Interpreting machine learning models

or how to turn a random forest into a white box

Ando Saabas

About me

- Senior applied scientist at Microsoft,
  - Using ML and statistics to improve call quality in Skype
  - Various projects on user engagement modelling, Skype user graph analysis, call reliability modelling, traffic shaping detection
- Previously, working on programming logics with Tarmo Uustalu

Machine learning and model interpretation

- Machine learning studies algorithms that learn from data and make predictions
- Learning algorithms are about *correlations* in the data

- In contrast, in data science and data mining, understanding *causality* is essential
- Applying domain knowledge requires understanding and interpreting models

Usefulness of model interpretation

- Often, we need to understand individual predictions a model is making.

For example a model may

- Recommend a treatment for a patient or estimate a disease to be likely. The **doctor** needs to understand the reasoning.
- Classify a user as a scammer, but the user disputes it. The **fraud analyst** needs to understand why the model made the classification.
- Predict that a video call will be graded poorly by the user. The **engineer** needs to understand why this type of call was considered problematic.

Usefulness of model interpretation cont.

- Understanding differences on a dataset level.
  - Why is a new software release receiving poorer feedback from customers when compared to the previous one?
  - Why are grain yields in one region higher than the other?

- Debugging models. A model that worked earlier is giving unexpected results on newer data.

Algorithmic transparency

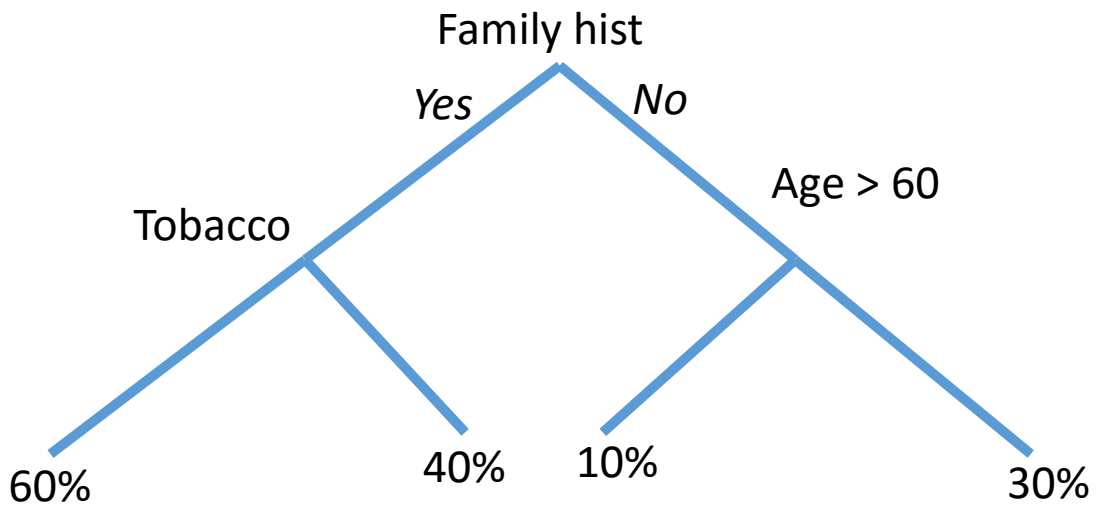
- Algorithmic transparency becoming a requirement in many fields

- French Conseil d'Etat (State Council's) recommendation in „Digital technology and fundamental rights"(2014) : *Impose to algorithm-based decisions a transparency requirement, on personal data used by the algorithm, and* ***the general reasoning it followed****.*

- Federal Trade Commission (FTC) Chair Edith Ramirez: *The agency is concerned about 'algorithmic transparency /../'* (Oct 2015).
  FTC Office of Technology Research and Investigation started in March 2015 to tackle algorithmic transparency among other issues

Interpretable models

- Traditionally, two types of (mainstream) models considered when interpretability is required
- Linear models (linear and logistic regression)
  - $Y = a + b_1 x_1 + \cdots + b_n x_n$
  - heart_disease = 0.08*tobacco + 0.043*age + 0.939*famhist + … (from *Elements of Statistical Learning*)
- Decision trees



Example: heart disease risk prediction

CORONARY DISEASE RISK PREDICTION SCORE SHEET FOR MEN BASED ON TOTAL CHOLESTEROL LEVEL

FPM Tool

Step 1

Age	
Years	Points
30-34	-1
35-39	0
40-44	1
45-49	2
50-54	3
55-59	4
60-64	5
65-69	6
70-74	7

Step 2

Total Cholesterol		
(mg/dl)	(mmol/L)	Points
<160	≤4.14	-3
160-199	4.15-5.17	0
200-239	5.18-6.21	1
240-279	6.22-7.24	2
≥280	≥7.25	3

Key	
Color	Risk
green	Very low
white	Low
yellow	Moderate
rose	High
red	Very high

Step 3

HDL - Cholesterol		
(mg/dl)	(mmol/L)	Points
<35	≤0.90	2
35-44	0.91-1.16	1
45-49	1.17-1.29	0
50-59	1.30-1.55	0
≥60	≥1.56	-2

Step 4

Blood Pressure	
Systolic	Diastolic

Step 7 (sum from steps 1-6)

Adding up the points	
Age	_____
Total Cholesterol	_____
HDL Cholesterol	_____
Blood Pressure	_____
Diabetes	_____
Smoker	_____
Point Total	_____

Step 8 (determine CHD risk from point total)

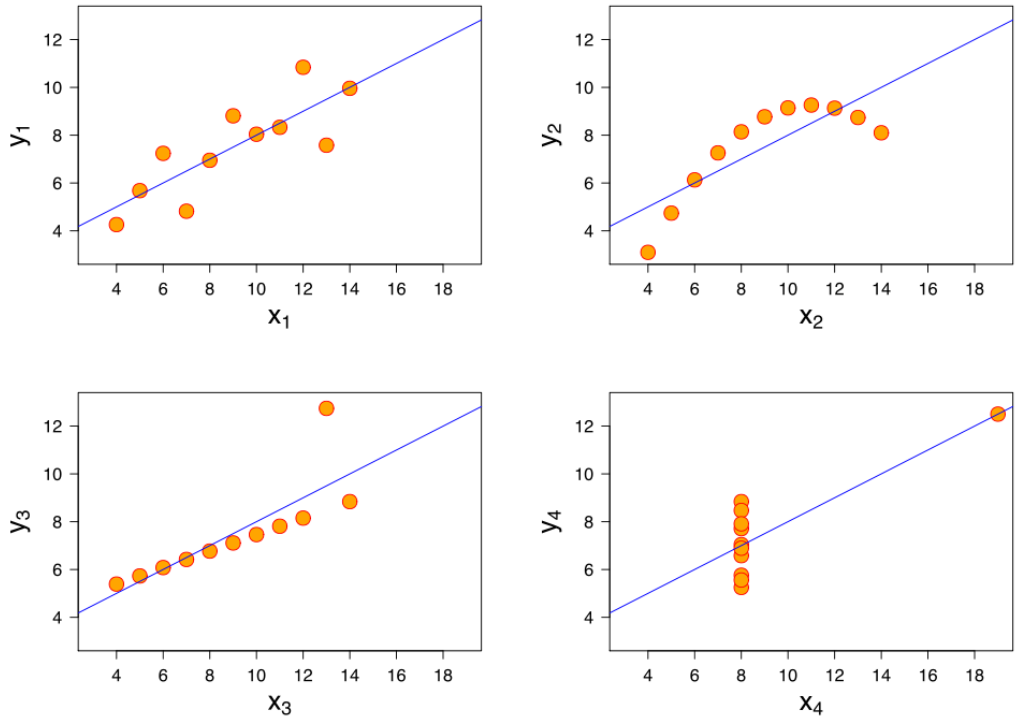
CHD Risk	
Point Total	10 Yr CHD Risk
≤-1	2%
0	3%
1	3%
2	4%
3	5%
4	7%
5	8%
6	10%
7	13%
8	16%
9	20%
10	25%
11	31%
12	37%
13	45%
≥14	≥53%

- Essentially a linear model with integer coefficients
- Easy for a doctor to follow and explain

From *National Heart, Lung and Blood Institute.*

Linear models have drawbacks

- Underlying data is often non-linear



Equal

- Mean
- Variance
- Correlation
- Linear regression model

  y = 3.00 + 0.500x

Tackling non-linearity

- Feature binning: create new variables for various intervals of input features
  - For example, instead of feature *x*, you might have
    - x_between_0_and_1
    - x_between_1_and_2
    - x_between_2_and_4 etc
  - Potentially massive increase in number of features

