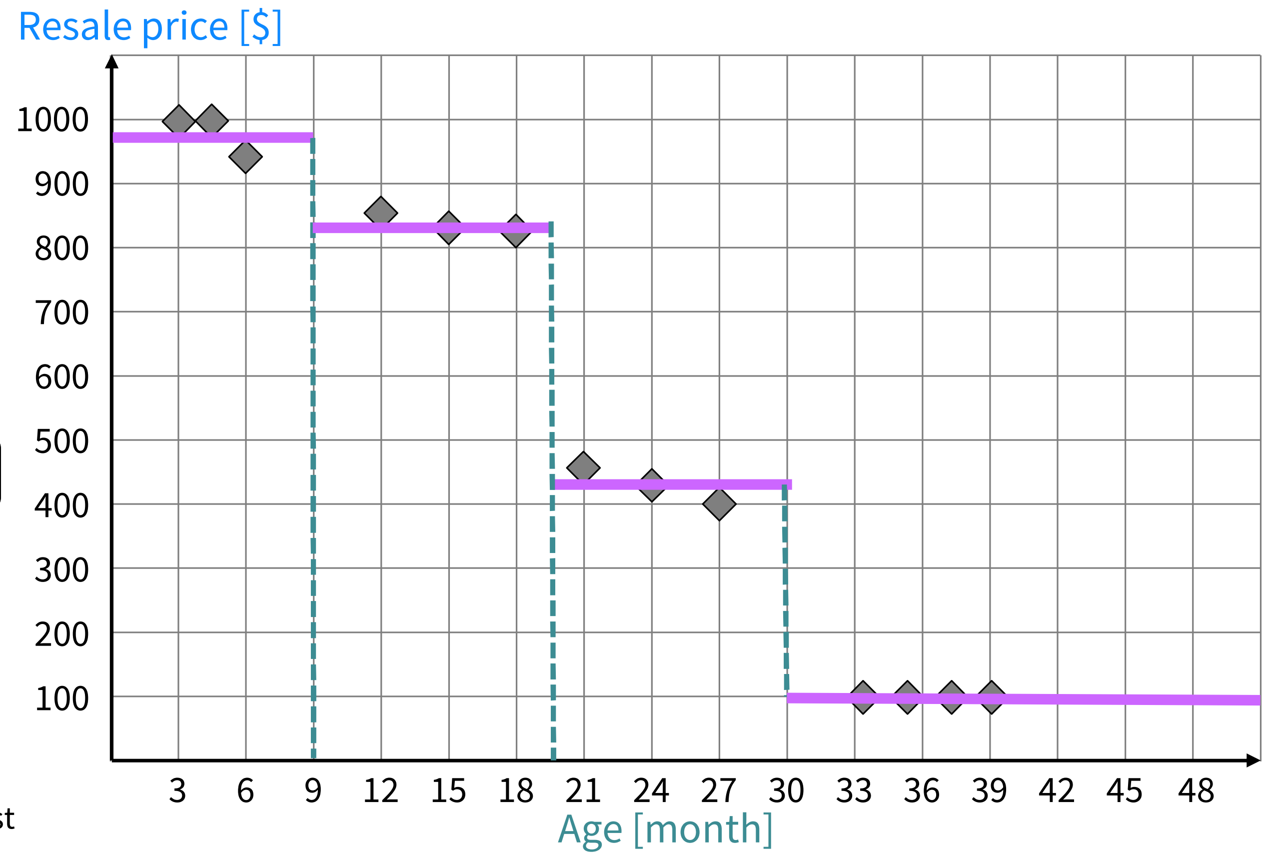
    Instead, components are resold individually. This implies a new relationship between age and prices.
- **Linear regression is not suitable**
  - A line does not represent the data well
  - We observe relatively high residuals



Regression Tree Example

Recursive partitioning of the data into homogeneous subgroups





A regression tree partitions the data by a sequence of yes-no questions. These splits or decision rules compare the value of the feature Age to a threshold. Subgroups that are not split further are called *leaf or terminal nodes*. Each leaf node is associated with a prediction of the resale price. This forecast is equal to the average resale price of the data points in that leaf node.

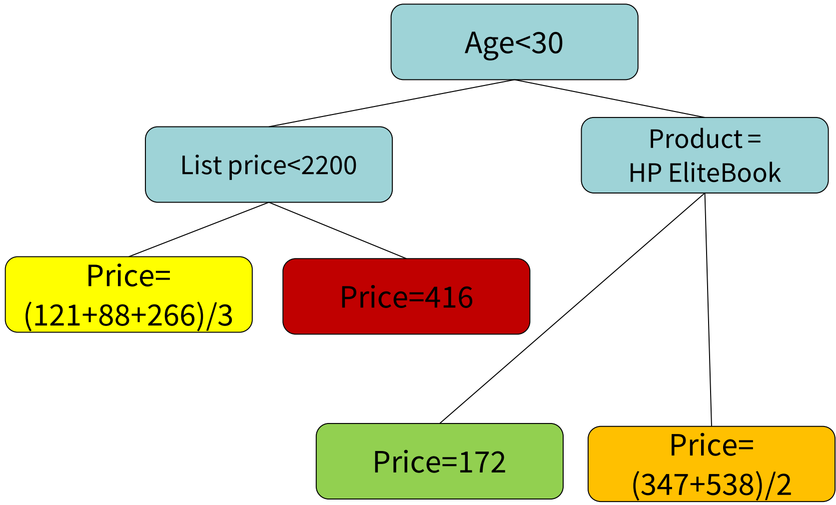
Regression Trees in a Nutshell

Training Find the structure of the tree using the training data.

- Split on which feature?
- Split on which threshold?
- When stop splitting?

Training data incl. Y

Product	List price	Age	…	Resale price
Dell XPS 15’	2,500	36	…	347
Dell XPS 15’	2,500	24	…	416
Dell XPS 17’	3,000	36	…	538
HP Envy 17’	1,300	24	…	121
HP EliteBook	1,900	36	…	172
Lenovo Yoga 11’	799	12	…	88
Lenovo Yoga 13’	1,100	12	…	266
…	…	…	…	…



Testing Identify the right leaf node for a new example.

- Put example down the tree
- Apply decision rules to feature values
- Forecasts equals leaf node prediction

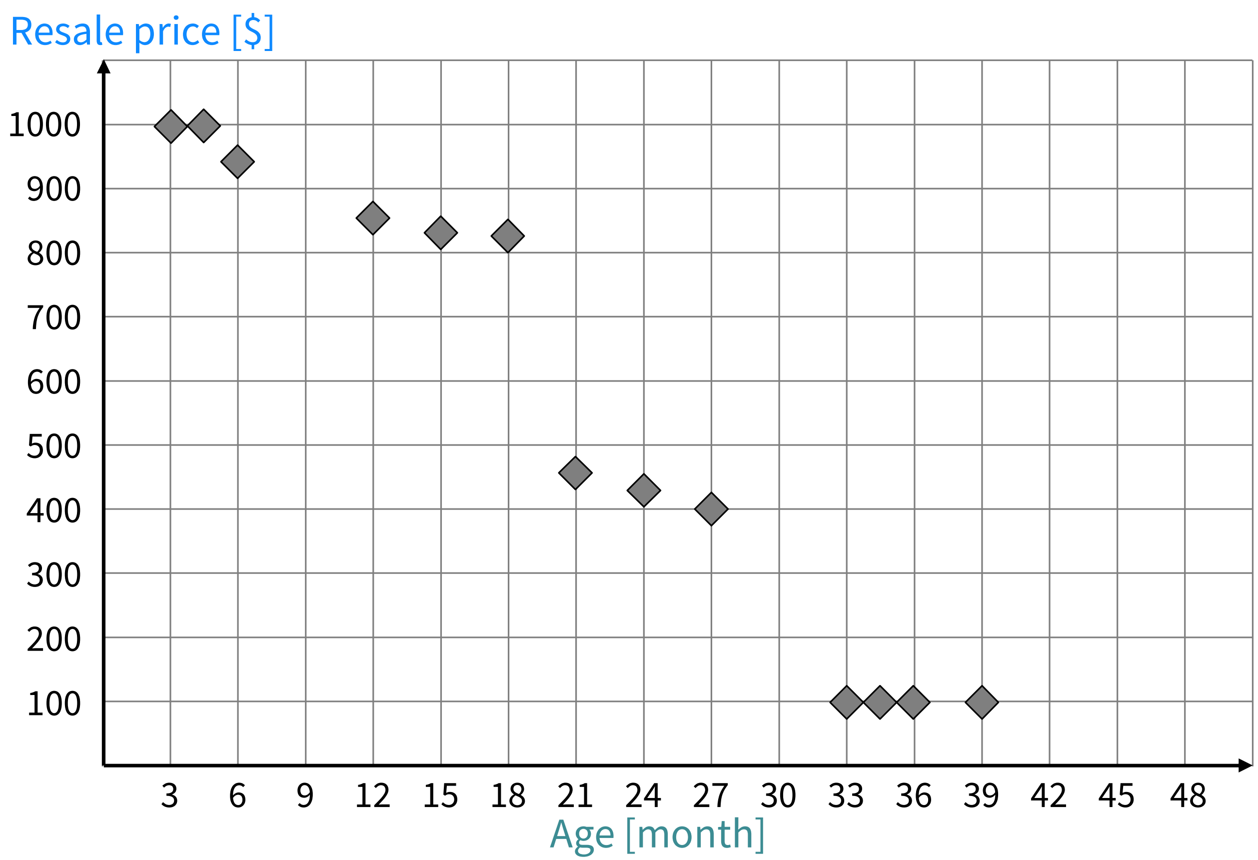
Product	List price	Age	…	Tree Forecast
HP Envy 15’	1,150	12	…	158,33
Lenovo Yoga 13’	1,100	36	…	442,5
Dell XPS 15’	2,500	12	…	416
HP EliteBook	2,100	48	…	172
Lenovo Yoga 14’	2,300	24	…	416
HP EliteBook	1,900	24	…	158,33
…	…	…	…	…

Key Steps in Regression Tree Learning

- **Deciding how to grow a tree, that is on split points**
  - Recursive partitioning approach
    - Find an optimal split partitioning the entire training data into two subgroups (child nodes)
    - For each child node, find an optimal split for the data in that child node
    - Repeat the partitioning of the data until some stopping criterion is met
  - Assessing candidate splits using a loss function
    - Same concept as in linear regression
    - Minimize the sum of squared residuals
- **Deciding when to stop tree growing**
  - Complex trees with many leaves will not forecast accurately (overfitting problem, see later)
  - Tree pruning seeks a balance between fitting the training data well while not letting the tree become too complicated

Regression Tree Learning: Finding an Optimal Split

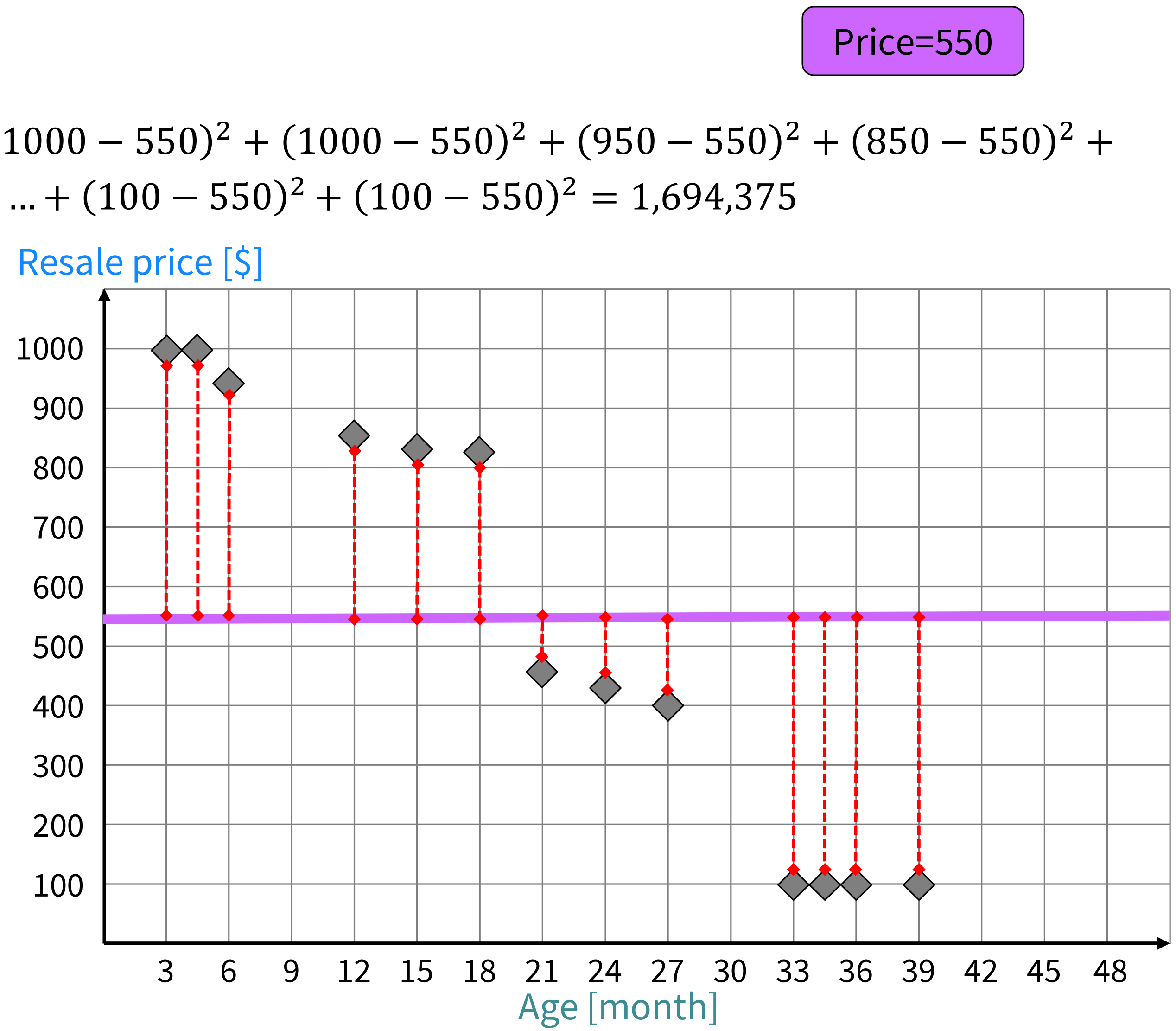
- **Exemplary training data set**
  - One feature Age
  - N=13 data points
- **A split partitions the training set into two subgroups**
  - A subgroup must not be empty
  - We have N-1 ways to split the data on the single feature Age
- **Assessing candidate splits**
  - Tree forecast for a subgroup is constant
  - Average resale price among subgroup members
  - Since this is the training data, we know actual prices and can compute averages
  - So we can also compute residuals as: true resale price – tree forecast



