Introduction to Data Science

Lesson 7
Feature Selection, Regularization and Sparsity

Marija Stankova Medarovska, PhD
marija.s.medarovska@uacs.edu.mk

High-dimensional Data

Curse of dimensionality

Problems arise when analyzing data in high-dimensional spaces

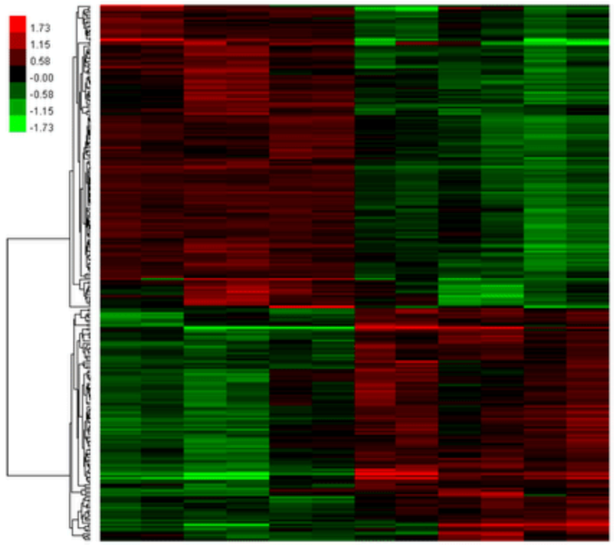
What is a “high” dimension?

In a machine learning context, any time $p > n$, we are in a high- dimensional setting. Problems with high dimensionality can be experienced even when $n > p$, if p is large. The amount of data you need depends on the complexity of your task and algorithm.

p = dimension of input, n = number of samples

Examples of high dimensional spaces

Image data



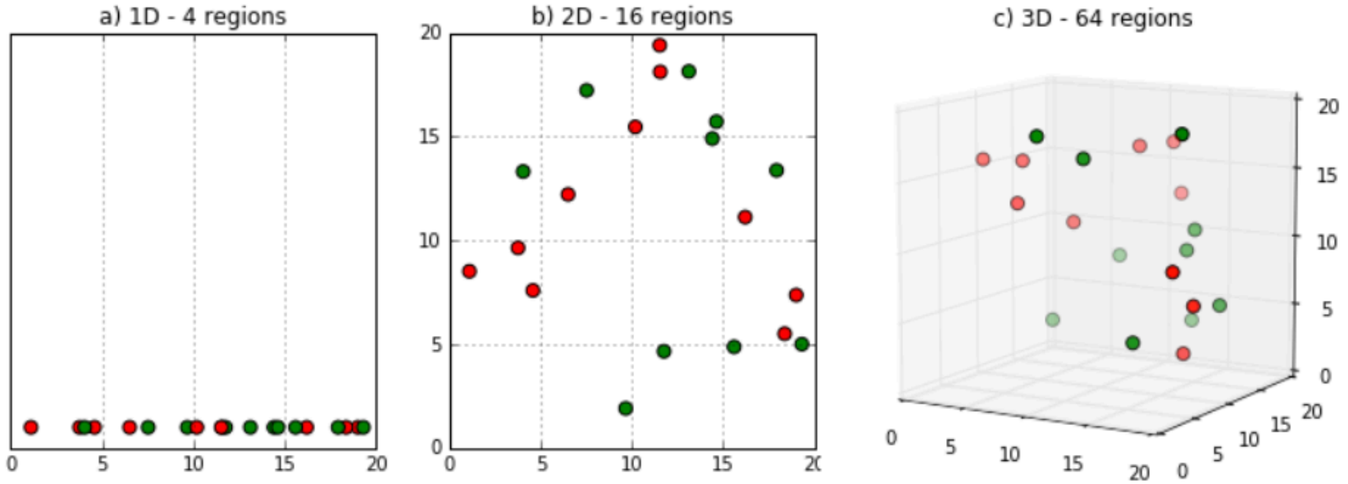
Gene expression data

Features	I	go	to	school	They	We
Document ID						
1	1	1	1	1	0	0
2	0	1	1	1	1	0
3	0	1	1	1	0	1