- Basis expansion (non-linear transformations of underlying features)

$$Y = 2x_1 + x_2 - 3x_3$$

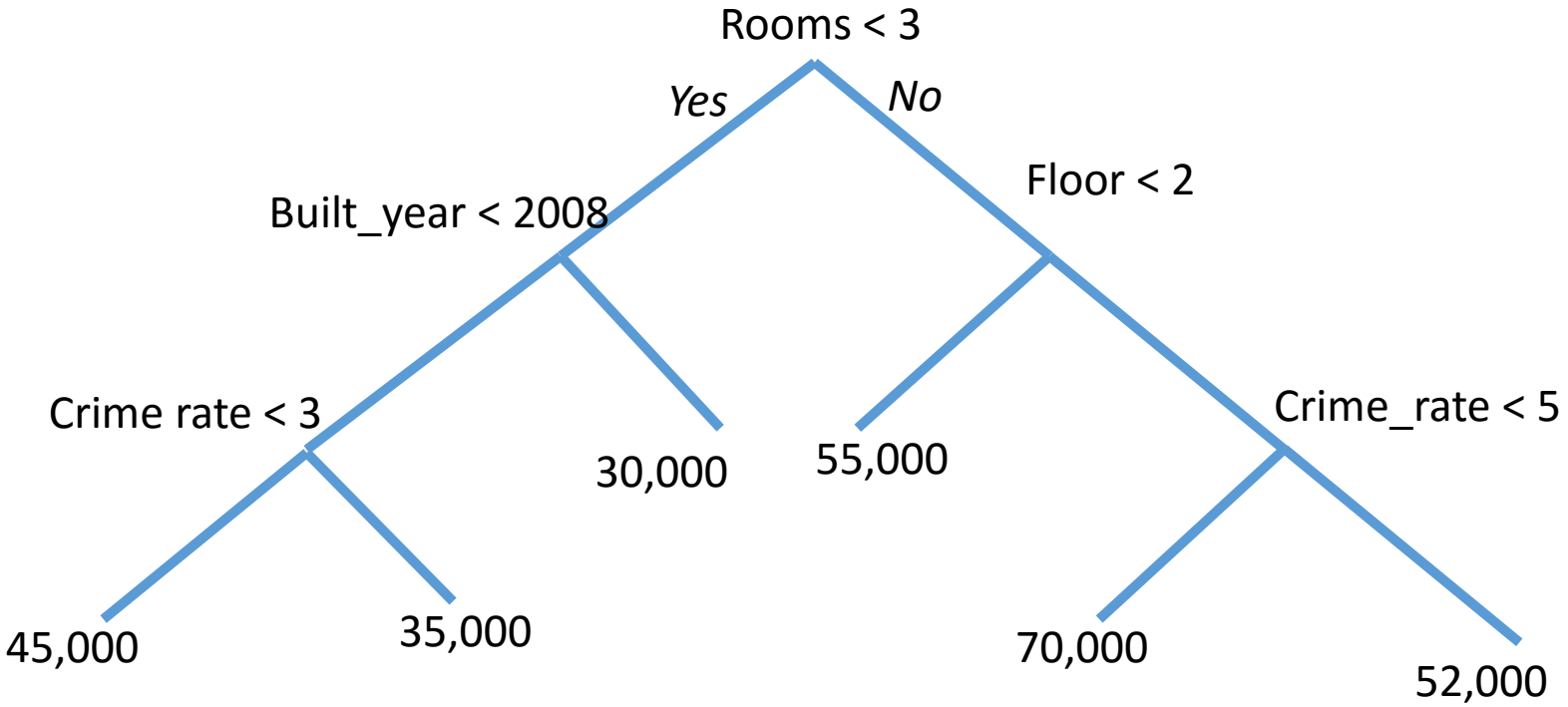
vs

$$Y = 2{x_1}^2 - 3x_1 - \log(x_2) + \sqrt{x_2}x_3 + \ \ldots$$

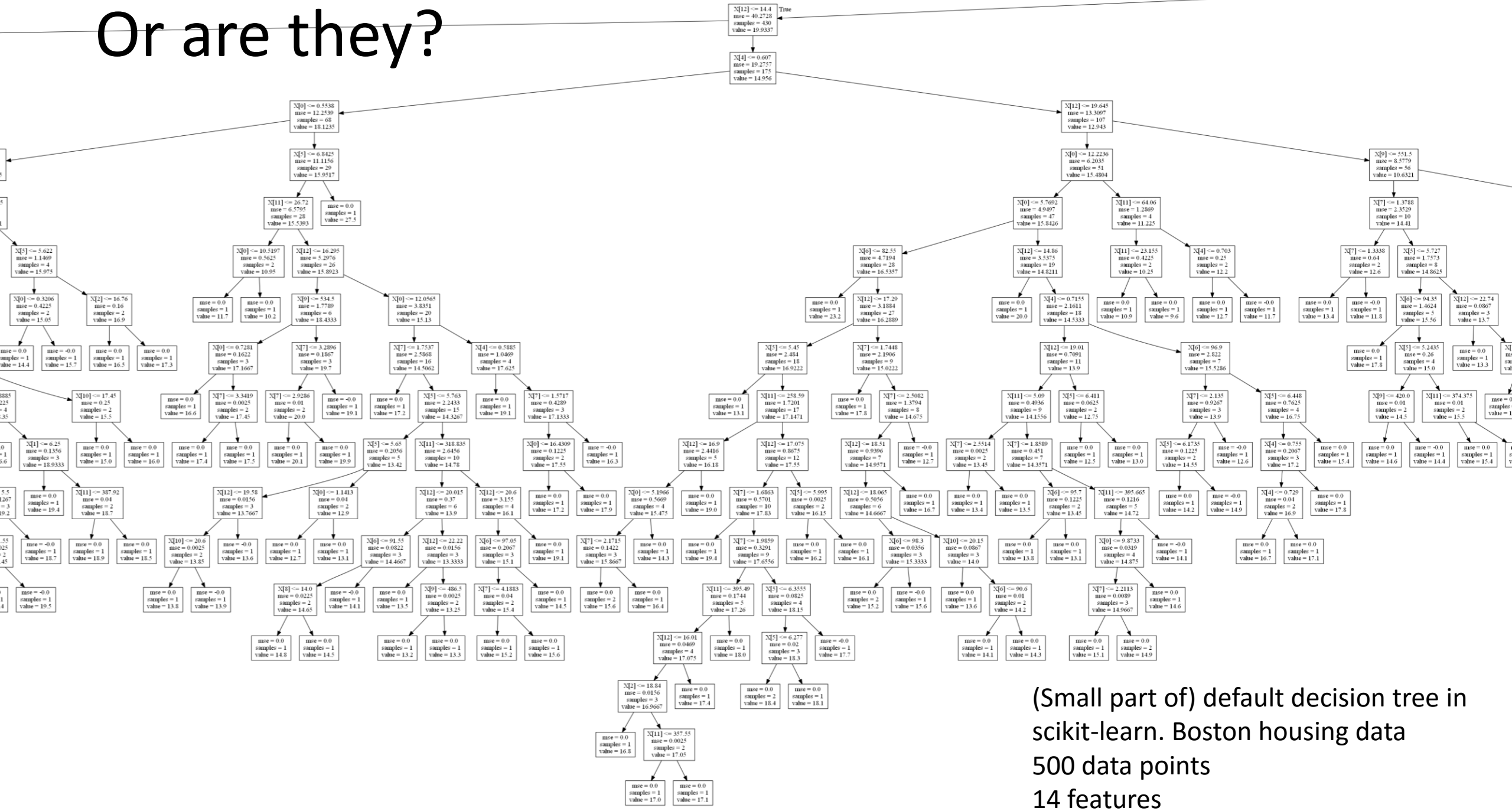
In both cases performance is traded for interpretability

Decision trees

- Decision trees can fit to non-linear data
- They work well with both categorical and continuous data, classification and regression
- Easy to understand



Or are they?



(Small part of) default decision tree in scikit-learn. Boston housing data
500 data points
14 features

Decision trees

- Decision trees are understandable only when they are (very) small
  - Tree of depth *n* has up 2^n leaves and 2^n -1 internal nodes. With depth 20, a tree can have up to 1048576 leaves
  - Previous slide had <200 nodes

- Additionally decision trees are high variance method – low generalization, tend to overfit

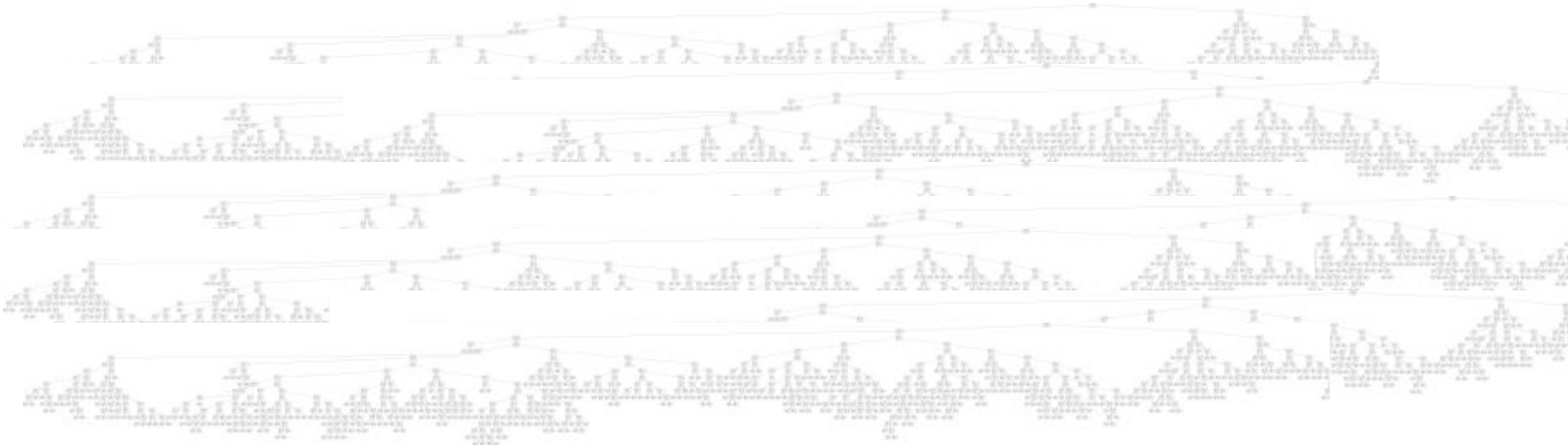
Random forests

- Can learn non-linear relationships in the data well
- Robust to outliers
- Can deal with both continuous and categorical data
- Require very little input preparation (see previous three points)
- Fast to train and test, trivially parallelizable
- High accuracy even with minimal meta-optimization