Assessing Candidate Splits

$$SSR = (1000 - 550)^2 + (1000 - 550)^2 + (950 - 550)^2 + (850 - 550)^2 + \dots + (100 - 550)^2 + (100 - 550)^2 = 1{,}694{,}375$$

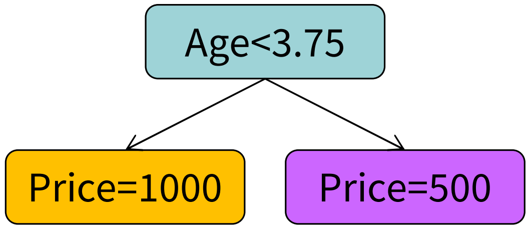
- **Scenario before splitting**
  - The root of the tree includes the entire training data set
  - The forecast of this simplest possible 'tree' is the average resale price in the training set
  - For the example data, the average resale price is equal to 550
- **Residuals**
  - Difference between observed resale prices and 'tree' forecast
  - Total sum of squared residuals (SSR) tells how well the tree represents the training data



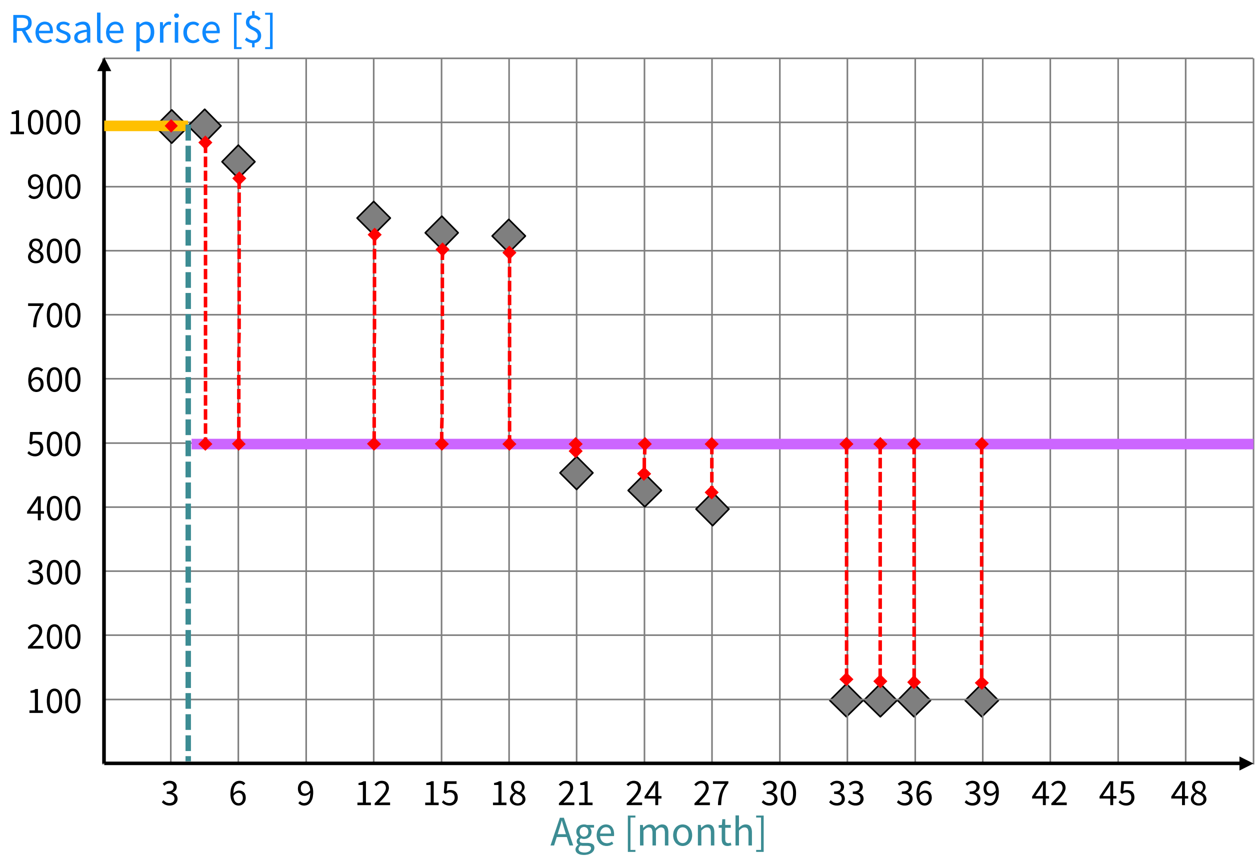
Finding an Optimal Split

Enumerative search strategy

$$SSR = (1000 - 1000)^2 + (1000 - 500)^2 + \ldots + (100 - 500)^2 = 1{,}443{,}073$$



- **Begin with the two data points with smallest Age**
- **Assess split that cuts the data between these points**
  - Age values of the two example are, respectively, 3 and 4.5
  - So consider split Age < 3.75
- **Determine tree forecasts for the two leaves**
- **Compute SSR of this split**

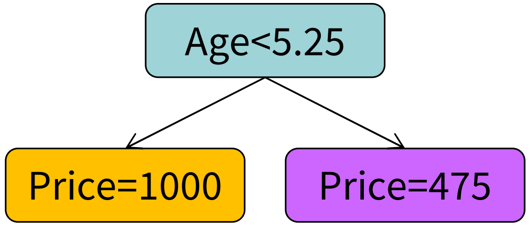


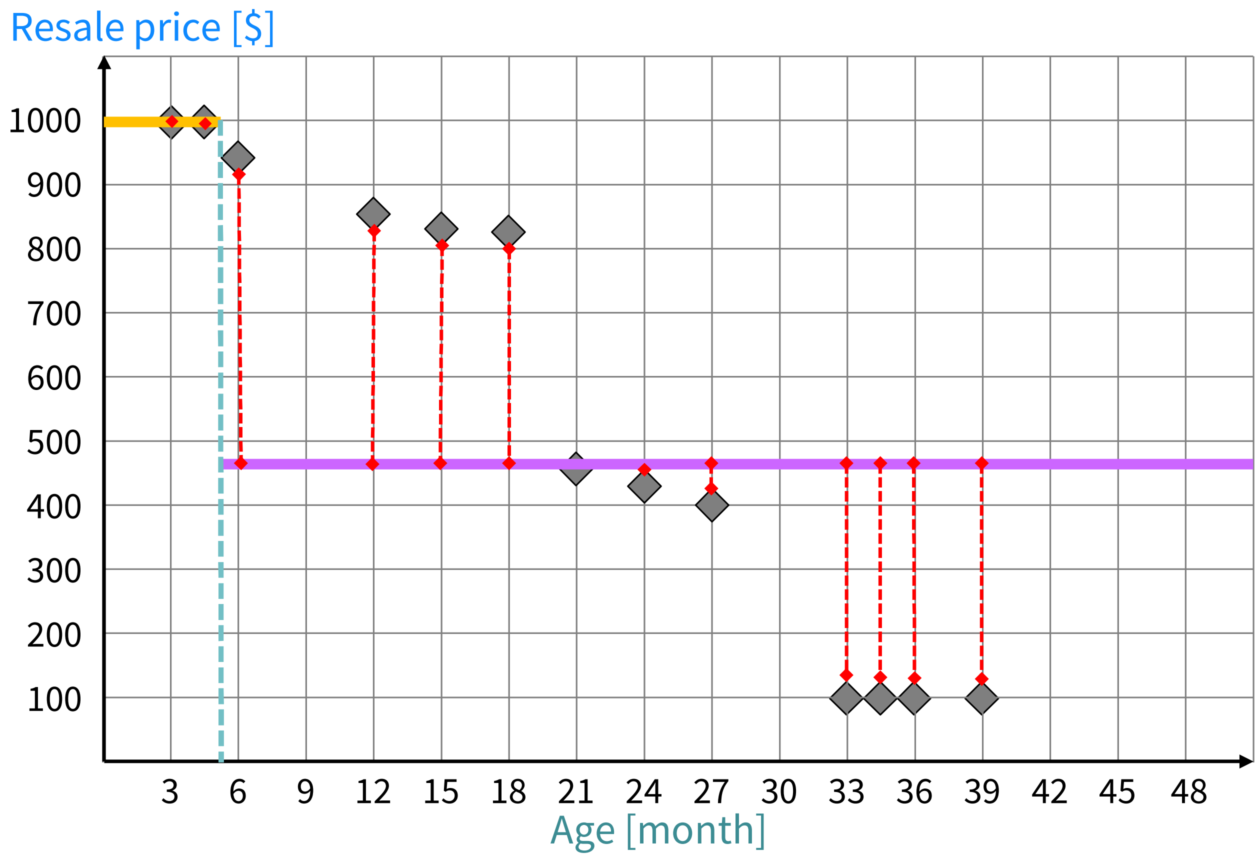
Finding an Optimal Split

Enumerative search strategy

- **Proceed with the next pair of neighboring data points**
- **Assess split that cuts the data between these points**
  - Age values of the two example are, respectively, 4.5 and 6
  - So consider split Age < 5.25
- **Determine tree forecasts for the two leaves**
- **Compute SSR of this split**

$$SSR = (1000 - 1000)^2 + (1000 - 1000)^2 + (950 - 475)^2 + \cdots + (100 - 475)^2 = 1{,}443{,}073$$

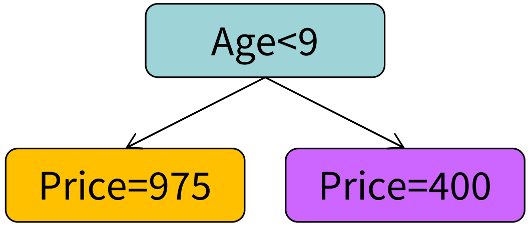


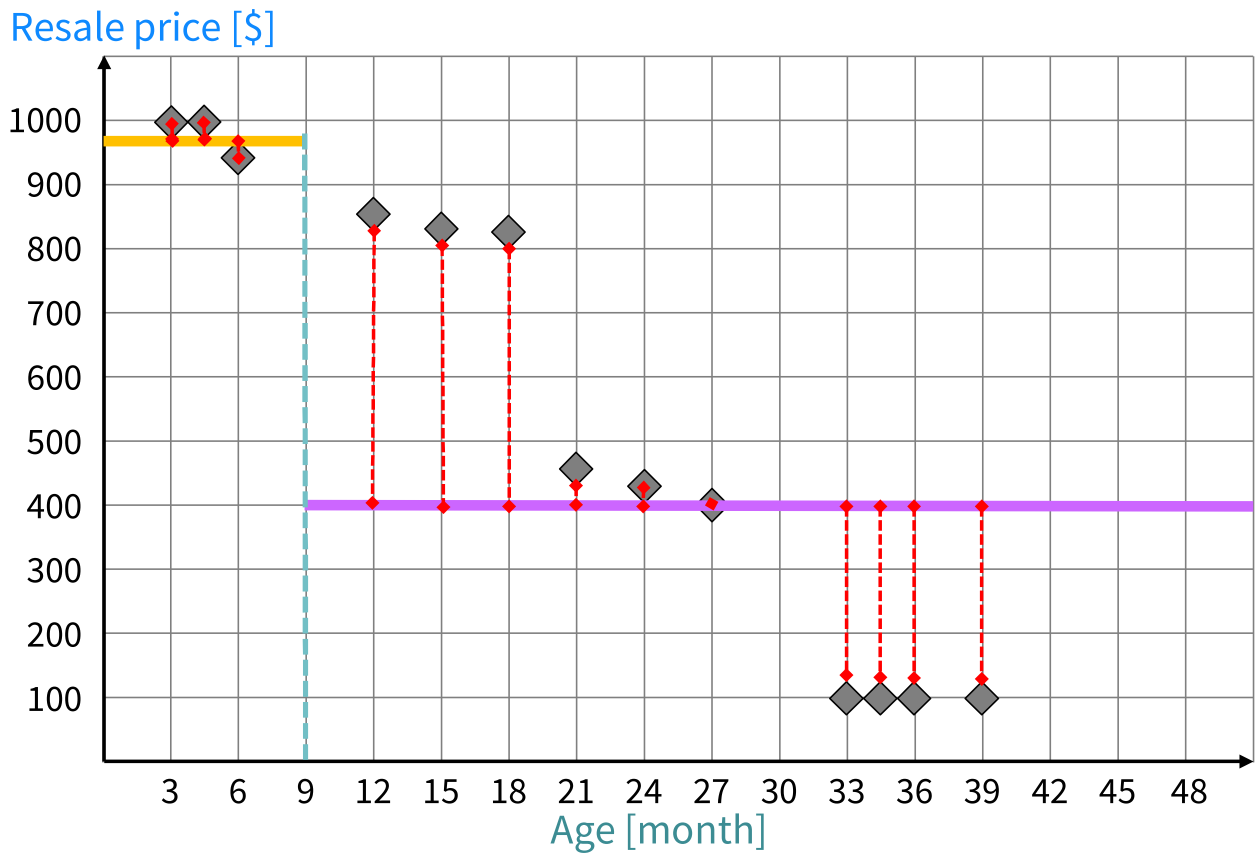


Finding an Optimal Split
Enumerative search strategy

- **Continue in the same way until all candidate splits have been explored**
- **Keep track of the SSR**

$$SSR = (1000 - 975)^2 + \dots + (950 - 975)^2 + (850 - 400)^2 + \dots + (100 - 400)^2 = 925{,}479$$