Text data

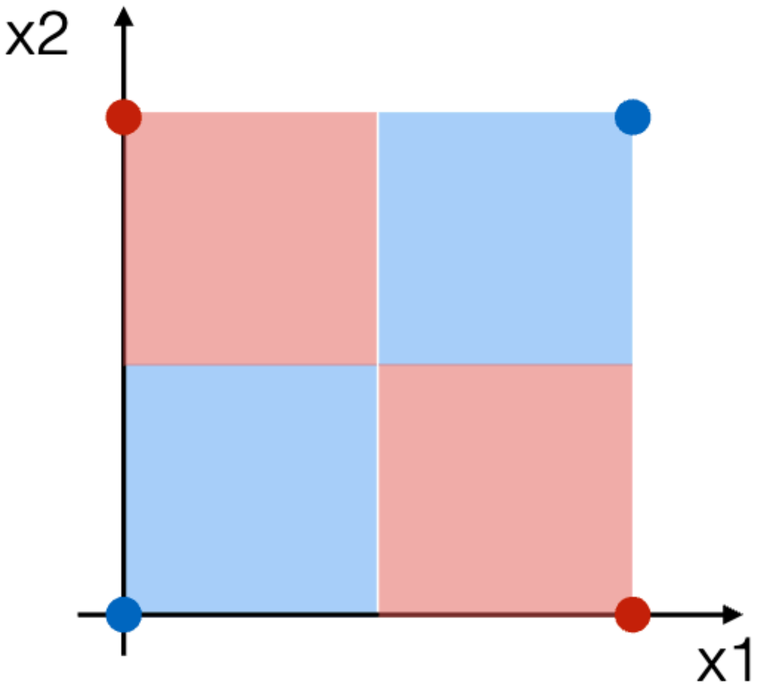
Curse of dimensionality

The underlying issue is that as dimensionality increases, the volume of the space grows so fast that the available data becomes sparse.
It becomes difficult to tell how regions of feature space correspond to the response



Curse of dimensionality

Sample	x1	x2	y
1	0	0	0
2	1	1	0
3	0	1	1
4	1	0	1



Curse of dimensionality

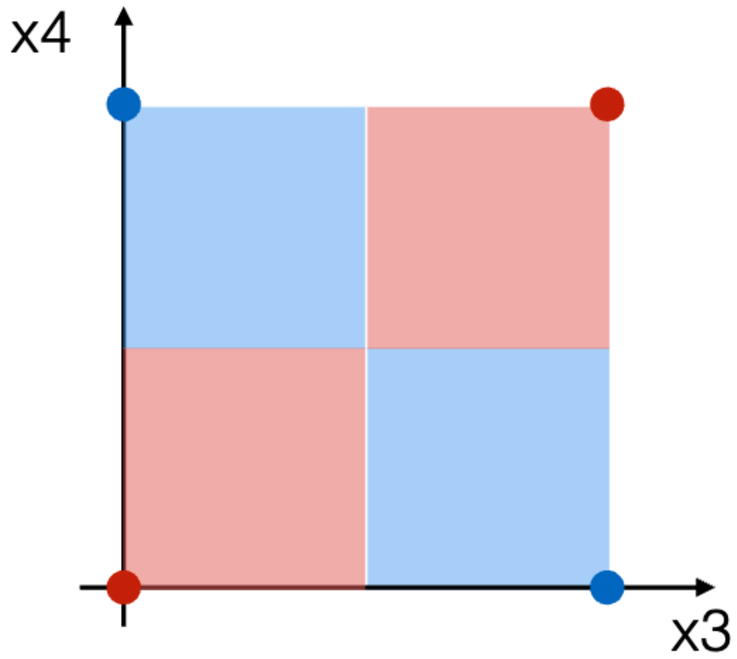
Sample	x1	x2	x3	x4	y
1	0	0	1	0	0
2	1	1	0	1	0
3	0	1	1	1	1
4	1	0	0	0	1

Signal dimensions Noise dimensions

Curse of dimensionality

Your model might learn…

Sample	x3	x4	y
1	1	0	0
2	0	1	0
3	1	1	1
4	0	0	1



…decision boundaries that are noise

Hughes Phenomenon*

With a fixed number of training samples, the predictive power of a classifier or regressor first increases as number of features used increases, but then decreases