- Considered to be a **black box** that is difficult or impossible to interpret

Random forests as a black box

- Consist of a large number of decision trees (often 100s to 1000s)
- Trained on bootstrapped data (sampling with replacement)
- Using random feature selection



Random forests as a black box

- *"Black box models such as random forests can't quantify the impact of each predictor to the predictions of the complex model“*, in PRICAI 2014: Trends in Artificial Intelligence

- *"Unfortunately, the random forest algorithm is a black box when it comes to evaluating the impact of a single feature on the overall performance"*. In Advances in Natural Language Processing 2014

- *“(Random forest model) is close to a black box in the sense that it uses 810 features /../ reduction in the number of features would allow an operator to study individual decisions to have a rough idea how the global decision could have been made”.* In Advances in Data Mining: Applications and Theoretical Aspects: 2014

Understanding the model vs the predictions

- Keep in mind, we want to understand why a particular decision was made. Not necessarily every detail of the full model

- As an analogy, we don't need to understand how a brain works to understand why a person made a particular a decision: simple explanation can be sufficient

- Ultimately, as ML models get more complex and powerful, hoping to understand the models themselves is doomed to failure
- **We should strive to make models explain their decisions**

Turning the black box into a white box

- In fact, random forest predictions **can** be explained and interpreted, by decomposing predictions into mathematically *exact* feature contributions

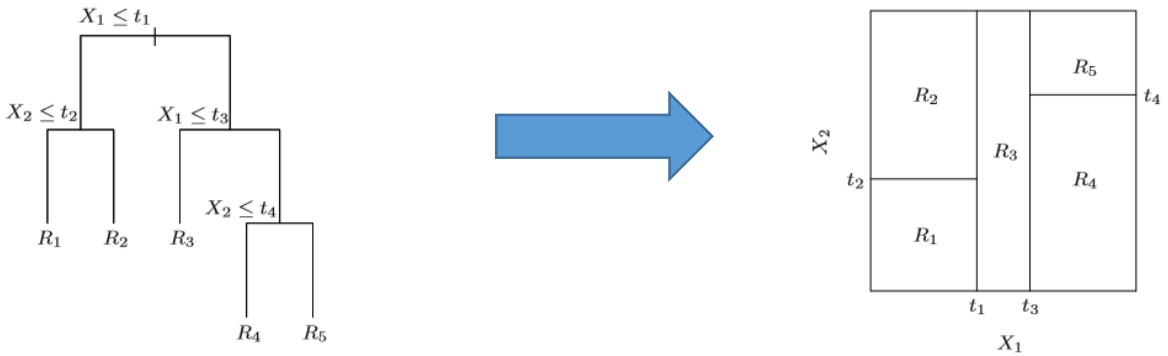
- Independently of the
  - number of features
  - number of trees
  - depth of the trees

Revisiting decision trees

- Classical definition (from *Elements of statistical learning*)

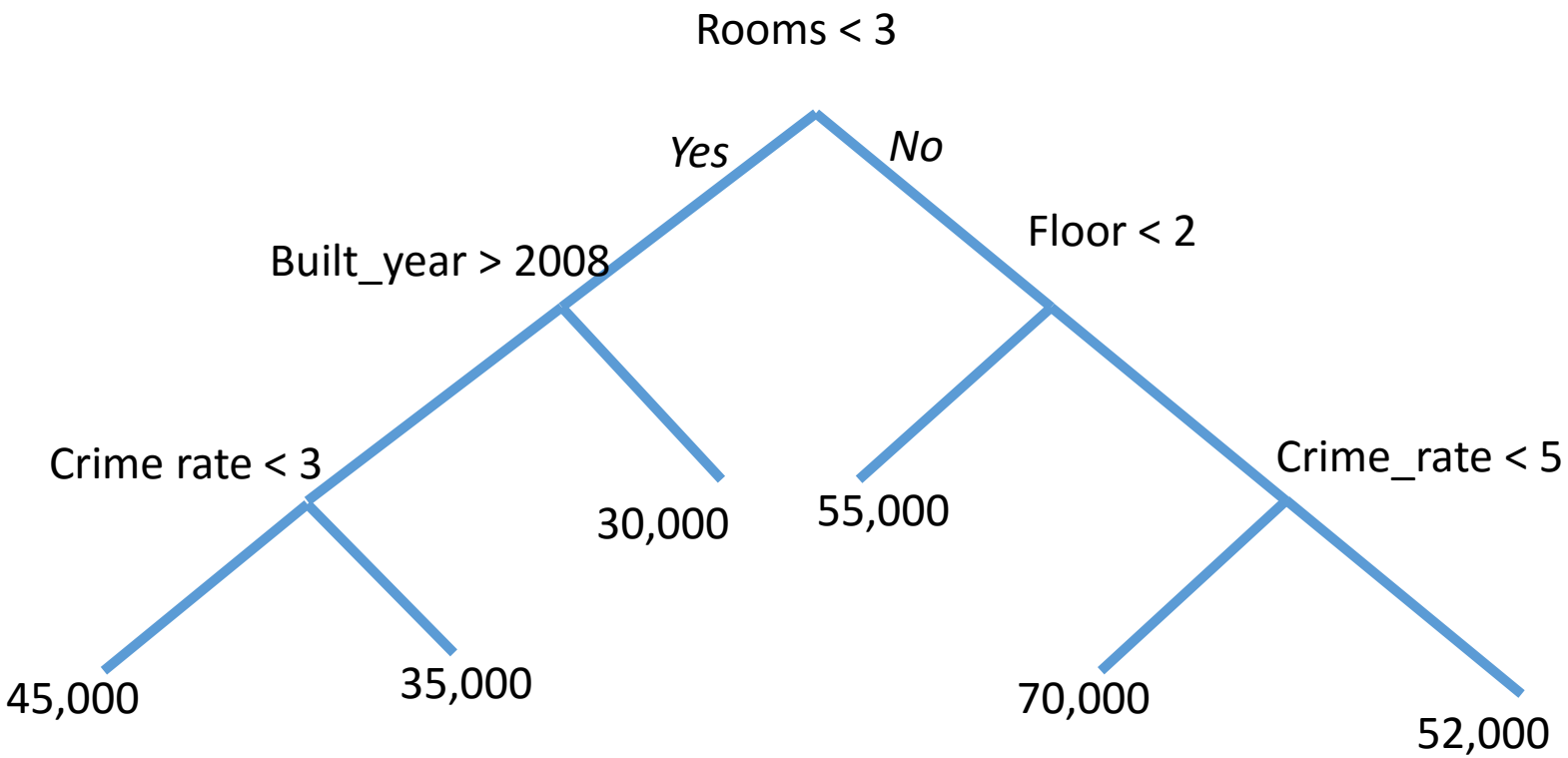
$$dt(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$$

- Tree divides the feature space into M regions R_m (one for each leaf)

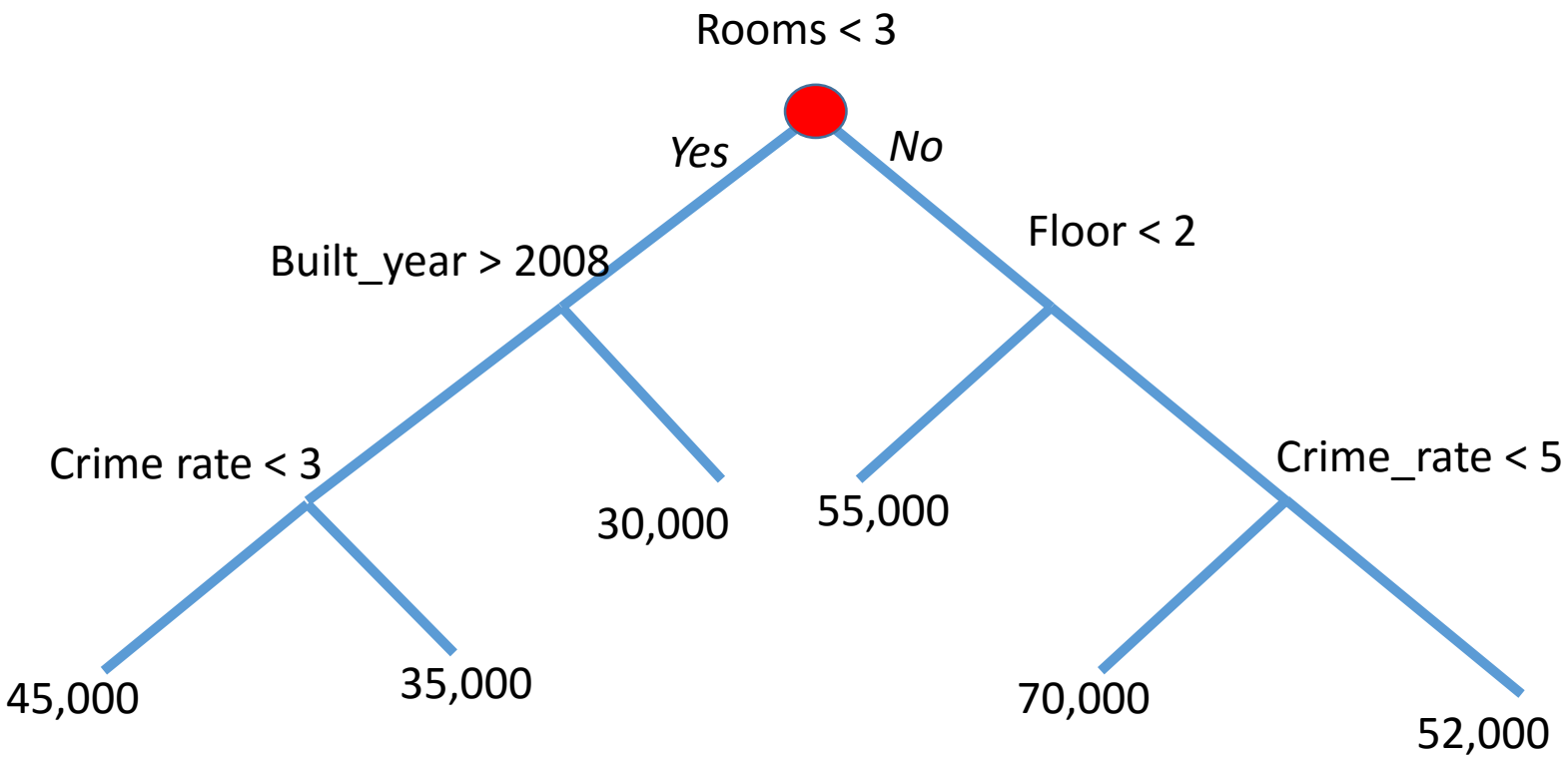


- Prediction for feature vector *x* is the constant c_m associated with region R_m the vector *x* belongs to