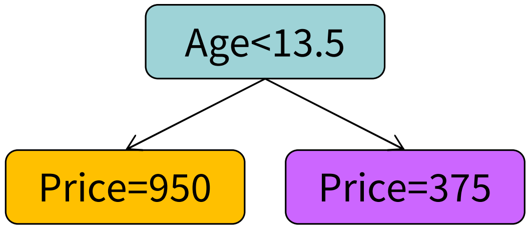


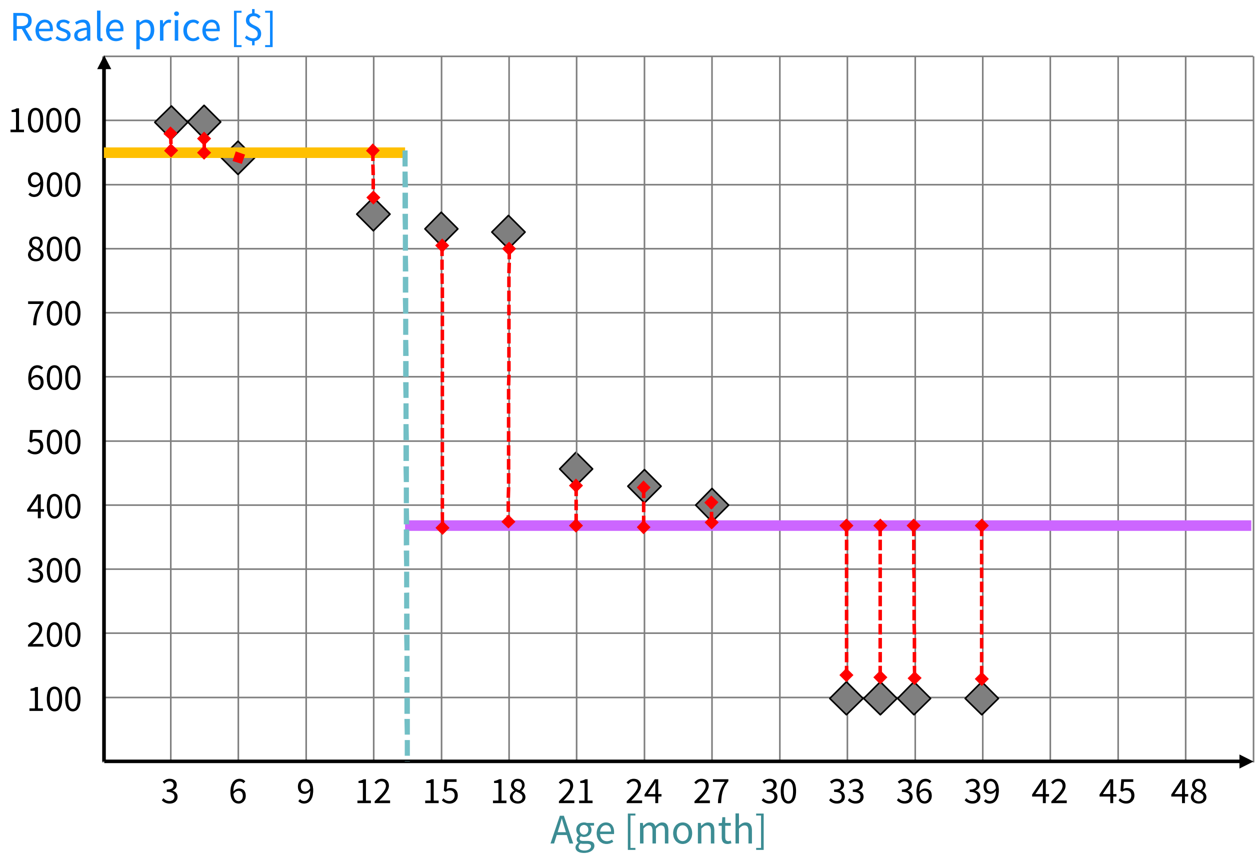


Finding an Optimal Split
Enumerative search strategy

- **Continue in the same way until all candidate splits have been explored**
- **Keep track of the SSR**

$$SSR = (1000 - 950)^2 + \dots + (850 - 950)^2$$
$$+ (825 - 375)^2 + \dots + (100 - 375)^2$$
$$= 730{,}972$$



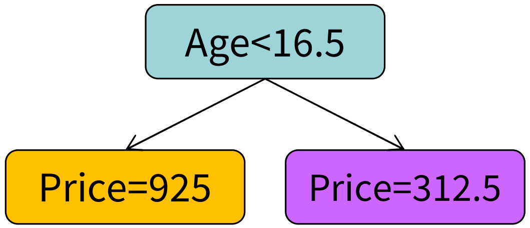


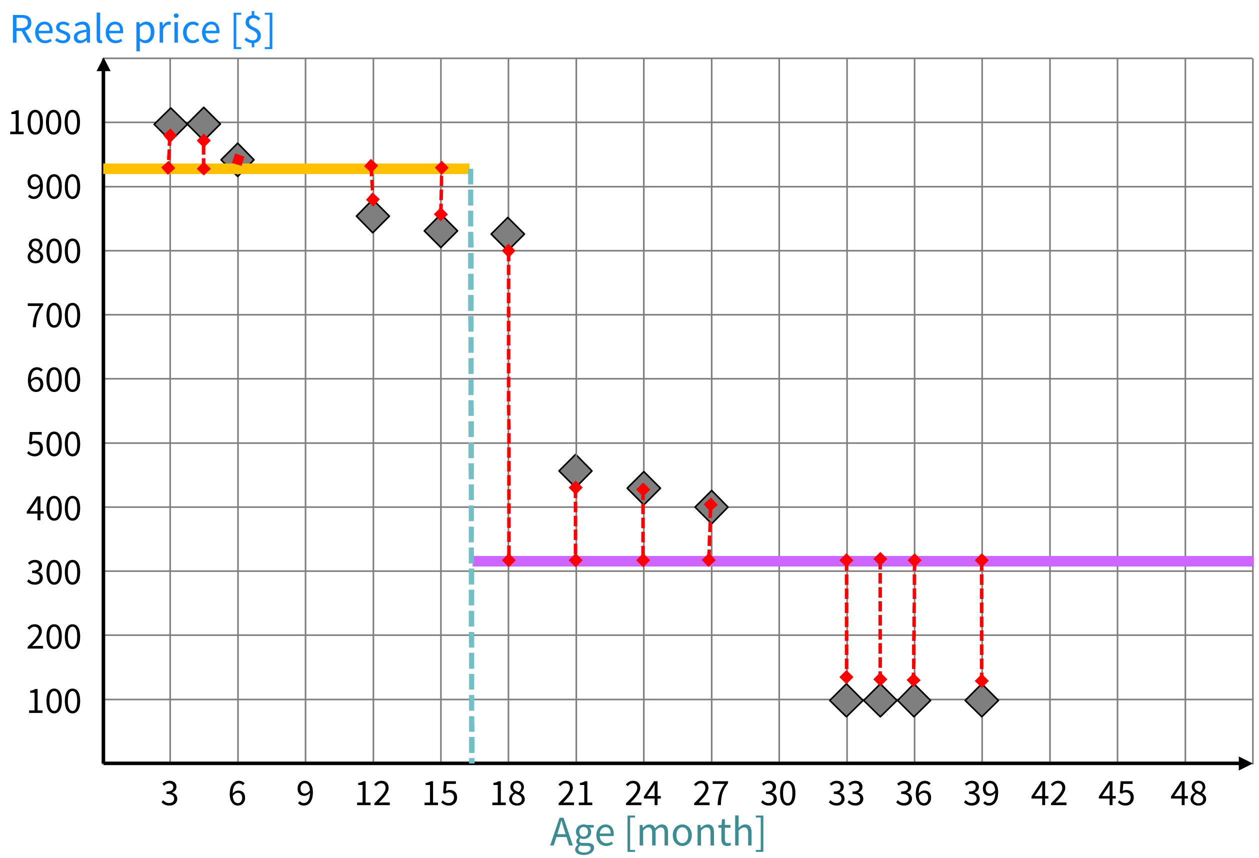
Finding an Optimal Split

Enumerative search strategy

- **Continue in the same way until all candidate splits have been explored**
- **Keep track of the SSR**

$$SSR = (1000 - 925)^2 + \ldots + (825 - 925)^2$$
$$+ (825 - 312.5)^2 + \cdots + (100 - 312.5)^2$$
$$= 510{,}000$$

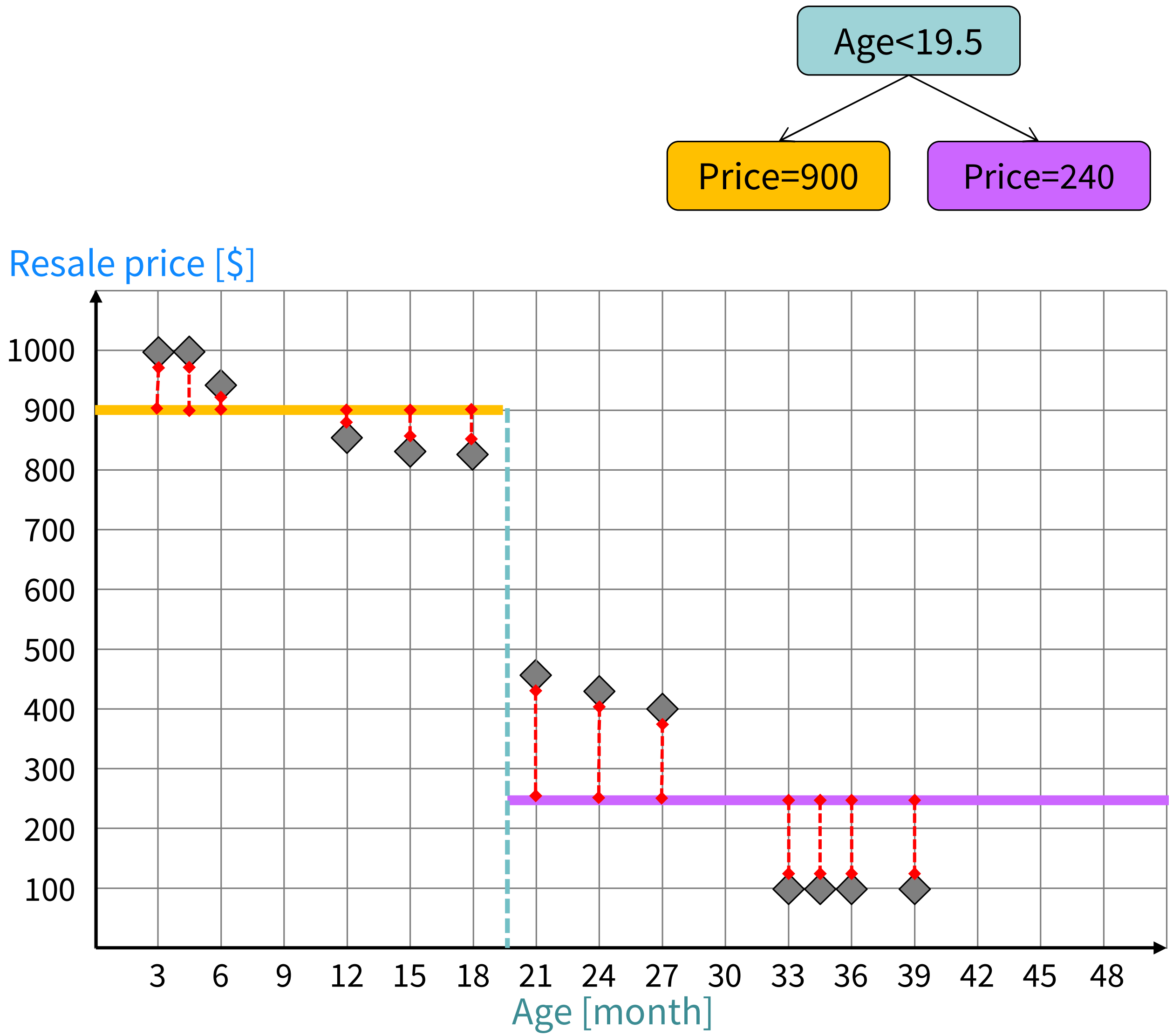




Finding an Optimal Split
Enumerative search strategy

- **Continue in the same way until all candidate splits have been explored**
- **Keep track of the SSR**

$$SSR = (1000 - 900)^2 + \ldots + (825 - 900)^2$$
$$+ (450 - 240)^2 + \cdots + (100 - 240)^2$$
$$= 218{,}155$$