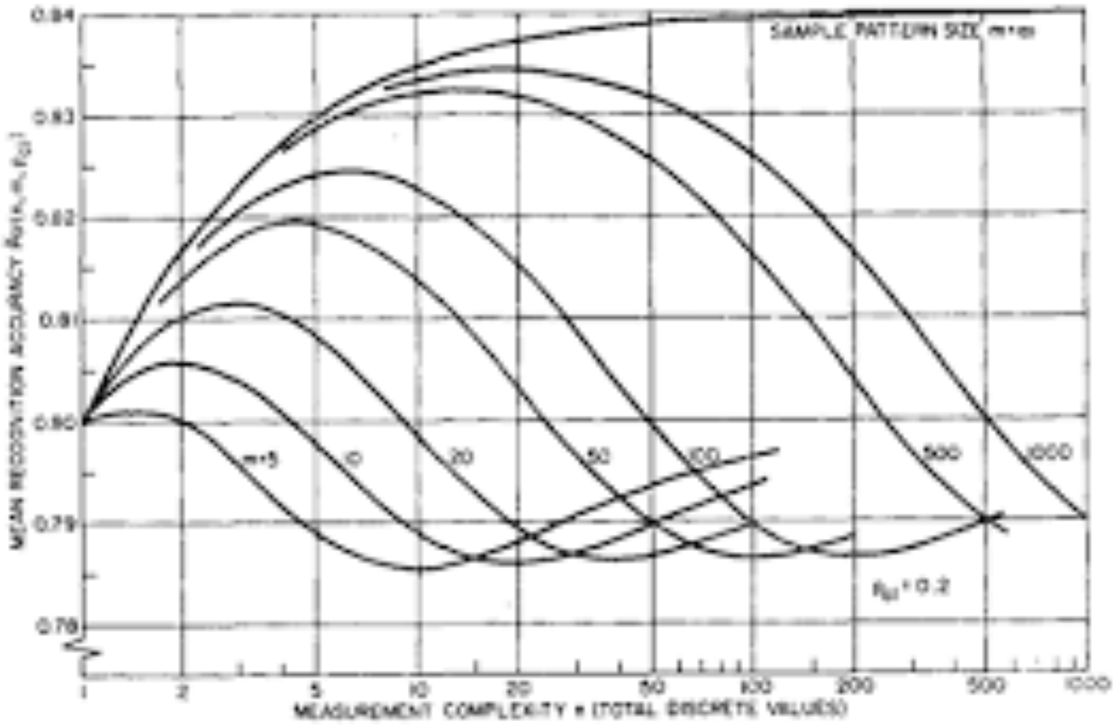


Fig. 4. Finite data set accuracy ($p_{ci} = \frac{1}{2}$).

Back to Linear Regression

In the regression setting, the standard linear model is:

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p \,.$$

If n >> p, then the least squares estimates tend to have low variance and will generalize well to unseen data.

If n not much bigger than p, least squares (even with a linear model!) will have high variance and can overfit.

If p > n, there is no longer a unique least squares estimate

Back to Linear Regression

It is often the case that some or many of the variables used in a multiple regression model are in fact not associated with the response.

Including such *irrelevant* variables leads to unnecessary complexity in the resulting model.

By removing these variables—that is, by setting the corresponding coefficient estimates to zero—we can obtain a model that is more easily interpreted.

Now least squares is extremely unlikely to yield any coefficient estimates that are exactly zero. In what follows, we will see some approaches for automatically performing *feature selection*—that is, for excluding irrelevant variables from a multiple regression model.

Solutions

Subset Selection

Identify a subset of the p features that we think are related to the response

Shrinkage

Fit a model on all p features, but shrink their coefficients toward zero to reduce model variance

Dimension Reduction

Project p features into a lower dimensional space, then build a model in this lower space

Solutions

Subset Selection

Identify a subset of the p features that we think are related to the response

Shrinkage

Fit a model on all p features, but shrink their coefficients toward zero to reduce model variance

Dimension Reduction

Project p features into a lower dimensional space, then build a model in this lower space

Future lecture (unsupervised)

Subset Selection

Best Subset Selection

To perform *best subset selection*, we fit a separate least squares regression for each possible combination of the p input variables.

Algorithm:

1. Start with a model M_0 that has no predictors.
2. For $k = 1, 2, \ldots, p$:
   (a) Fit all models that contain exactly k predictors.
   (b) Pick the best model M_k
3. Pick the best model from $M_0, \ldots, M_p$ using validation set R^2 or other metric.

Best Subset Selection

Pros: Exhaustive; finds best possible subset of features

Cons: While best subset selection is a simple and conceptually appealing approach, it suffers from computational limitations. The number of possible models that must be considered grows rapidly as the number of input variables p increases

e.g. if p = 10, you have to consider approximately 1,000 models

e.g. if p = 20, you have to consider over 1,000,000 models!

Stepwise Selection

Stepwise selection is a computationally efficient alternative to best subset selection

- Forward Stepwise Selection
- Backward Stepwise Selection
- Hybrid approaches

Forward Stepwise Selection

Forward stepwise selection greedily adds variables from the p input variables that give the greatest additional improvement to model fit.