Example decision tree – predicting apartment prices

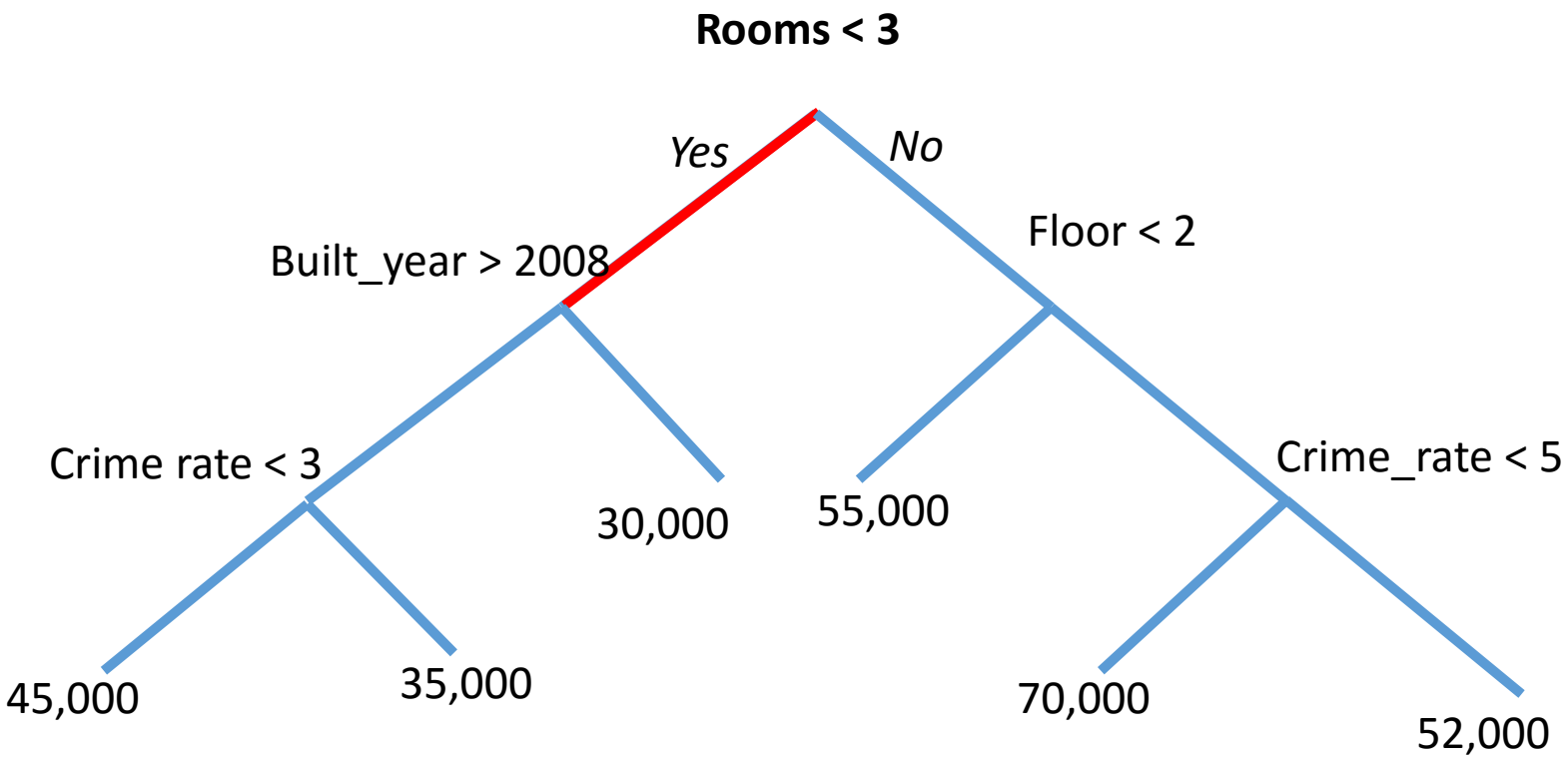


Estimating apartment prices



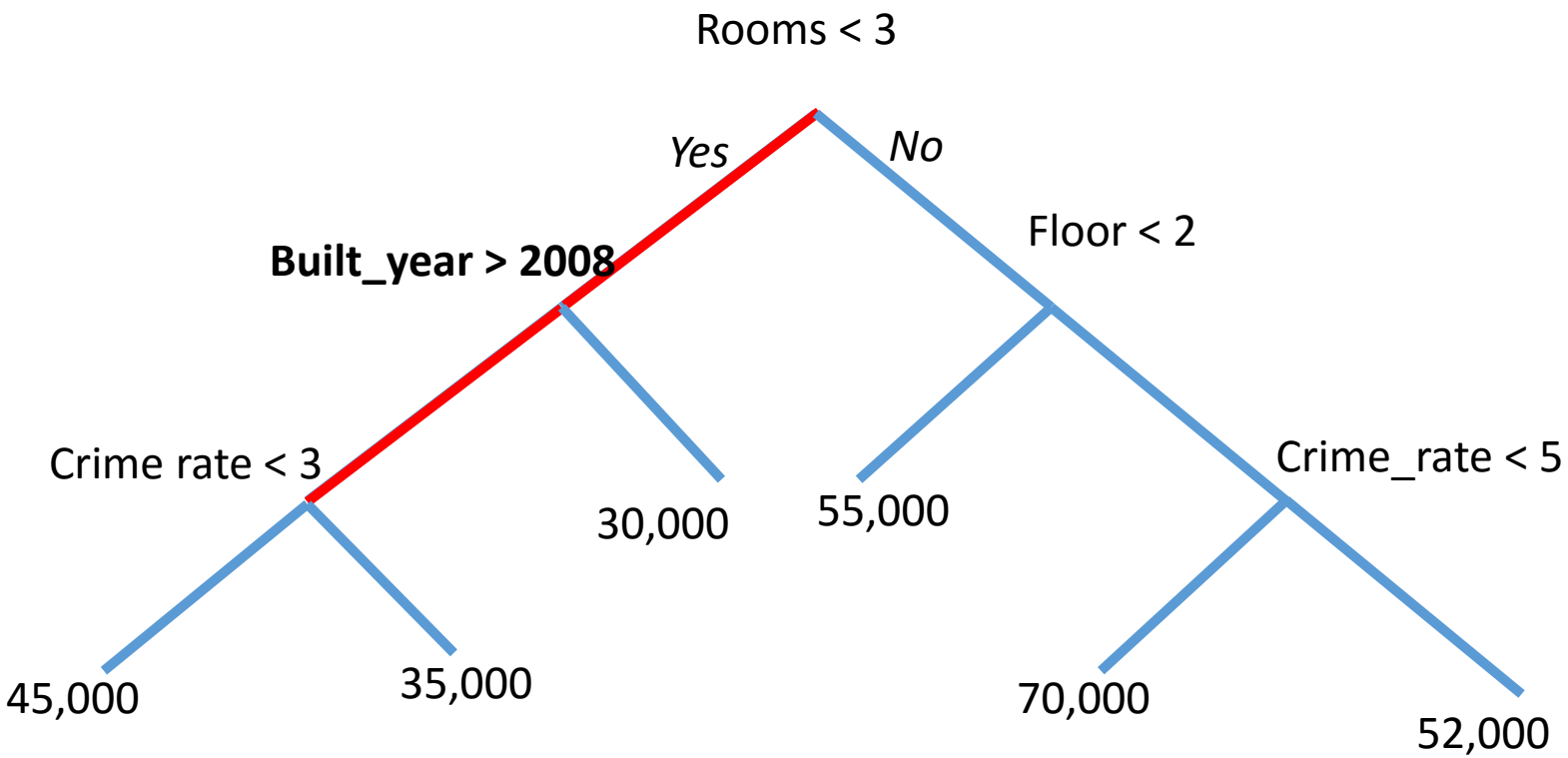
Assume an apartment **[2 rooms; Built in 2010; Neighborhood crime rate: 5]**
We walk the tree to obtain the price