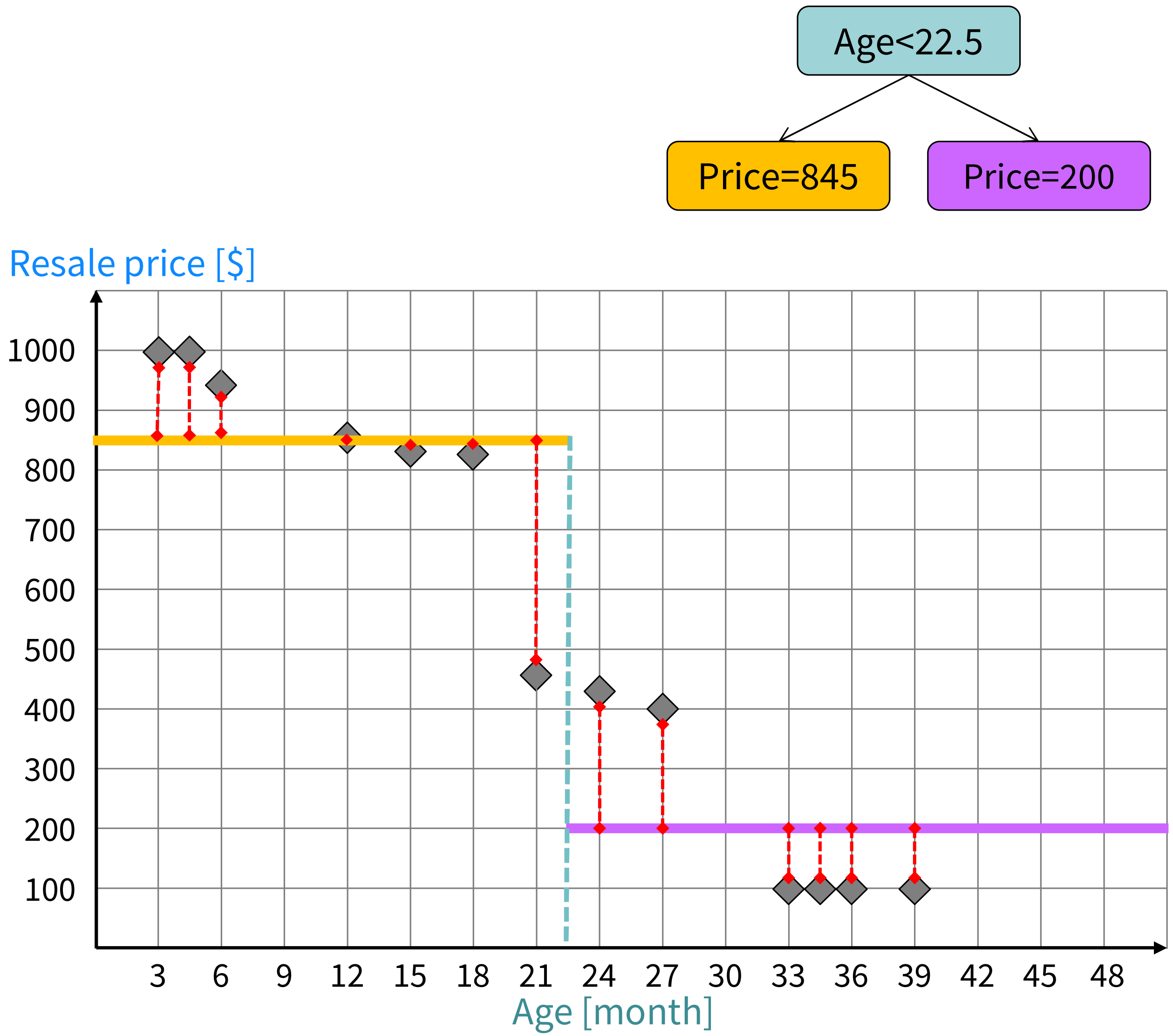


Finding an Optimal Split

Enumerative search strategy

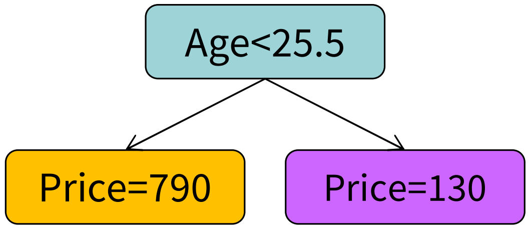
- **Continue in the same way until all candidate splits have been explored**
- **Keep track of the SSR**

$$SSR = (1000 - 845)^2 + \ldots + (450 - 845)^2$$
$$+(425 - 200)^2 + \cdots + (100 - 200)^2$$
$$= 346{,}414$$



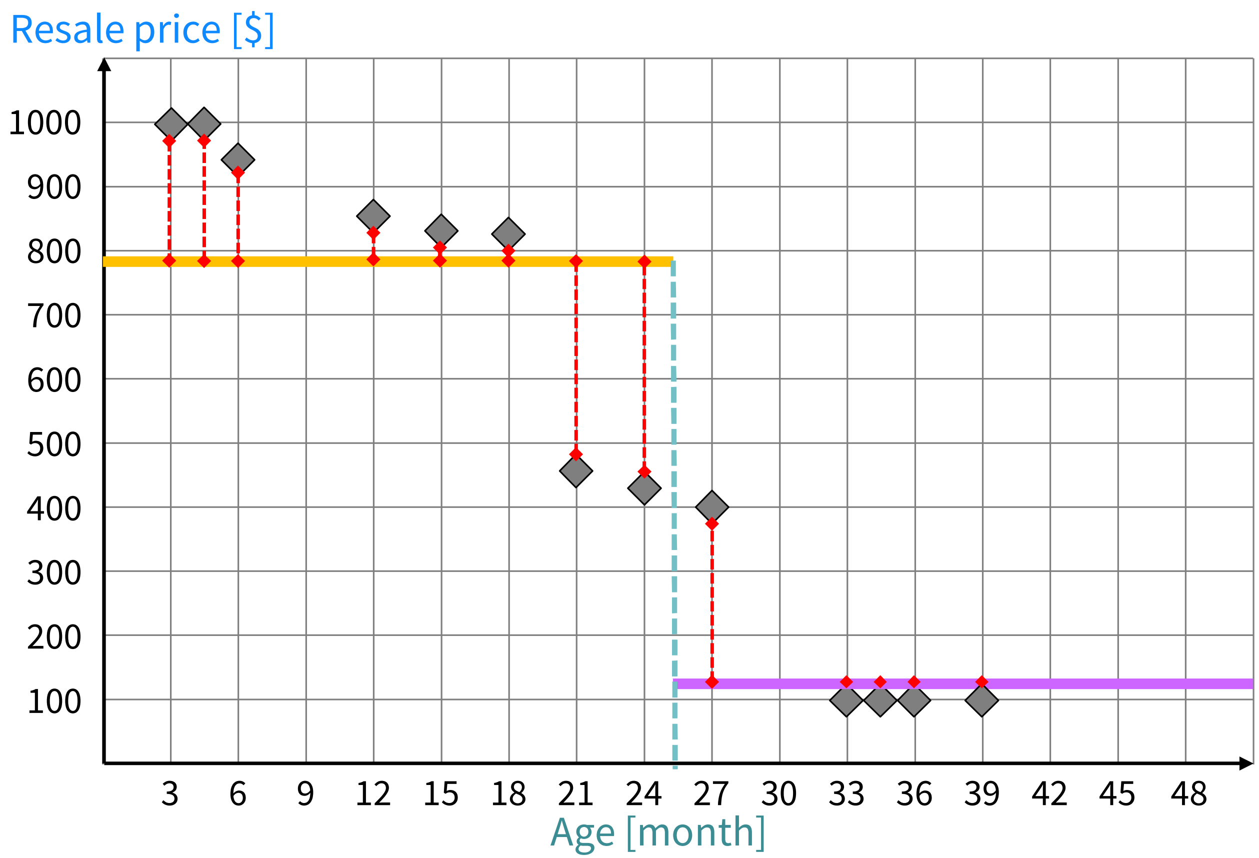
Finding an Optimal Split

Enumerative search strategy

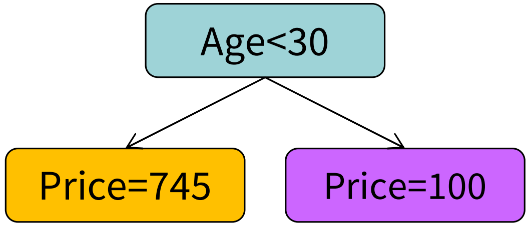


- **Continue in the same way until all candidate splits have been explored**
- **Keep track of the SSR**

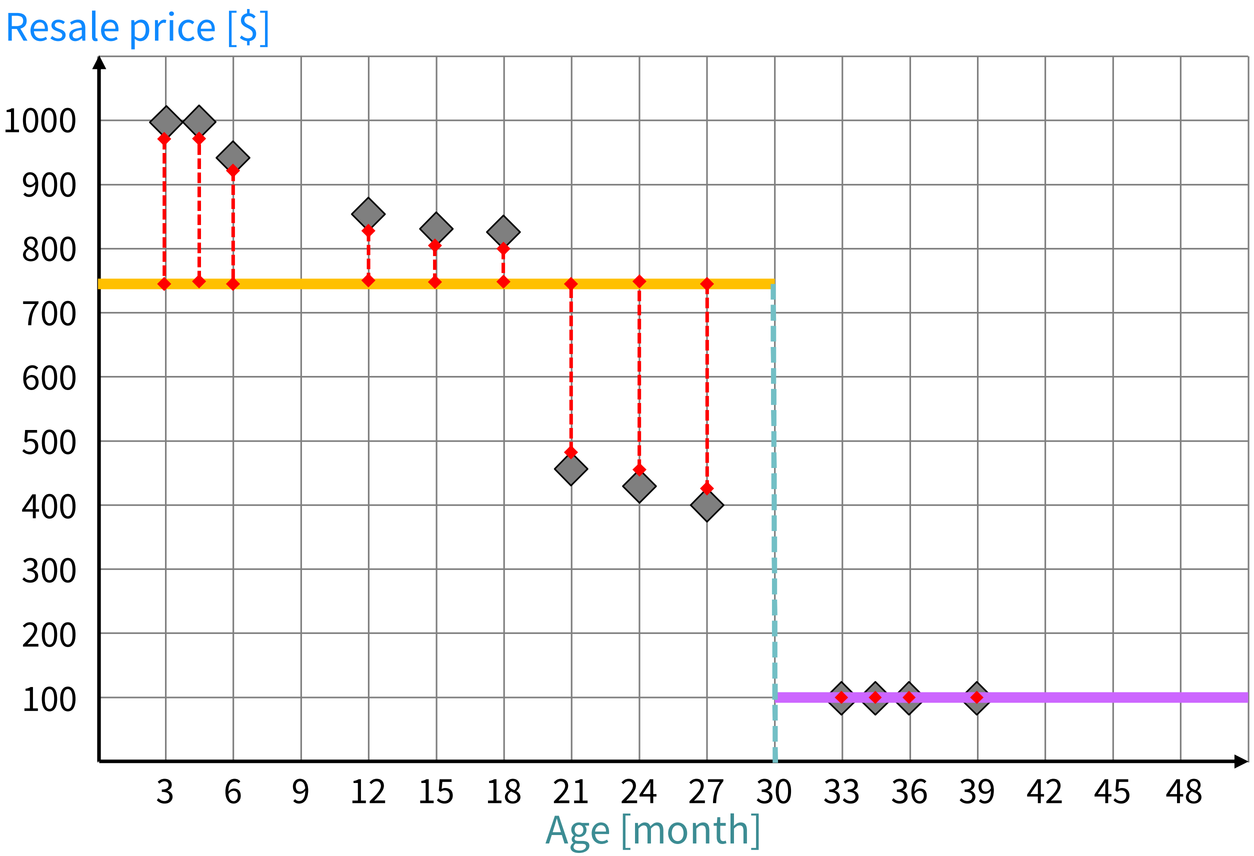
$$SSR = (1000 - 790)^2 + \ldots + (425 - 790)^2$$
$$+ (400 - 130)^2 + \cdots + (100 - 130)^2$$
$$= 440{,}672$$



Finding an Optimal Split
Enumerative search strategy



- **Continue in the same way until all candidate splits have been explored**
- **Keep track of the SSR**



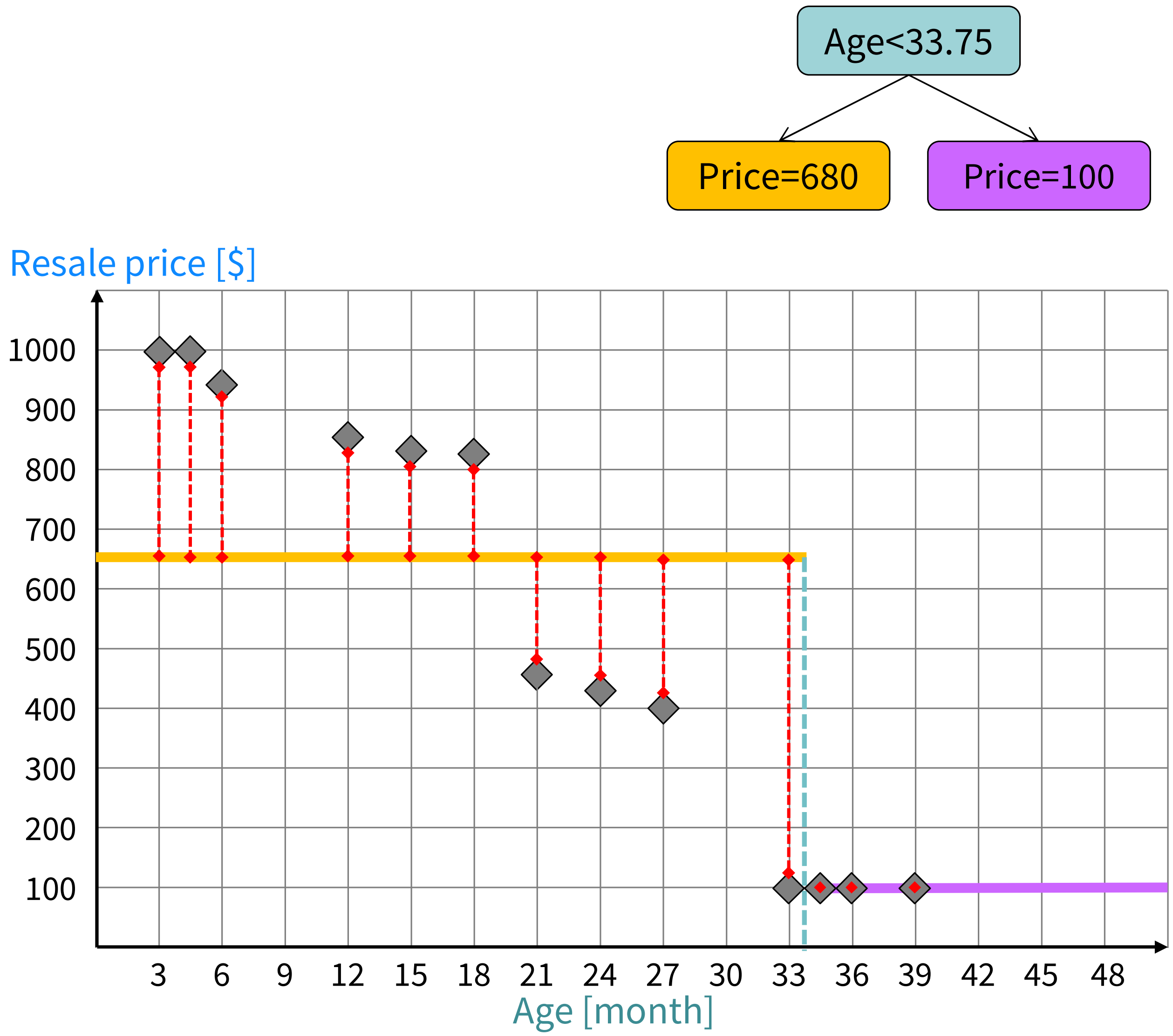
$$SSR = (1000 - 745)^2 + \ldots + (400 - 745)^2$$
$$+(100 - 100)^2 + \cdots + (100 - 100)^2$$
$$= 504{,}306$$