Algorithm:

1. Start with a model M_0 that has no predictors.
2. For $k = 0, 1, \ldots, p - 1$:
   (a) Consider all $p - k$ models that add one predictor to M_k.
   (b) Pick the best model M_{k+1}
3. Pick the best model from $M_0, \ldots, M_p$ using validation set R^2 or other metric.

Backward Stepwise Selection

Backwards stepwise selection greedily removes variables from the p input variables that give the greatest additional improvement to model fit.

Algorithm:

1. Start with a model M_p that has all predictors.
2. For $k = p, p - 1, \ldots, 1$:
   (a) Consider all k models that remove one predictor from M_k.
   (b) Pick the best model M_{k-1}
3. Pick the best model from $M_0, \ldots, M_p$ using validation set R^2 or other metric.

Hybrid approaches

As another alternative, hybrid versions of forward and backward stepwise selection are available, in which variables are added to the model sequentially, in analogy to forward selection. However, after adding each new variable, the method may also remove any variables that no longer provide an improvement in the model fit.

Stepwise Selection methods are computationally efficient alternative to best subset selection, however they are not guaranteed to get best subset of input variables for the final model

Regularization

Regularization Methods

The subset selection methods involve using least squares to fit a linear model that contains a subset of the predictors.

As an alternative, we can fit a model containing all p input variables using a technique that *constrains* or *regularizes* the coefficient estimates, or equivalently, that *shrinks* the coefficient estimates towards zero.

Shrinking coefficients reduces the model variance.

Ridge Regression

Recall from that the least squares fitting procedure estimates $\beta_0, \beta_1, \ldots, \beta_p$ using the values that minimise:

$$\text{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2$$

Ridge regression (also known as L2 regularization) estimate the values that minimise:

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

penalty term

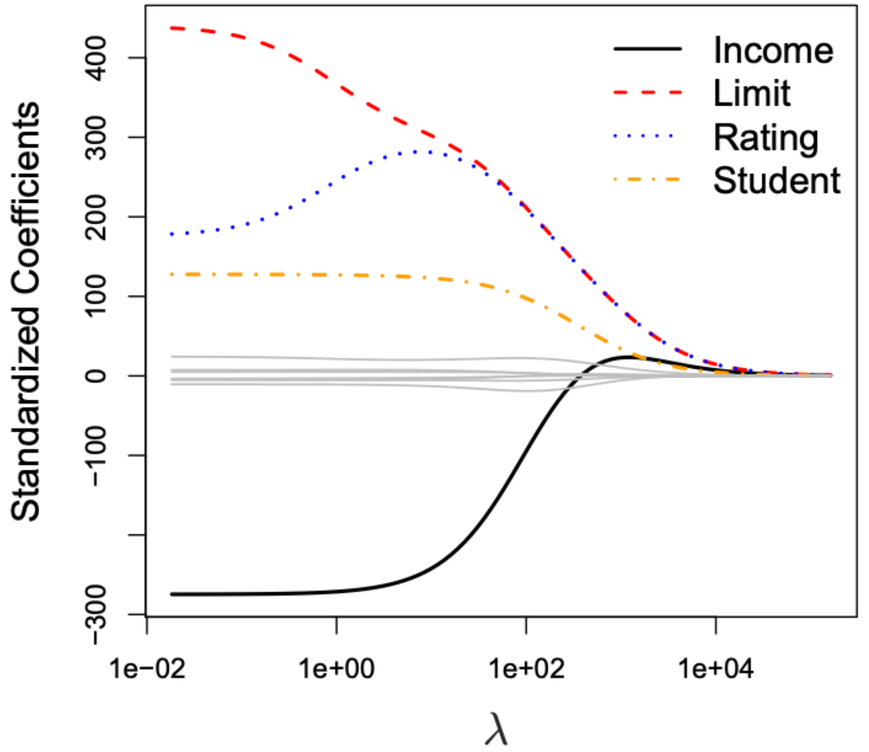
where $\lambda \geq 0$ is λ is a hyperparameter that needs to be tuned and serves to control the relative impact of the two terms on the regression coefficient estimates

Ridge Regression

When $\lambda = 0$, the penalty term has no effect, and ridge regression produces least squares estimates.

As $\lambda \rightarrow \infty$, the impact of the shrinkage penalty grows. Ridge regression estimates $\rightarrow 0$.

Finding a good value of λ using a validation set is crucial to good ridge regression performance.



Ridge Regression

Pro: Computationally feasible, reduces variance in linear regression when $p > n$ or n is not much larger than p, allows for a unique solution when $p > n$

Con: Includes all p predictors in final model, does not perform feature selection (which boosts interpretability) by setting any β to zero. Increasing the value of λ will tend to reduce the magnitudes of the coefficients, but will not result in exclusion of any of the variables.