Estimating apartment prices



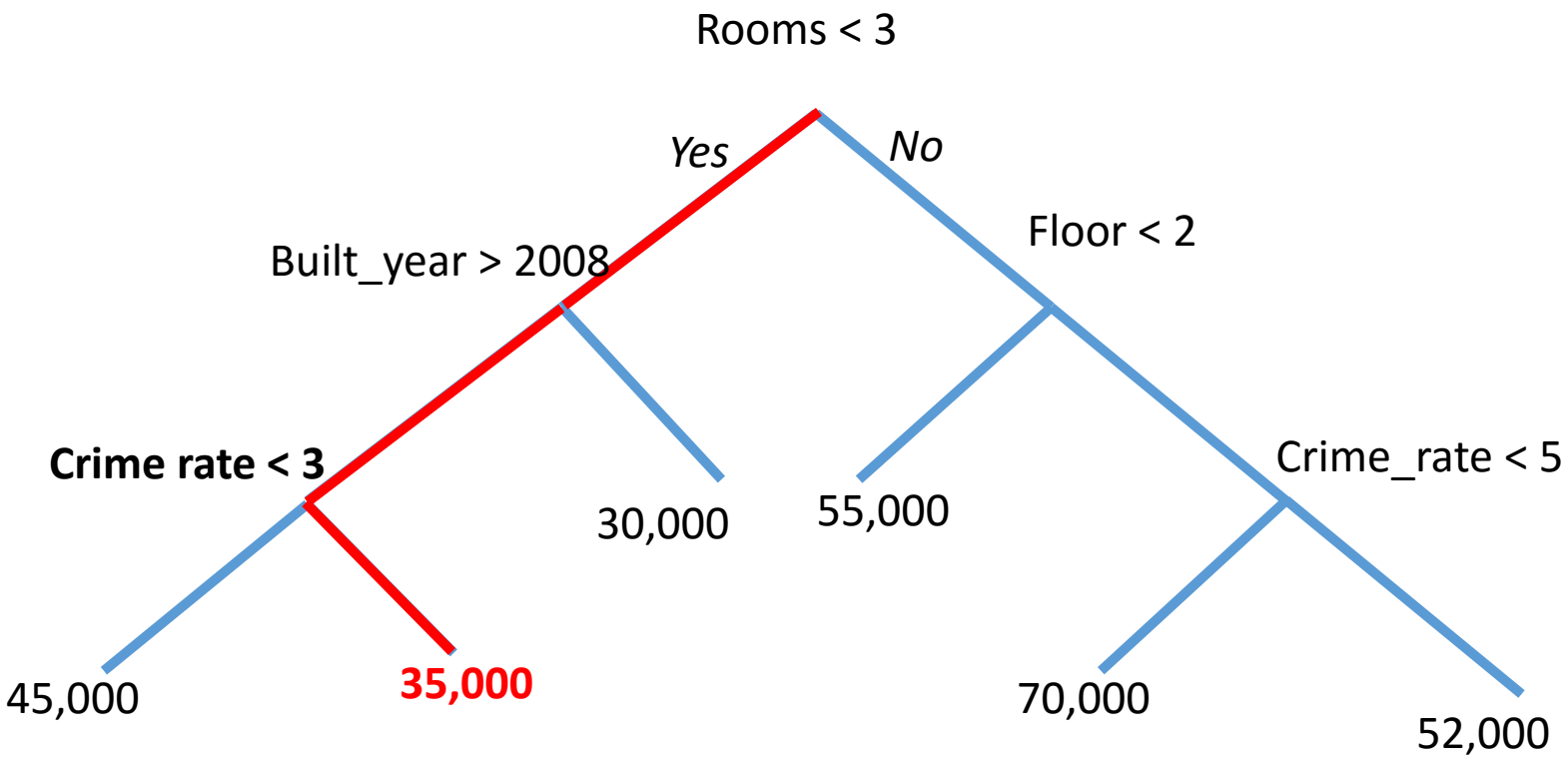
[2 rooms; Built in 2010; Neighborhood crime rate: 5**]**

Estimating apartment prices



[2 rooms; **Built in 2010;** Neighborhood crime rate: 5]

Estimating apartment prices



[2 rooms; Built in 2010; **Neighborhood crime rate: 5**]

Prediction: 35,000

Path taken: Rooms < 3, Built_year > 2008, Crime_rate < 3

Operational view

- Classical definition ignores the operational aspect of the tree.

- There is a **decision path** through the tree
- All nodes (not just the leaves) have a value associated with them

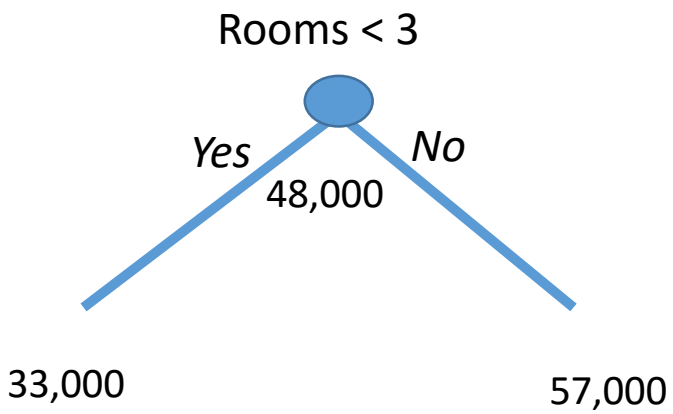
Internal values

- All internal nodes have a value assocated with them

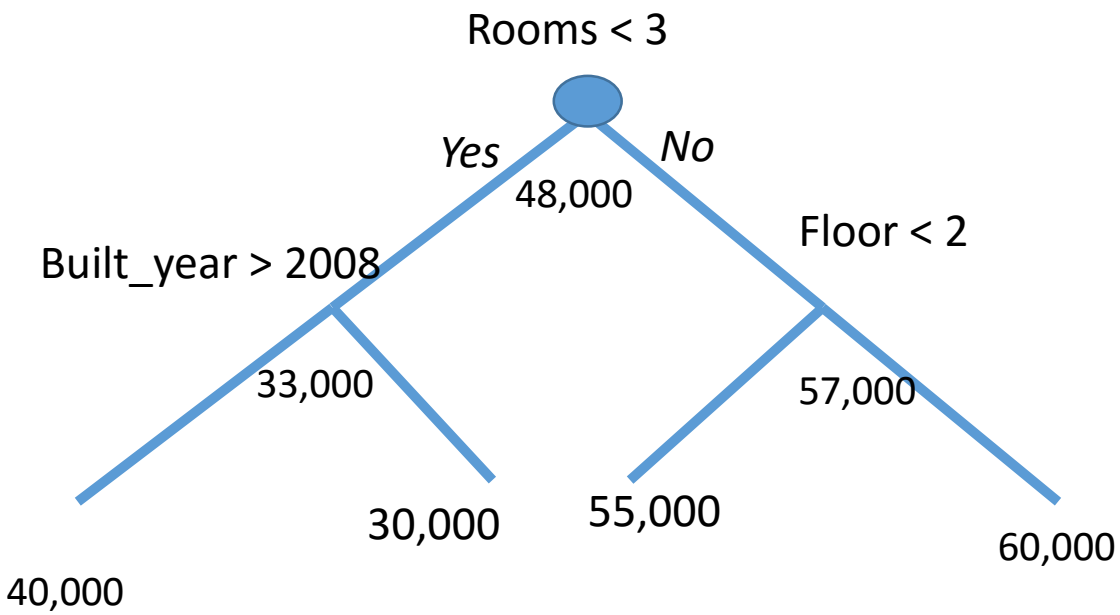
48,000

- At depth 0, prediction would simply be the dataset mean (assuming we want to minimize squared loss)
- When training the tree, we keep expanding it, obtaining new values