Finding an Optimal Split

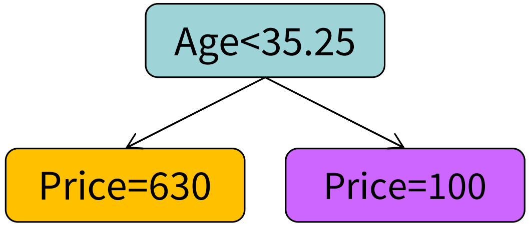
Enumerative search strategy

- **Continue in the same way until all candidate splits have been explored**
- **Keep track of the SSR**

$$SSR = (1000 - 680)^2 + \dots + (100 - 680)^2$$
$$+(100 - 100)^2 + \cdots + (100 - 100)^2$$
$$= 881{,}313$$

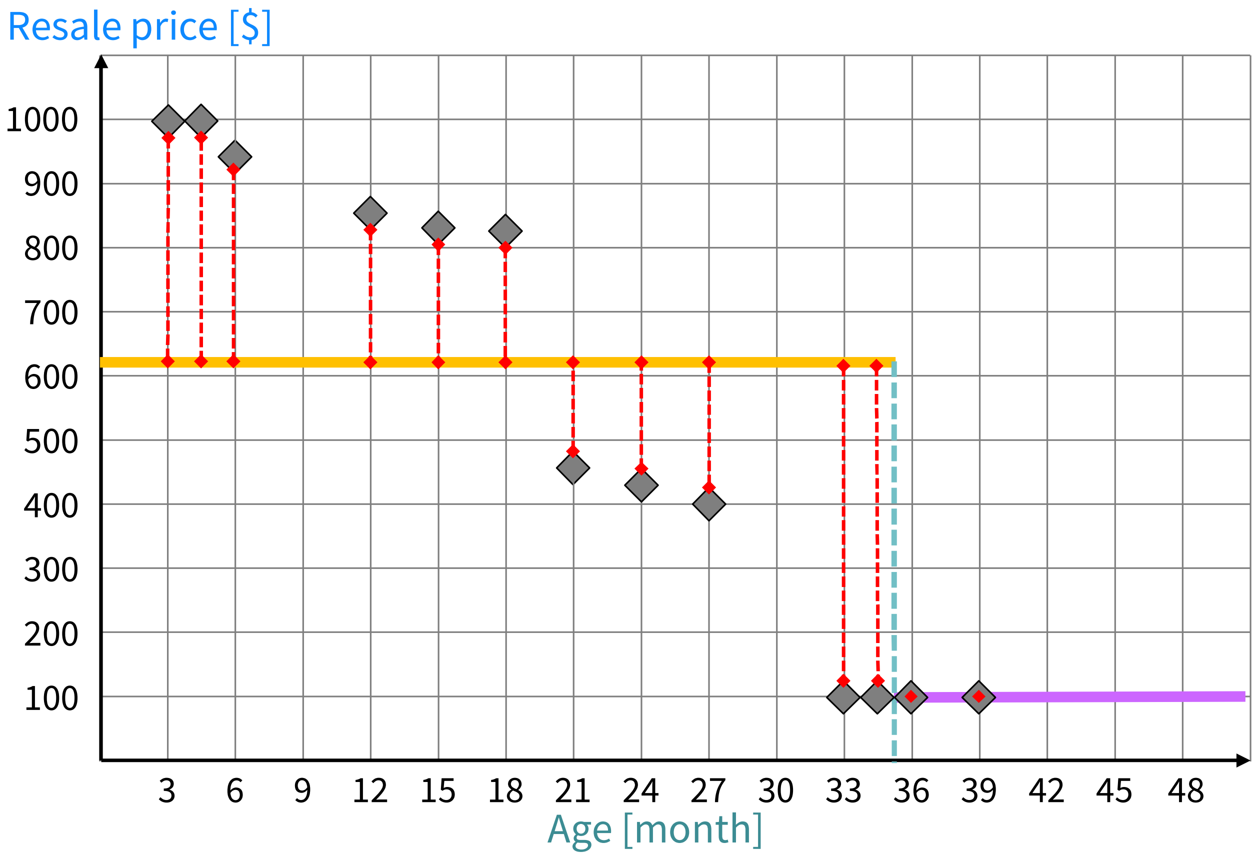


Finding an Optimal Split
Enumerative search strategy



- **Continue in the same way until all candidate splits have been explored**
- **Keep track of the SSR**

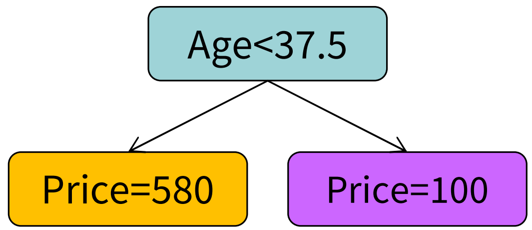
$$SSR = (1000 - 630)^2 + \ldots + (100 - 630)^2$$
$$+(100 - 100)^2 + (100 - 100)^2$$
$$= 1{,}189{,}773$$

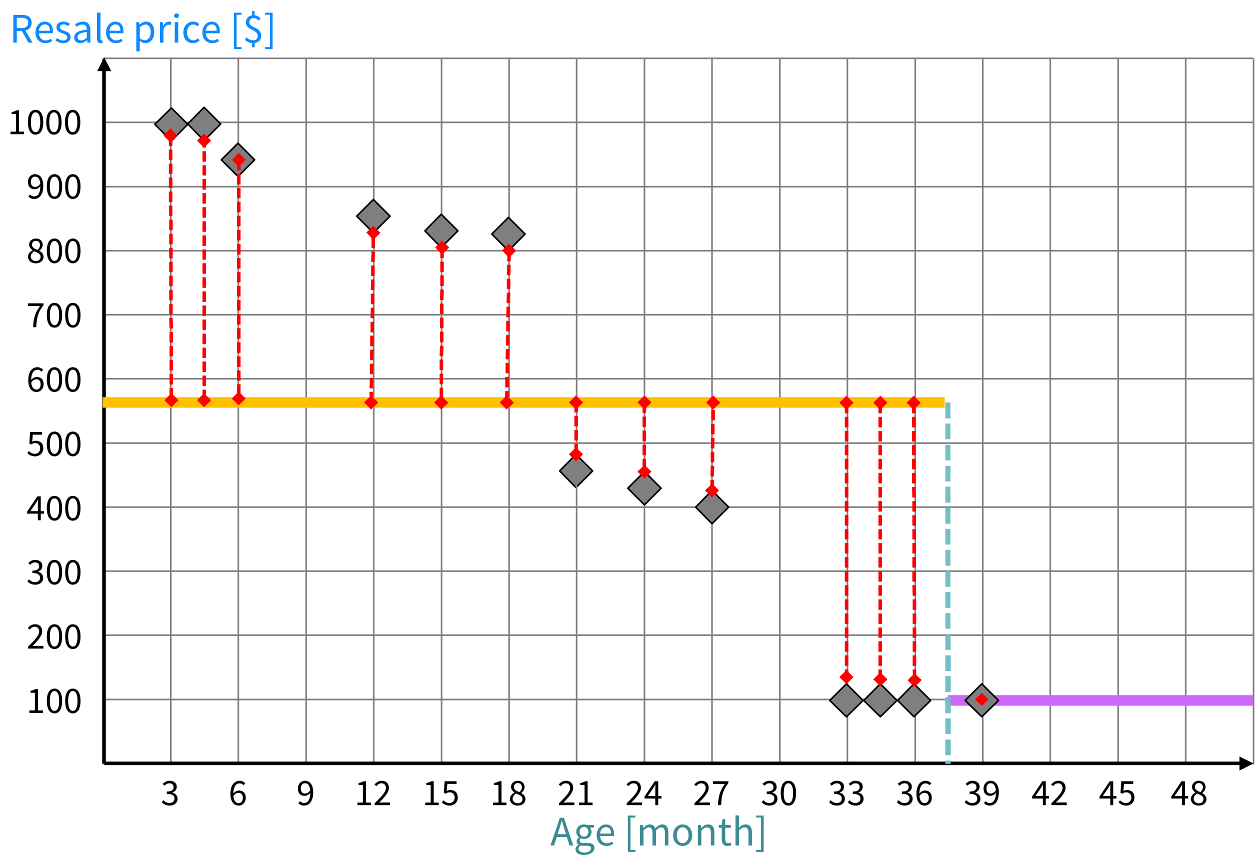


Finding an Optimal Split

Enumerative search strategy

- **Continue in the same way until all candidate splits have been explored**
- **Keep track of the SSR**





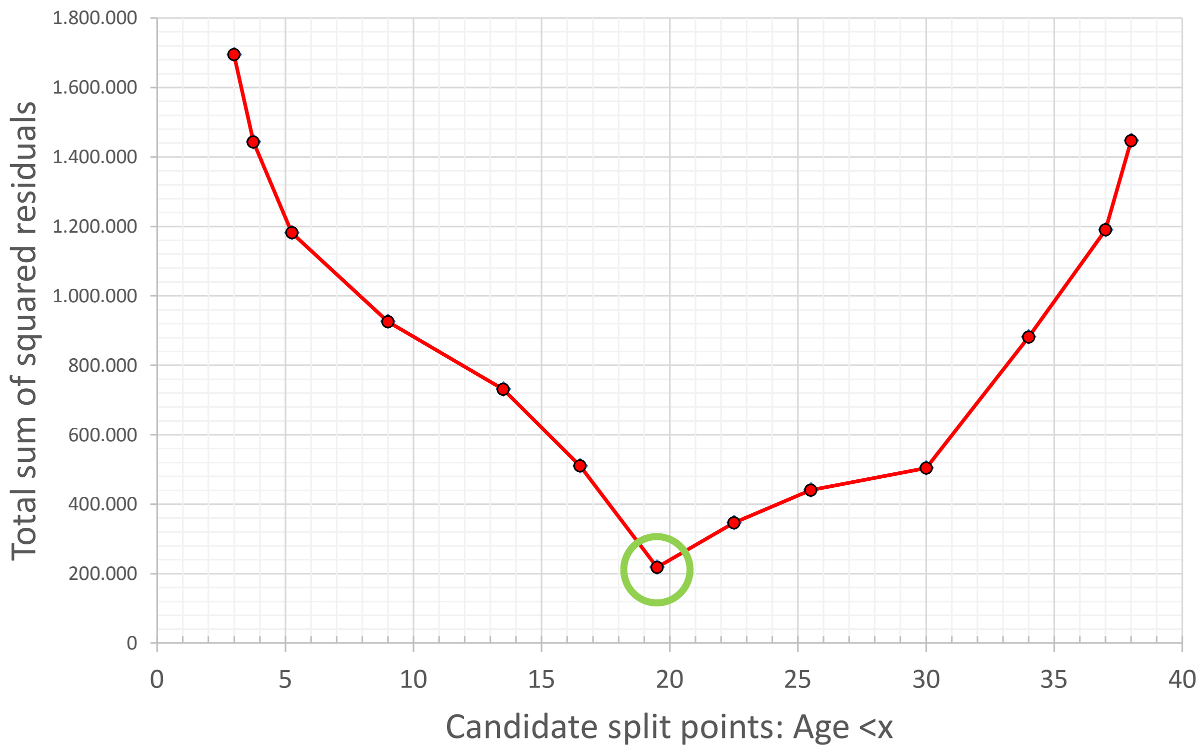
$$SSR = (1000 - 580)^2 + \ldots + (100 - 580)^2$$
$$+ (100 - 100)^2$$
$$= 1{,}446{,}823$$

Finding an Optimal Split

Select the overall best split among all candidate splits

- **We obtain a list of candidate split points on feature Age**
- **And the corresponding SSR**

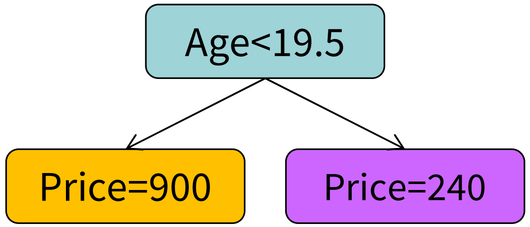
Candidate split Age<x	SSR
3	1.694.375
3,75	1.443.073
5,25	1.181.591
9	925.479
13,5	730.972
16,5	510.000
19,5	218.155
22,5	346.414
25,5	440.672
30	504.306
34	881.313
37	1.189.773
38	1.446.823



Regression Tree Learning Continued

Having found our first split, we search for the next best split in each child

Age [months]	Resale price [$]
3	1000
4,5	1000
6	950
12	850
15	825
18	825
21	450
24	425
27	400
33	100
34,5	100
36	100
39	100



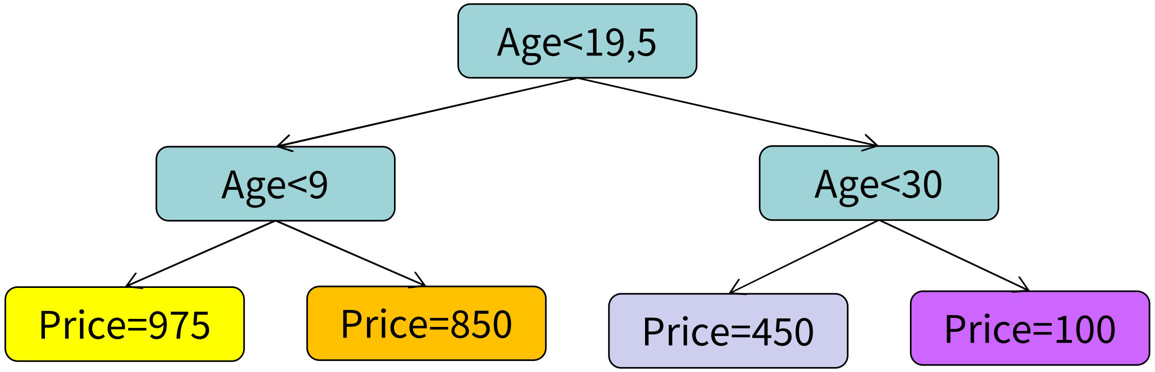
Find best split for these examples

Find best split for these examples

Following exactly the same logic as before. That is, test all candidate splits and pick the one that has minimal SSR.

Regression Tree with Four Leaf Nodes

Age [months]	Resale price [$]
3	1000
4,5	1000
6	950
12	850
15	825
18	825
21	450
24	425
27	400
33	100
34,5	100
36	100
39	100



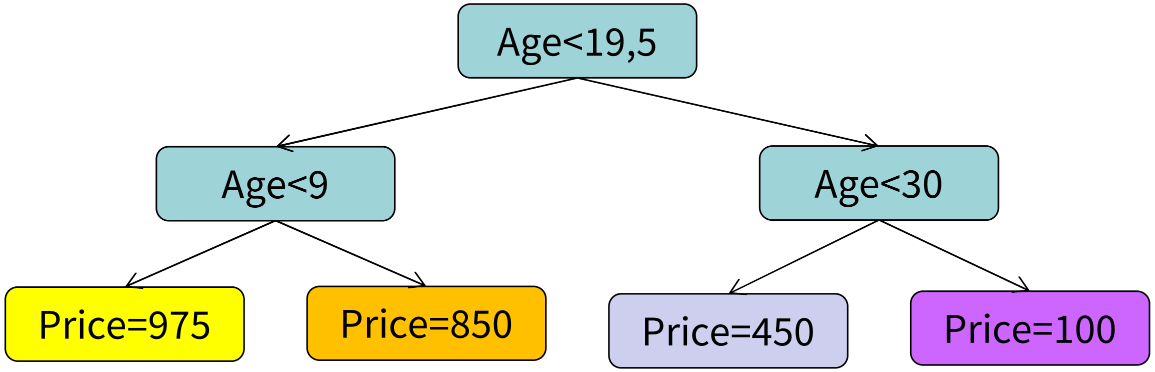
Notice how a tree with k=4 leaf nodes partitions the training set into *k=4* subgroups.

Also recall that each leaf node outputs an estimate of the target variable. This estimate is simply the average value of the target variable computed among the training set examples that fall into the leaf node.

When we process new data and generate forecasts, determine to which leaf node a new example belongs and then take that leaf node's output value as forecast. Consequently, a tree with *k=4* leaf nodes can forecast no more than *k=4* distinct values. In other words, no matter what new data we put down the above tree, the forecasted resale price of the tree will always be a value in {975, 850, 450, 100}.

Regression Tree with Four Leaf Nodes

Age [months]	Resale price [$]
3	1000
4,5	1000
6	950
12	850
15	825
18	825
21	450
24	425
27	400
33	100
34,5	100
36	100
39	100



In theory, we can continue growing the tree until

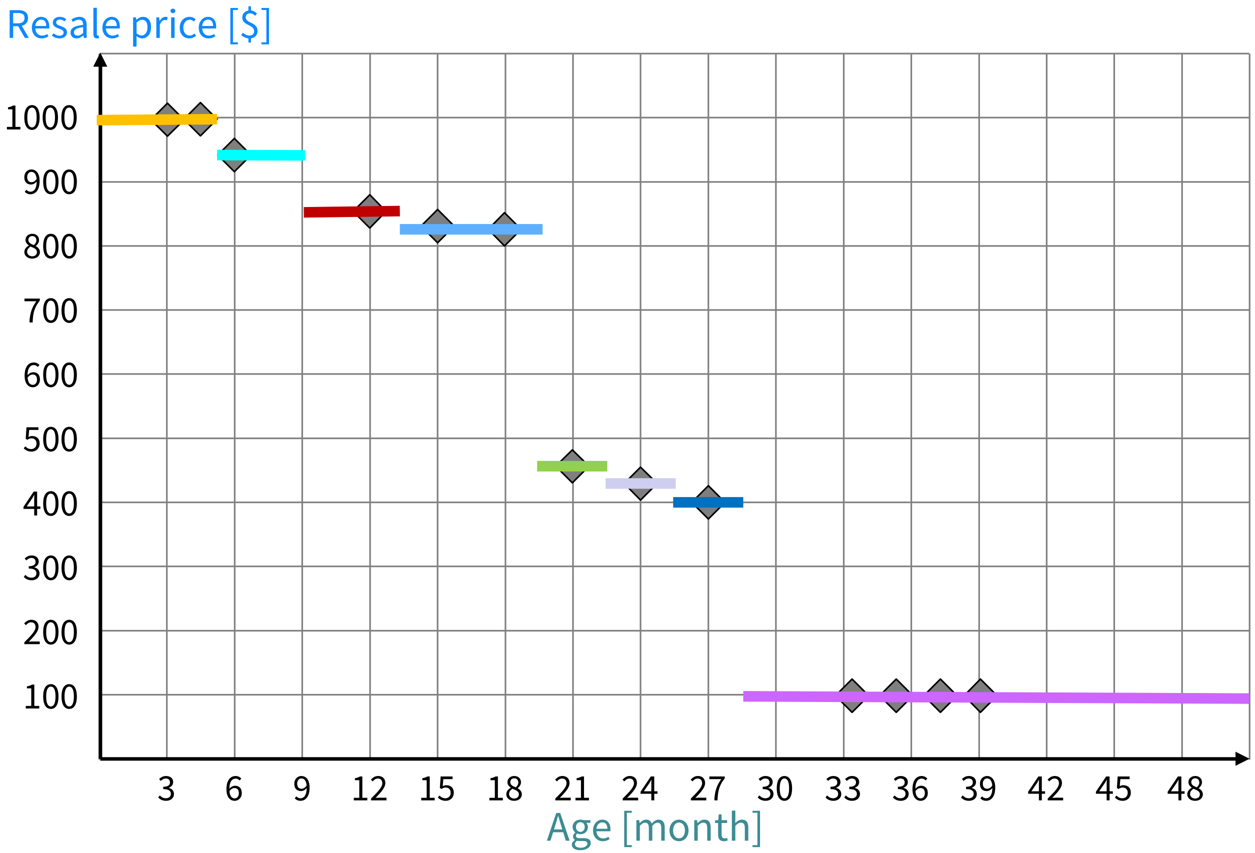
- Each leaf node has only one data point
- Further splits do no longer improve SSR

But would that be a good idea?

Maximally Grown Regression Tree for our Data

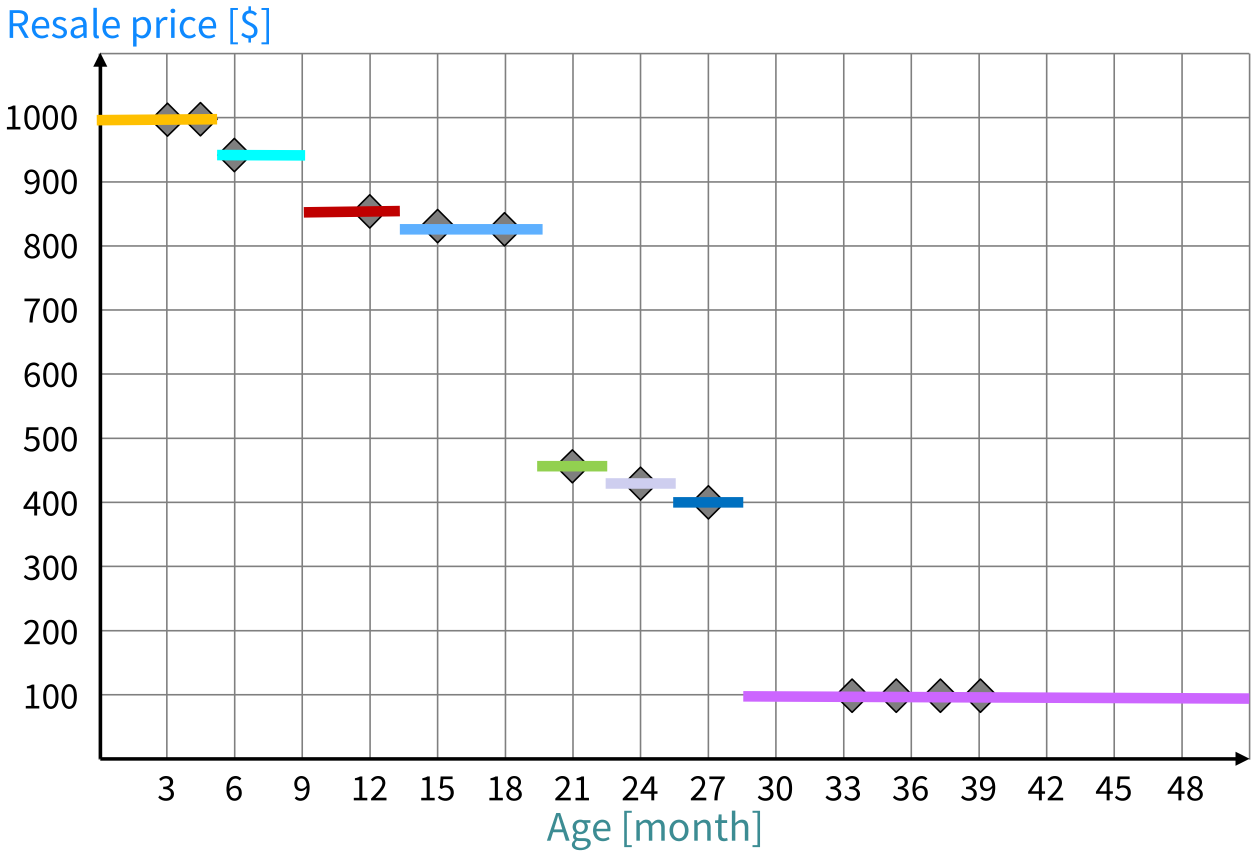
SSR in each leaf node is zero so further splits do not make sense

Age [months]	Resale price [$]
3	1000
4,5	1000
6	950
12	850
15	825
18	825
21	450
24	425
27	400
33	100
34,5	100
36	100
39	100