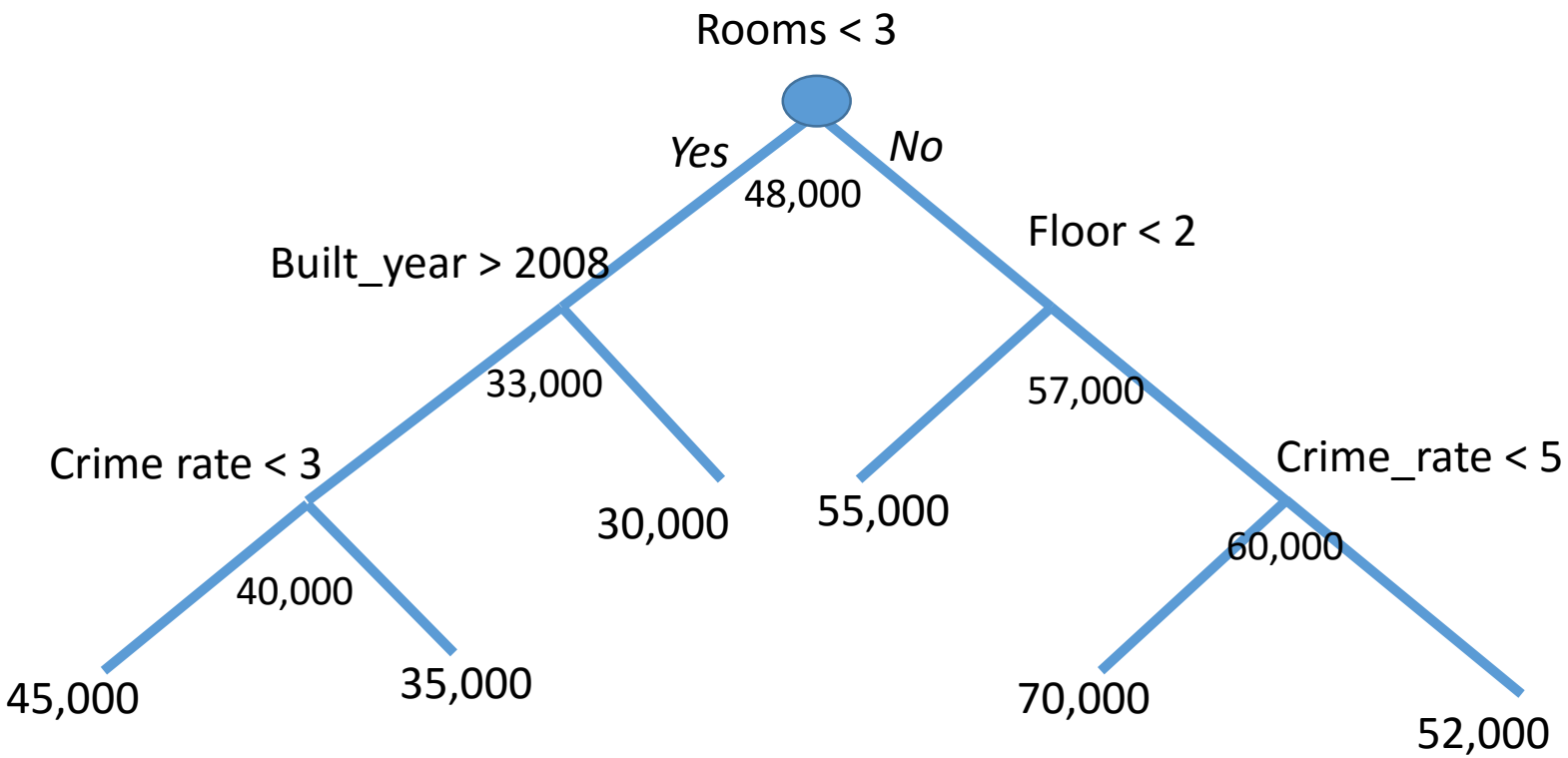
Internal values



Internal values



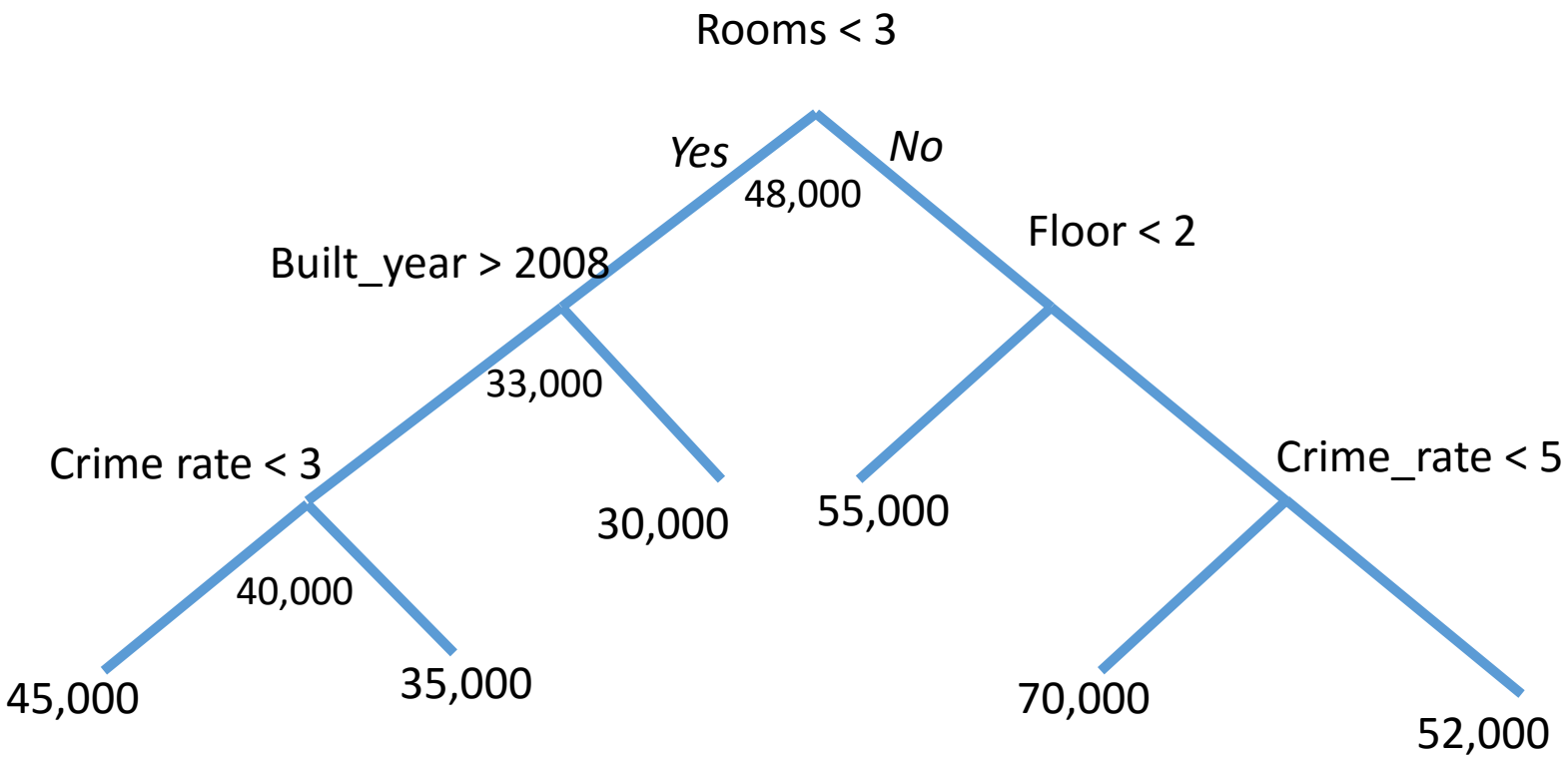
Internal values



Operational view

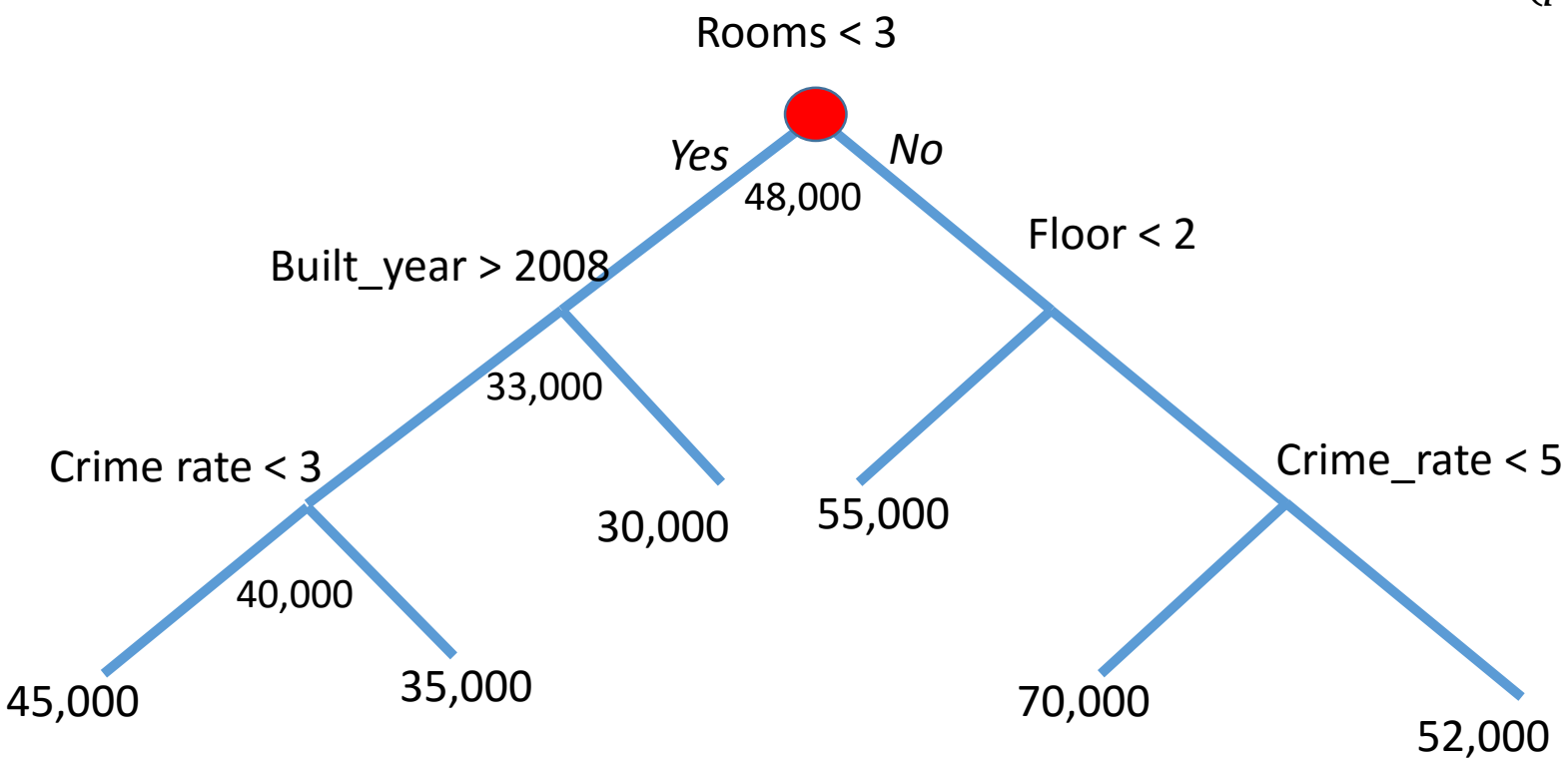
- All nodes (not just the leaves) have a value associated with them
- Each decision along the path contributes something to the final outcome
- A feature is associated with every decision
- We can compute the final outcome in terms of feature contributions

Estimating apartment prices revisited



Estimating apartment prices

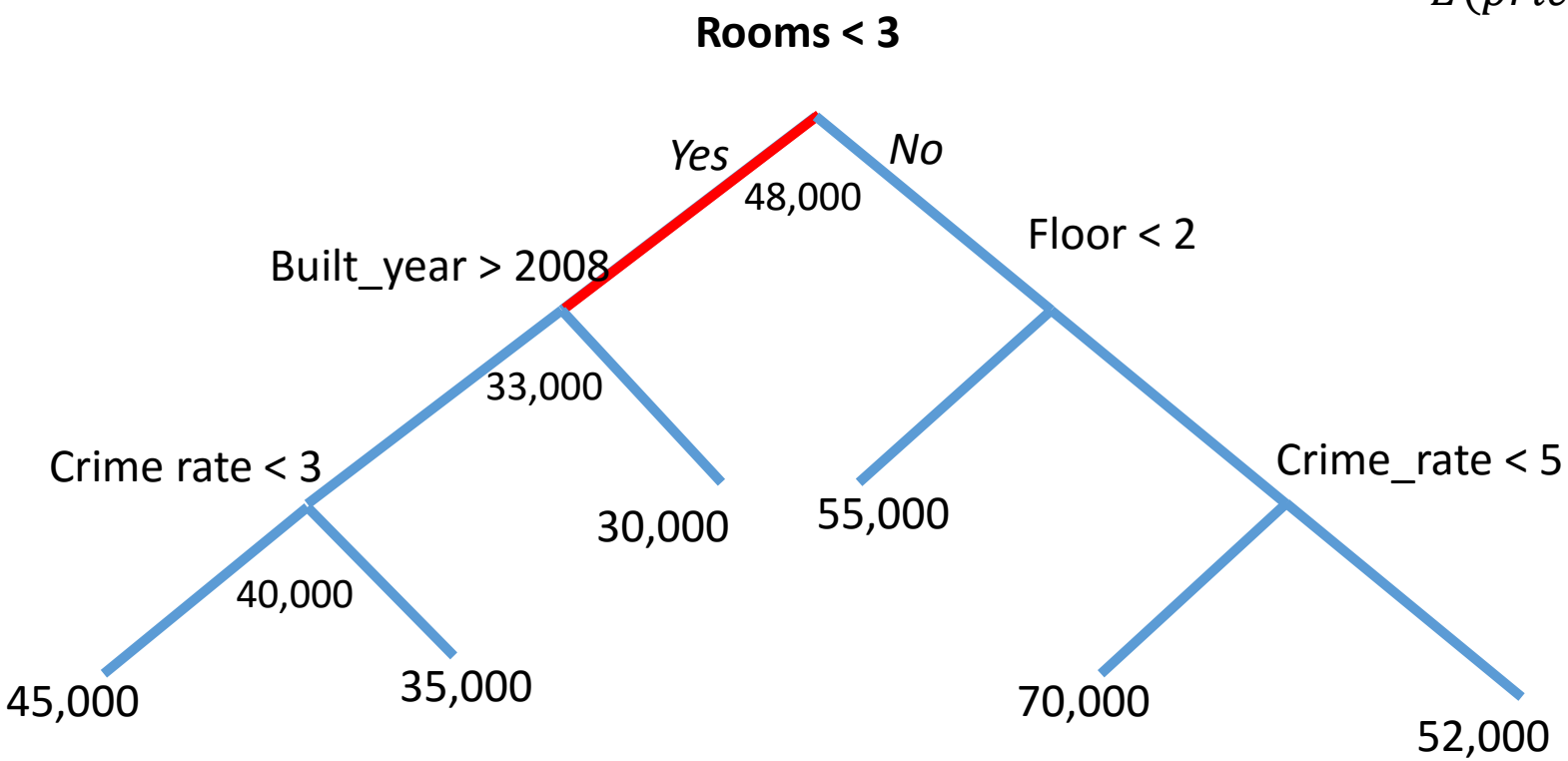
$E(price) = 48{,}000$



Price = 48,000

Estimating apartment prices

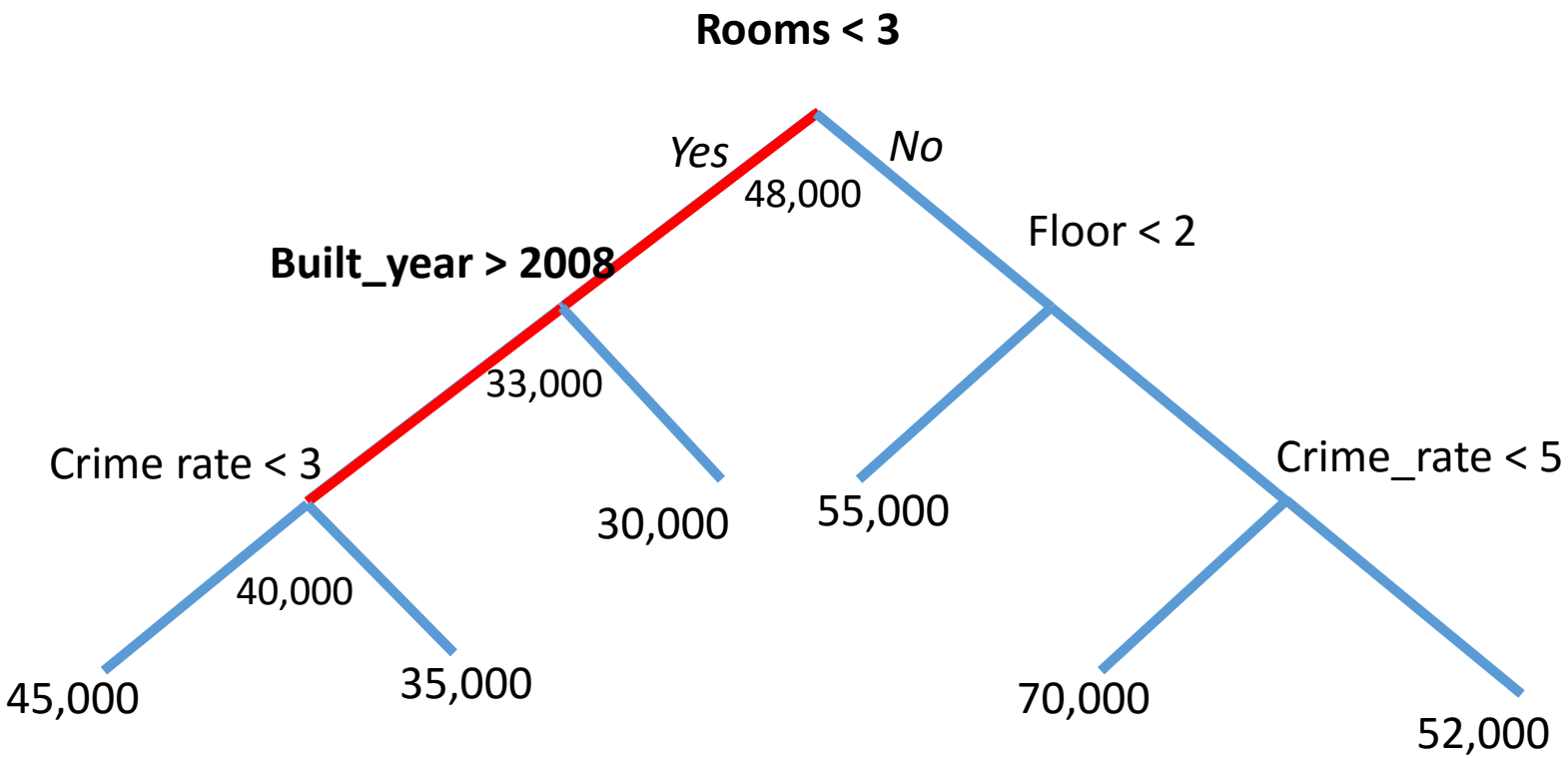
$E(price|rooms < 3) = 33{,}000$



Price = 48,000 – 15,000(Rooms)

[2 rooms]