Recall the Fundamental Problem of Overfitting

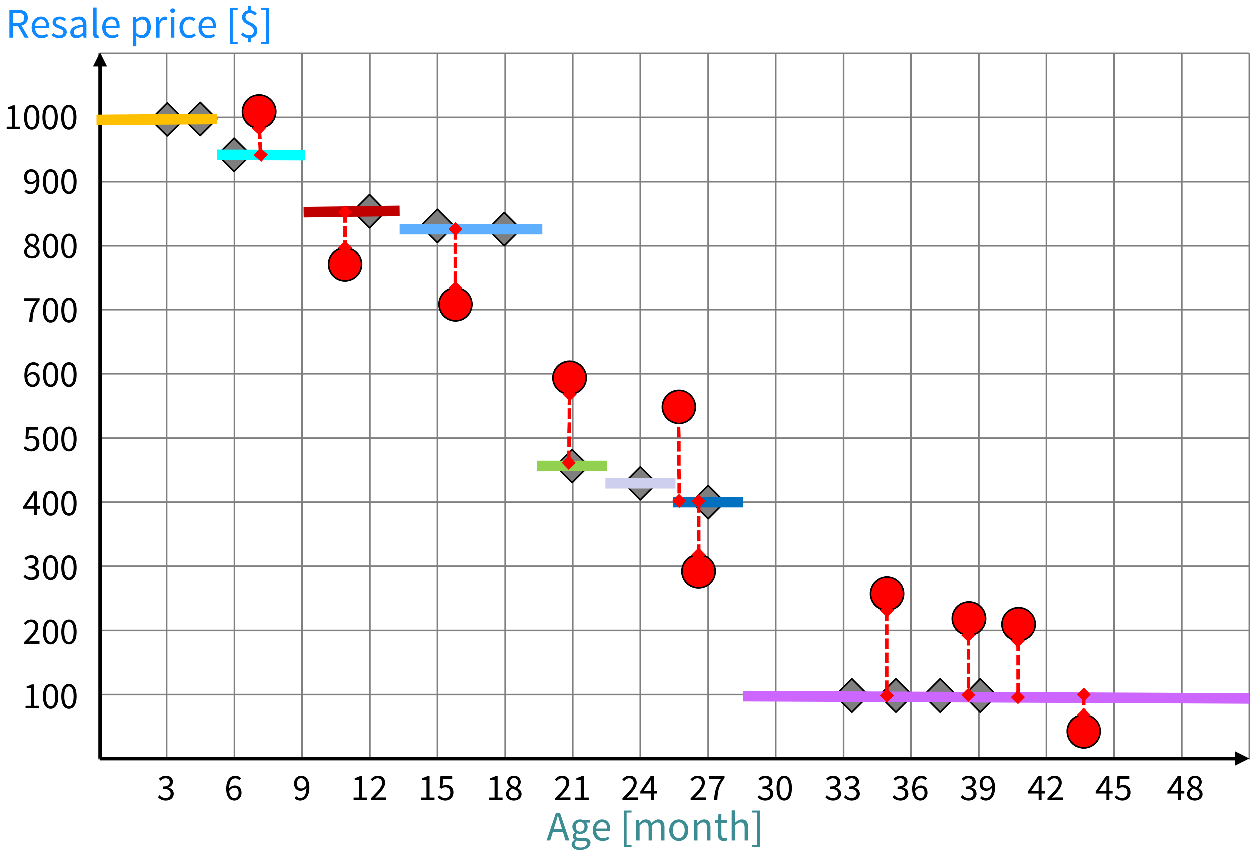
- **Tree is grown on the training sample**
  - True structure (i.e., how the target depends on feature values)
  - Random variation (i.e., noise)
- **Deep, complex trees likely overfit the training data**
  - They capture both, true structure & noise
  - Ability of the tree to forecast novel data likely much worse compared to training



Recall the Fundamental Problem of Overfitting

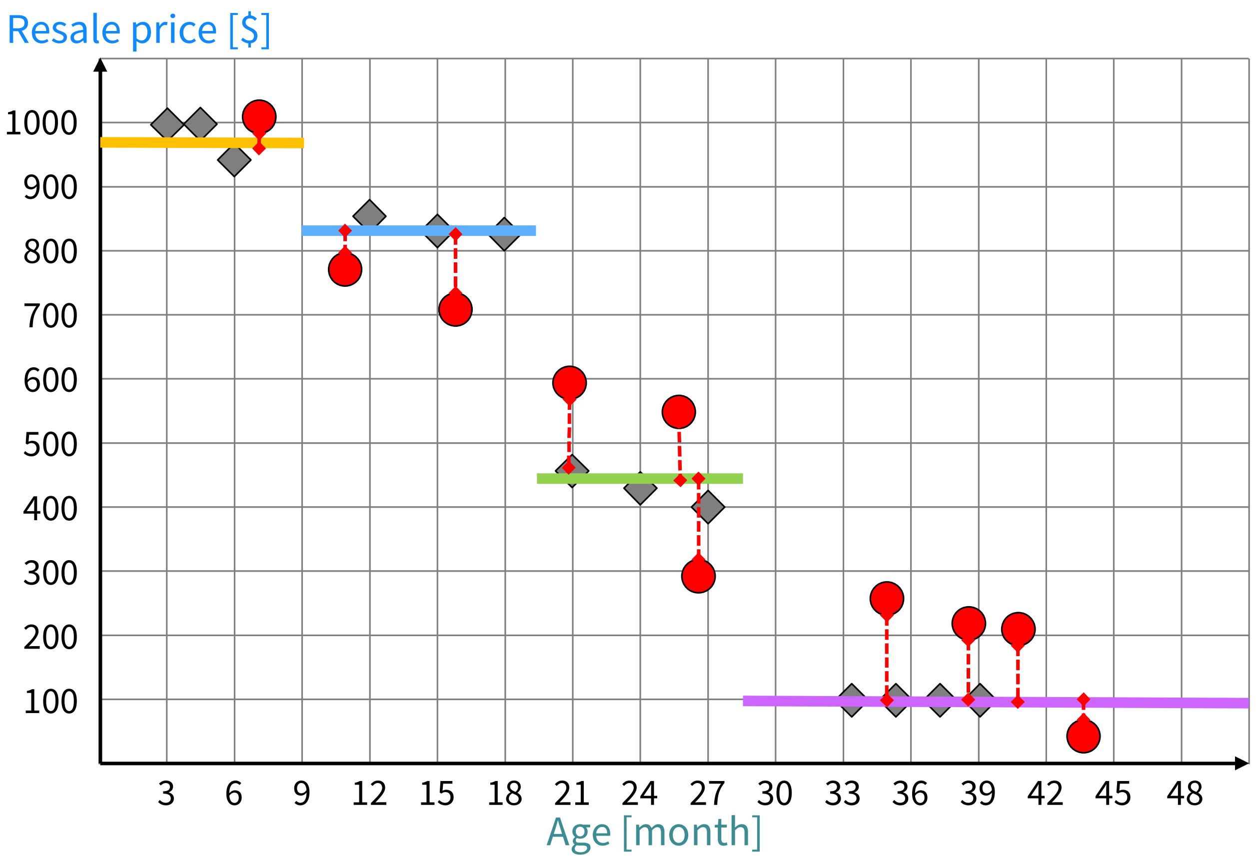
Adding some new data points to the picture

- **New test data points**
  - Similar trend as training data
  - Some differences due to sample variation
- **Tree performance**
  - Test set residuals far from zero
  - A simpler tree might have done better



Recall the Fundamental Problem of Overfitting
Adding some new data points to the picture

- **New** **test data points**
  - Similar trend as training data
  - Some differences due to sample variation
- **Tree performance**
  - Test set residuals far from zero
  - A simpler tree might have done better
- **Key take-away**
  - Some complexity (e.g., tree depth) is needed to fit the training data
  - To much complexity causes overfitting
  - Training models by minimizing SSR
    - Emphasizes model fit to the training set
    - Does not account for model complexity
    - And may thus lead to suboptimal models



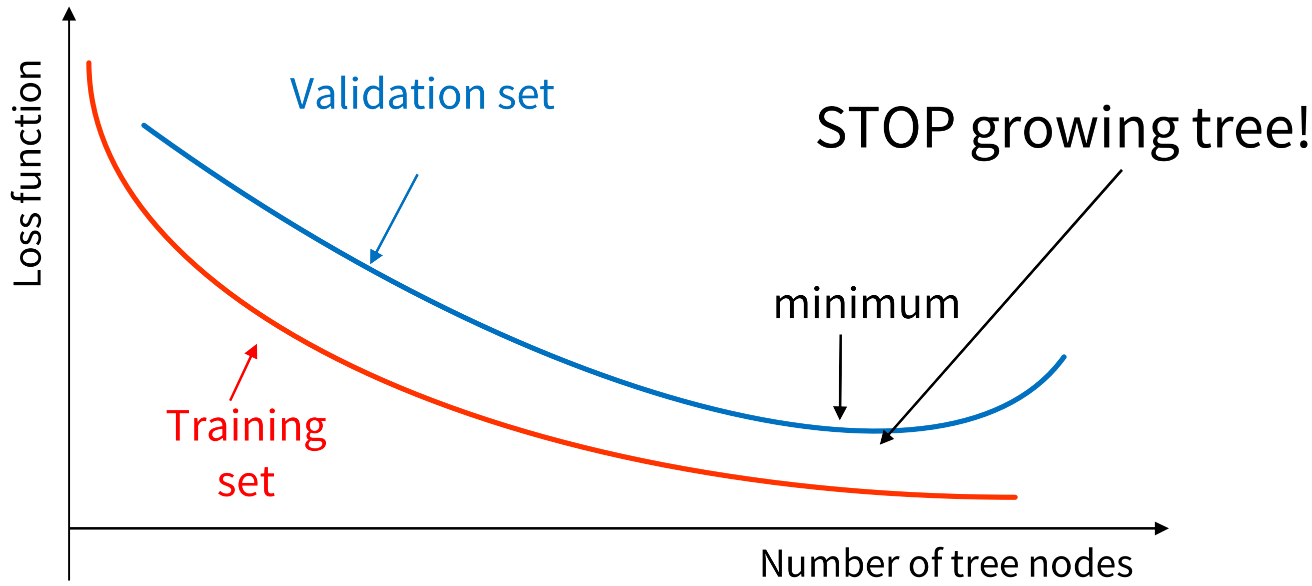
Tree Pruning to Protect Against Overfitting

Given the risk to overfit data, it is common practice to prune trees

- **Pruning is a way to reduce the complexity of the regression tree**
- **Pre-Pruning**
  - Tree learning algorithms expose meta-parameters to constrain complexity upfront
  - Most common examples include
    - Maximum depth of the tree
    - Maximum number of leaf nodes
    - Minimum number of data points to allow splitting a node
    - Minimum reduction of SSR that a split must achieve
- **Post Pruning**
  - Grow a tree to its full extend
  - Merge leaf nodes ex post to reduce complexity
  - Typically executed tracing the change in forecast accuracy on hold-out data (e.g., cross-validation)

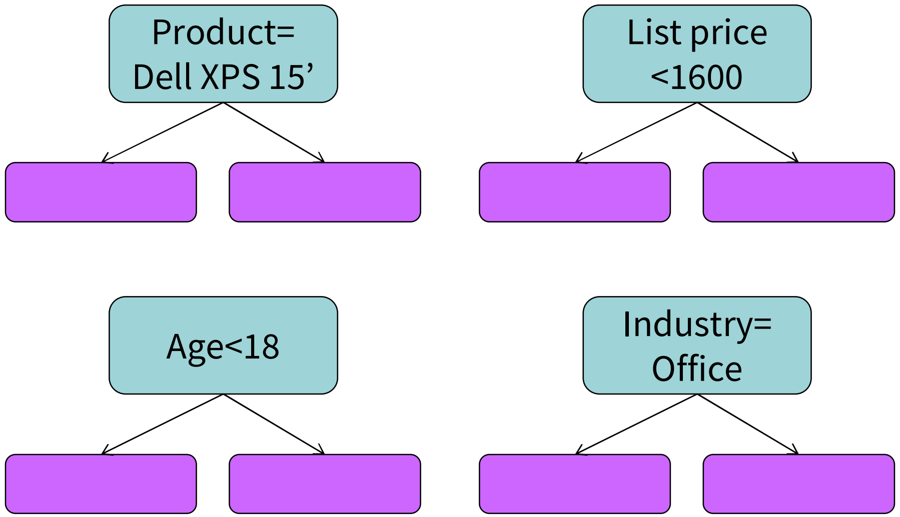
Decision Tree Pruning by Early Stopping

- **Partition data into training set (for splitting) and validation set (for stopping)**
- **Optimal tree is found where validation set loss is minimal**
  - Training loss decreases with model complexity whereas validation error increases at some point
  - This pattern follows from the bias-variance trade-off (see later)



Regression Tree Learning
Extension to multiple features

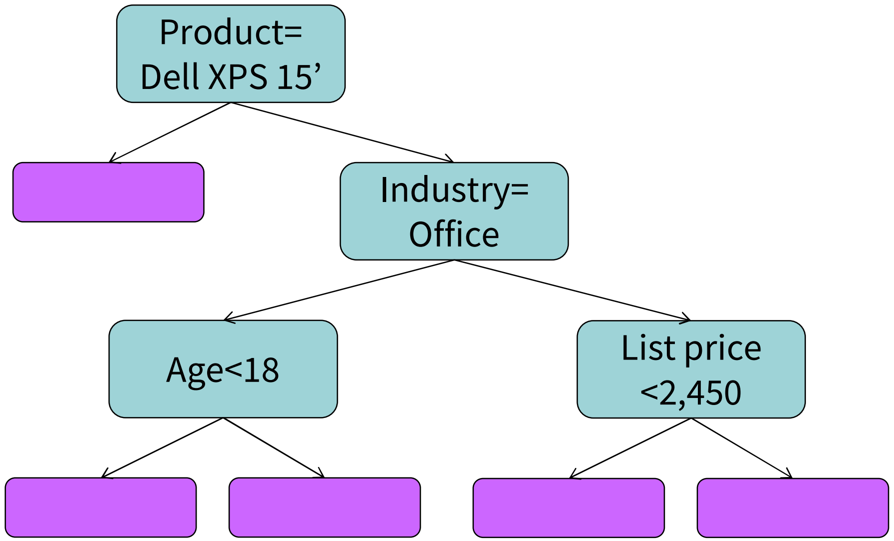
- **Real-word data sets comprise many features**
- **This does not alter our approach to find splits**
- **Find best split for each feature and then select the best overall split**



Product	List price [$]	Age [month]	Industry	…	Observed resale price [$]
Dell XPS 15’	2,500	36	Mining	…	347
Dell XPS 15’	2,500	24	Health	…	416
Dell XPS 17’	3,000	36	Manufacturing	…	538
HP Envy 17’	1,300	24	Office	…	121
HP EliteBook 850	1,900	36	Manufacturing	…	172
Lenovo Yoga 11’	799	12	Office	…	88
Lenovo Yoga 13’	1,100	12	Office	…	266
…	…	…	…	…	…

Regression Tree Learning
Extension to multiple features

- **Real-word data sets comprise many features**
- **This does not alter our approach to find splits**
- **Find best split for each feature and then select the best overall split**