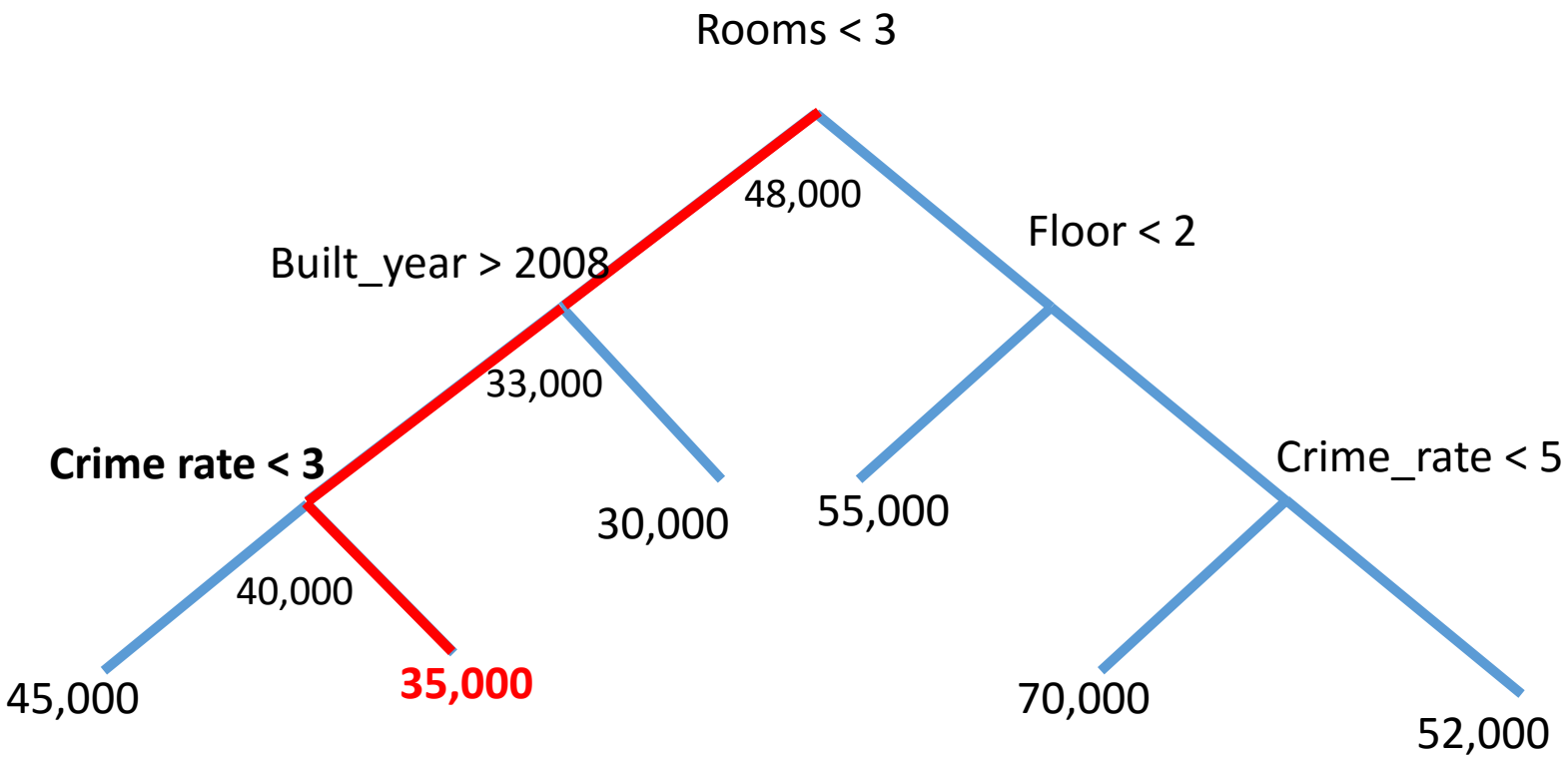
Estimating apartment prices



Price = 48,000 – 15,000(Rooms) + 7,000(Built_year)

[Built 2010]

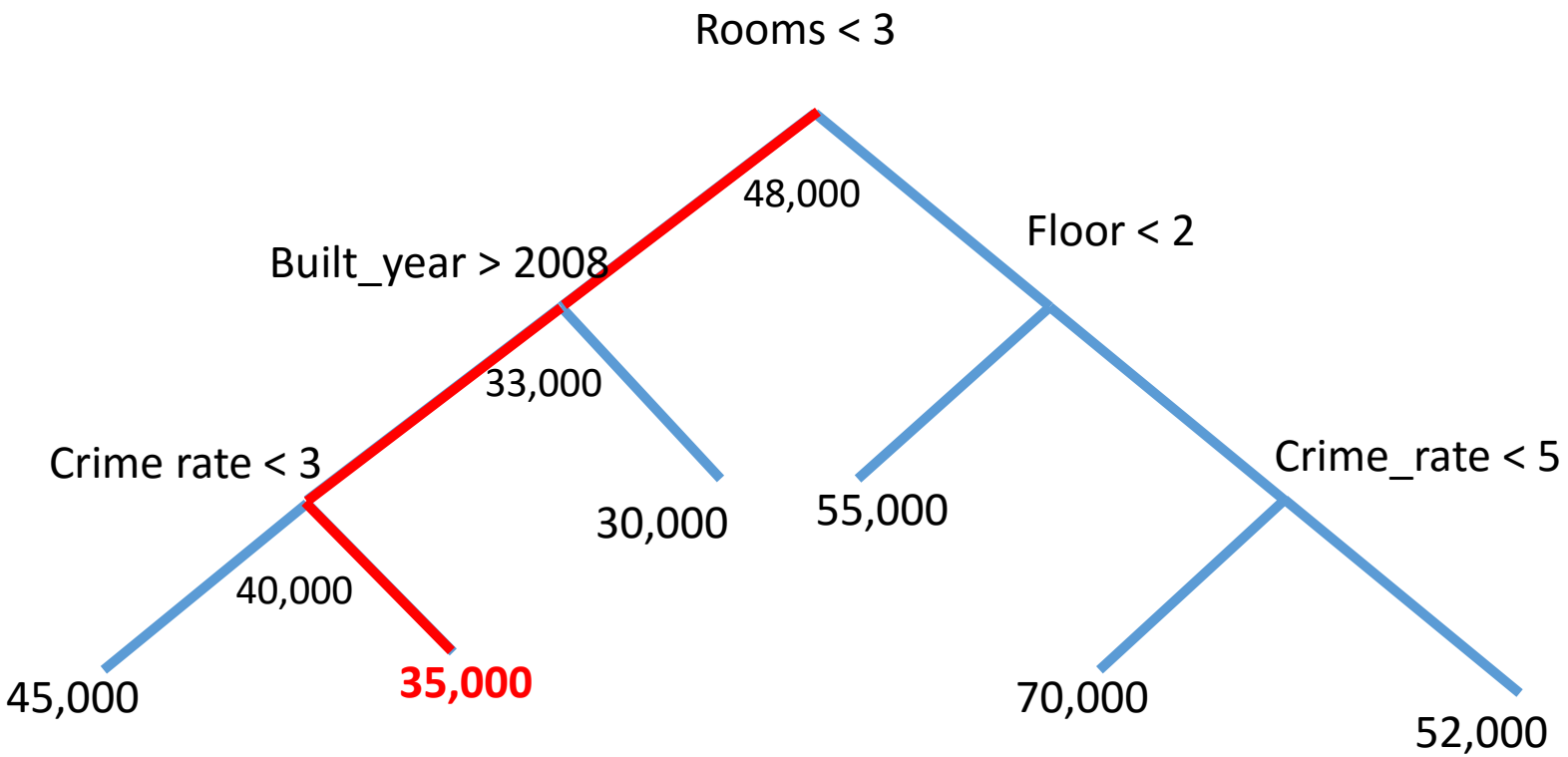
Estimating apartment prices



Price = 48,000 – 15,000(Rooms) + 7,000(Built_year) – 5,000(Crime_rate)

[Crime rate 5]

Estimating apartment prices



Price = 48,000 – 15,000(Rooms) + 7,000(Built_year) – 5,000(Crime_rate) = 35,000

Decomposition for decision trees

- We can define the the decision tree in term of the bias and contribution from each feature.

$$\text{dt}(x) = \sum_{m=1}^{M} c_m I(x \in R_m) \quad \Rightarrow \quad \text{dt}(x) = bias + \sum_{i=1}^{N} contr(i, x))$$

- Similar to linear regression (prediction = bias + feature_1contribution + … + feature_ncontribution), but on a prediction level, not model level
- Does not depend on the size of the tree or number of features
- Works equally well for
  - Regression and classification trees
  - Multivalued and multilabel data

Deeper inspections

- We can have more fine grained definition in addition to pure feature contributions
  - Separate negative and positive contributions
  - Contribution from decisions (floor = 1 → -15000)
  - Contribution from interactions (floor == 1 & has_terrace → 3000)
  - etc

- Number of features typically not a concern because of the long tail. In practice, top 10 features contribute the vast majority of the overall deviation from the mean

From decision trees to random forests

- Prediction of a random forest is the average of the predictions of individual trees

$$\mathrm{RF}(x) = \frac{1}{J}\sum_{j=1}^{J} dt_j(x)$$

- From distributivity of multiplication and associativity of addition, random forest prediction can be defined as the average of bias term of individual trees + sum of averages of each feature contribution from individual trees

$$\mathrm{RF}(x) = \frac{1}{J}\sum_{j=1}^{J} bias_j(x) + (\frac{1}{J}\sum_{j=1}^{J} contr_j(1,x) + \cdots + \frac{1}{J}\sum_{j=1}^{J} contr_j(n,x))$$

- prediction = bias + feature_1contribution + … + feature_ncontribution

Random forest interpretation in Scikit-Learn

- Path walking in general no supported by ML libraries
- Scikit-Learn: one of the most popular Python (and overall) ML libraries
- Patch since 0.17 (released Nov 8 2015) to include values/predictions at each node: allows walking the tree paths and collecting values along the way
- **Treeintepreter** library for decomposing the predictions
  - https://github.com/andosa/treeinterpreter
  - `pip install treeinterpreter`
  - Supports both decision tree and random forest classes, regression and classification

Using `treeinterpreter`

- Decomposing predictions is a one-liner

```
from treeinterpreter import treeinterpreter as ti
rf = RandomForestRegressor()
rf.fit(trainX,trainY)

prediction, bias, contributions = ti.predict(rf, testX)
#instead of prediction = rf.predict(testX)

#prediction == bias + contributions
assert(numpy.allclose(prediction, bias + np.sum(contributions, axis=1)))
```

Decomposing a prediction – boston housing data

```
prediction, bias, contributions = ti.predict(rf, boston.data)

>> prediction[0]
30.69
>> bias[0]
25.588

>> sorted(zip(contributions[0],  boston.feature_names),
>>             key=lambda x: -abs(x[0]))

[(4.338716569719558, 'RM'),
 (-1.0771391053864874, 'TAX'),
 (1.020711612907321 3, 'LSTAT'),
 (0.3889077481279 7702, 'AGE'),
 (0.3838148148153 9, 'ZN'),
 (-0.1039722222222205, 'CRIM'),
 (-0.09152069716775 6987, 'NOX')
...
```

Caveats

- The method assumes that fetures are actually interpretable in the first place.
  - This does not always hold: e.g. pixels in image classification
  - Can be overcome via postprocessing

- In presence of strong correlations in the input data, true causal features can be buried under non-causal but correlated features. Domain knowledge and feature tuning required in this case

Conclusions

- Model interpretation can be very beneficial for ML and data science practitioners in many tasks
- No need to understand the full model in many/most cases: explanation of decisions sufficient
- Random forests can be turned from black box into a white box
  - Each prediction can decomposed into bias and feature contribution terms
- Python library available for scikit-learn at https://github.com/andosa/treeinterpreter
  or `pip install treeinterpreter`