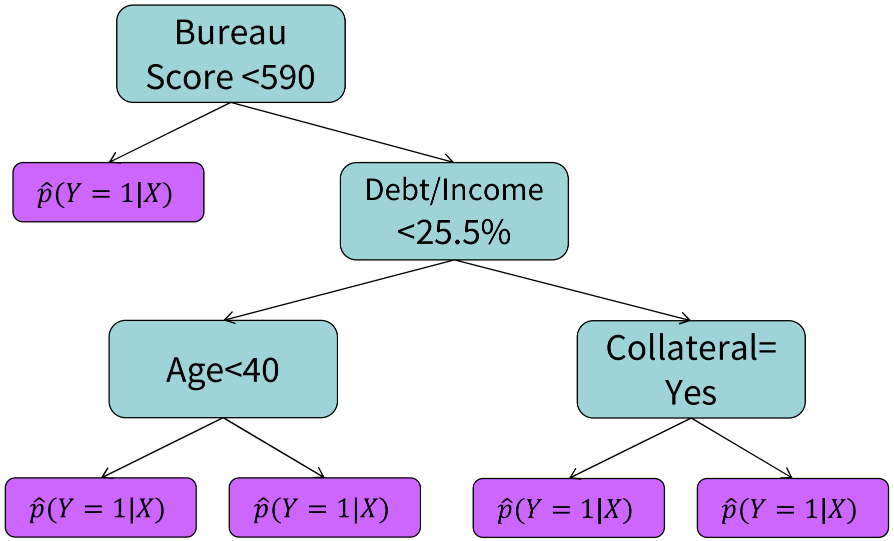
Product	List price [$]	Age [month]	Industry	…	Observed resale price [$]
Dell XPS 15'	2,500	36	Mining	…	347
Dell XPS 15'	2,500	24	Health	…	416
Dell XPS 17'	3,000	36	Manufacturing	…	538
HP Envy 17'	1,300	24	Office	…	121
HP EliteBook 850	1,900	36	Manufacturing	…	172
Lenovo Yoga 11'	799	12	Office	…	88
Lenovo Yoga 13'	1,100	12	Office	…	266
…	…	…	…	…	…

Regression Tree Learning in Pseudo-Code

```
Start from root node
For each feature
  Find best split
    Nominal variables: consider splits X=a, X=b, ...
    Numeric variables: consider splits X < a
    Assess quality of in terms of reducing SSR (or other loss function)
Compare best splits per feature across features
  SSR (best split at age) vs. SSR (best split at list price) vs. …
Select best overall split
Repeat above steps for each child node
Continue tree growing until stopping criterion or tree has
reached its full size
```

Classification Tree Learning

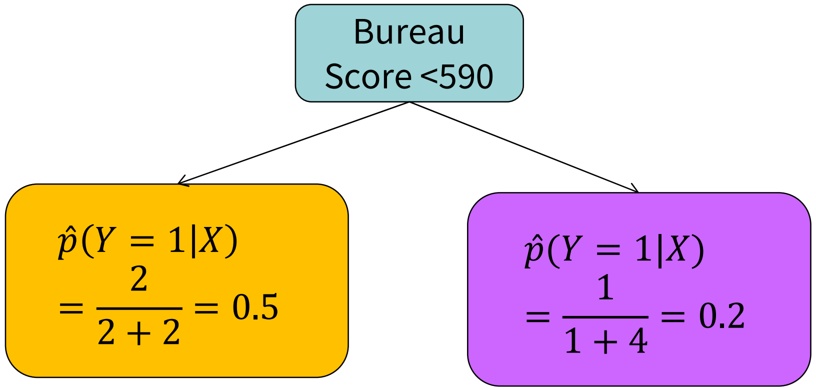
- **Leaf nodes predict class membership probability**
- **Discrete target variable necessitates a new splitting criterion**
- **Maximize purity of leaf-wise class distribution**



Bureau score	Collateral	Debt/ Income	Years at address	Age	…	Default
650	Yes	20%	2	<21	…	**No**
280	No	43%	0	21-29	…	**Yes**
750	Yes	27%	8	30-39	…	**No**
600	Yes	18%	4	40-50	…	**No**
575	No	33%	12	>50	…	**No**
715	Yes	24%	1	21-29	…	**No**
580	No	28%	6	40-50	…	**Yes**
410	Yes	29%	4	21-29	…	**No**
800	Yes	34%	10	40-50	…	**Yes**

Classification Tree Learning

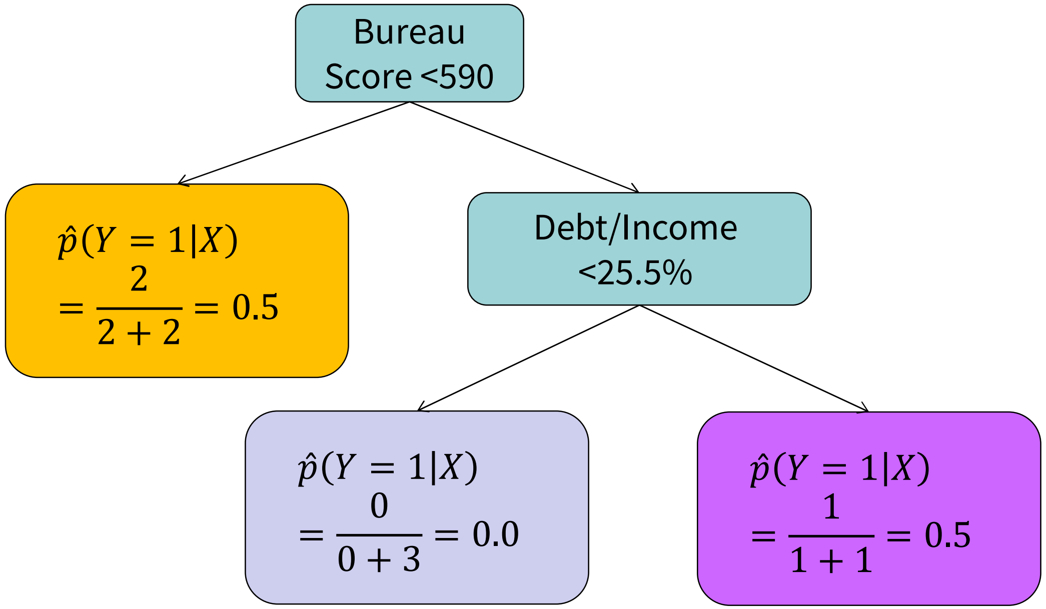
- **Leaf nodes predict class membership probability**
- **Discrete target variable necessitates a new splitting criterion**
- **Maximize purity of leaf-wise class distribution**



Bureau score	Collateral	Debt/ Income	Years at address	Age	…	Default
650	Yes	20%	2	<21	…	**No**
280	No	43%	0	21-29	…	**Yes**
750	Yes	27%	8	30-39	…	**No**
600	Yes	18%	4	40-50	…	**No**
575	No	33%	12	>50	…	**No**
715	Yes	24%	1	21-29	…	**No**
580	No	28%	6	40-50	…	**Yes**
410	Yes	29%	4	21-29	…	**No**
800	Yes	34%	10	40-50	…	**Yes**

Classification Tree Learning

- Leaf nodes predict class membership probability
- Discrete target variable necessitates a new splitting criterion
- Maximize purity of leaf-wise class distribution

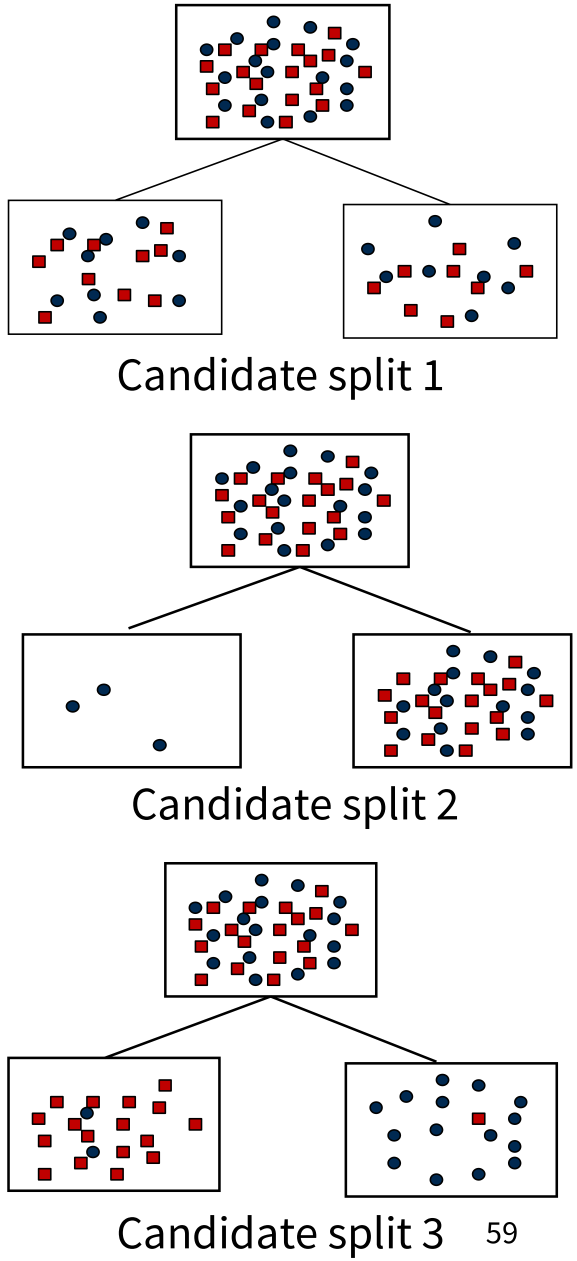


Bureau score	Collateral	Debt/ Income	Years at address	Age	…	Default
650	Yes	20%	2	<21	…	**No**
280	No	43%	0	21-29	…	**Yes**
750	Yes	27%	8	30-39	…	**No**
600	Yes	18%	4	40-50	…	**No**
575	No	33%	12	>50	…	**No**
715	Yes	24%	1	21-29	…	**No**
580	No	28%	6	40-50	…	**Yes**
410	Yes	29%	4	21-29	…	**No**
800	Yes	34%	10	40-50	…	**Yes**

Classification Tree Learning
Splitting criteria for classification

- **Indicators of node impurity**
  - A pure node is one for which all examples are of the same class
  - Maximal impurity if both classes are equally probable
  - Minimal impurity if all examples belong to the same class
- **Measurement**
  - Let $I(N)$ denote the impurity of some node N
  - Quality of a split is the weighted mean decrease in impurity
- **Information Gain (IG)**

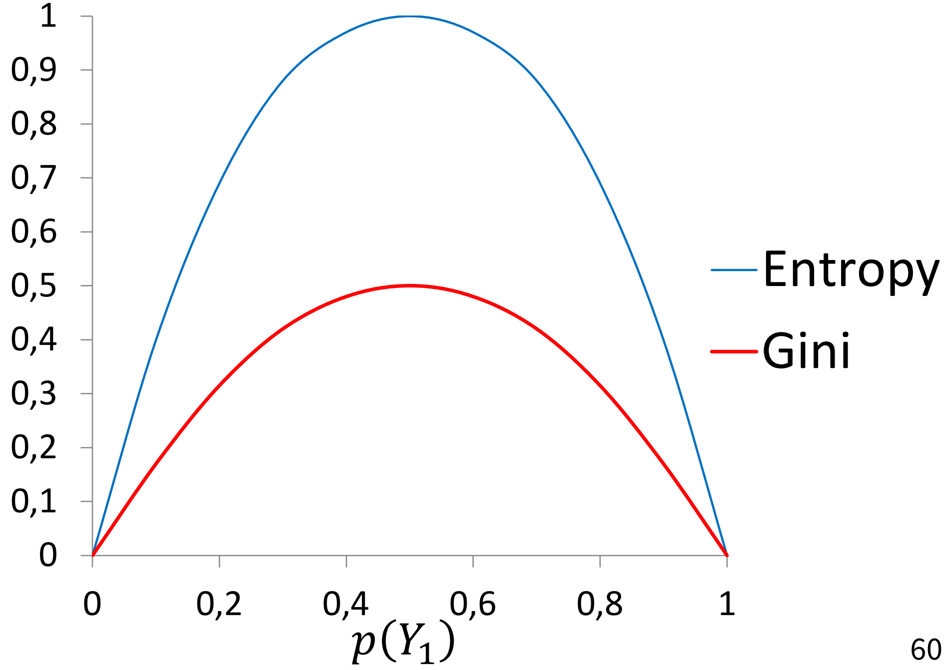
$$IG(N) = I(N) - p_{N_1} I(N_1) - p_{N_2} I(N_2)$$



Classification Tree Learning
Common choices of the impurity function to calculate gain

- **Information gain:** $IG(N) = I(N) - p_{N_1} I(N_1) - p_{N_2} I(N_2)$
- **Entropy:** $I(N) = -\sum_j p(Y_j|N) \cdot \log_2\left(p(Y_j|N)\right)$
- **Gini impurity:** $I(N) = 1 - \sum_j p(Y_j|N)^2$

Where j indexes different classes



Where p denotes probability, and Y_j the class label. Note that $p(Y_1) + p(Y_2) = 1$ for two-class problems. Also note that 0*log(0) is defined to be zero. Finally, if node N is split into two sub-nodes, N_1 and N_2, then p_{N1} and p_{N2} denote the fraction of examples in sub-node N_1 and N_2, respectively.

Summary

Summary

Learning goals

- Standard algorithms for supervised learning
- Linear models and decision trees

Findings

- Linear regression not suitable for classification
- Logit is a GLMs using the logistic link function
- Estimating logit models by maximum likelihood
- Trees recursively partition the data
- Splitting criteria measure the purity of partitions
- Pruning to avoid overfitting

What next

- Demo notebook on classification for credit risk
- How to assess predictive models
- Quality criteria, accuracy indicators, workflows

Thank you for your attention!

Stefan Lessmann

Chair of Information Systems
School of Business and Economics
Humboldt-University of Berlin, Germany

Tel. +49.30.2093.5742
Fax. +49.30.2093.5741

stefan.lessmann@hu-berlin.de
http://bit.ly/hu-wi

www.hu-berlin.de