Lasso Regression

In the case of the Lasso (L1 regularization), the penalty term has the effect of forcing some of the coefficient estimates to be exactly equal to zero when the tuning parameter λ is sufficiently large.

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j|$$

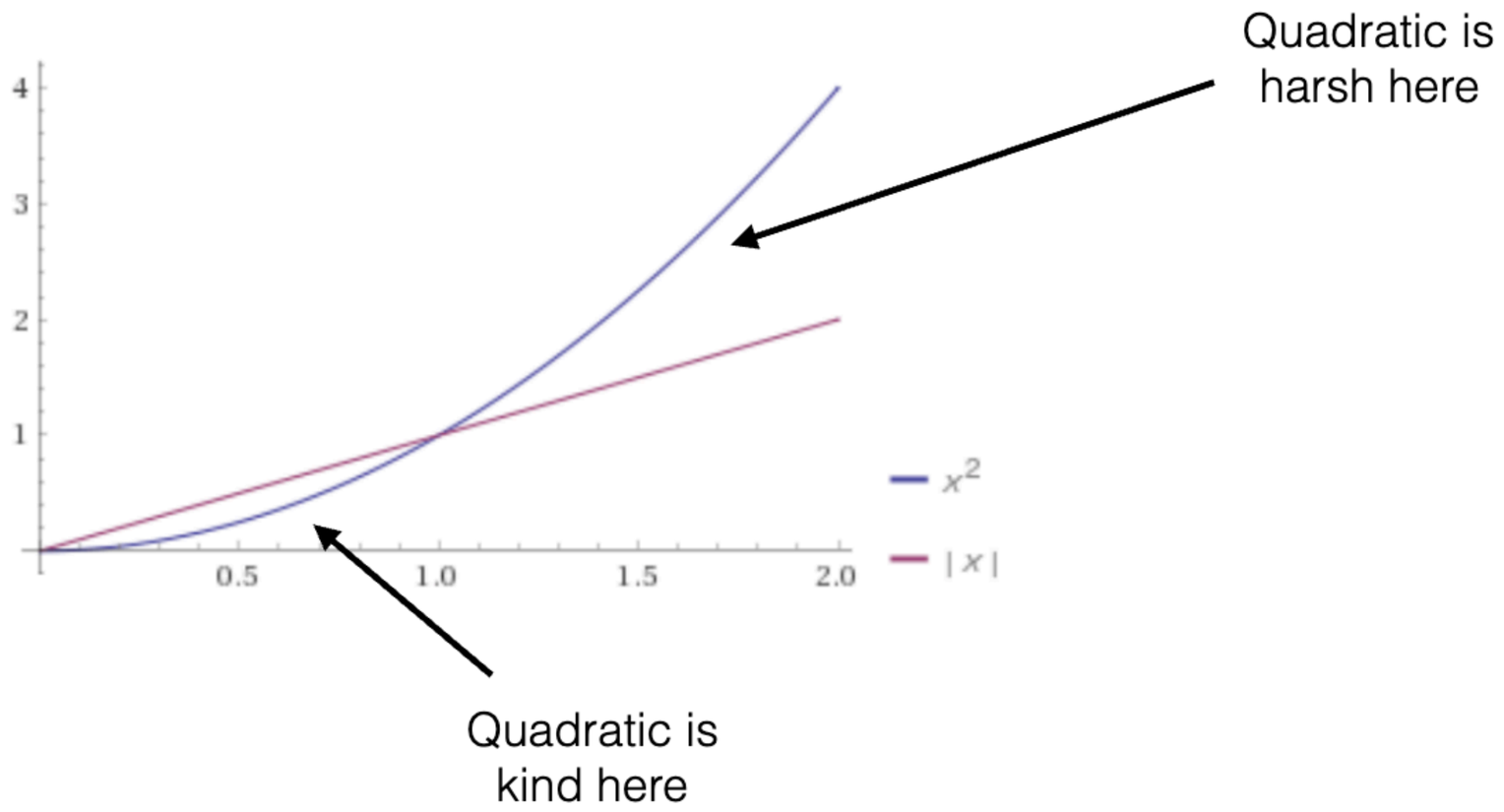
penalty term

Hence, much like best subset selection, the lasso performs *variable selection*. As a result, models generated from the lasso are generally much easier to interpret than those produced by ridge regression. We say that the lasso yields *sparse* models—that is, models that involve only a subset of the variables

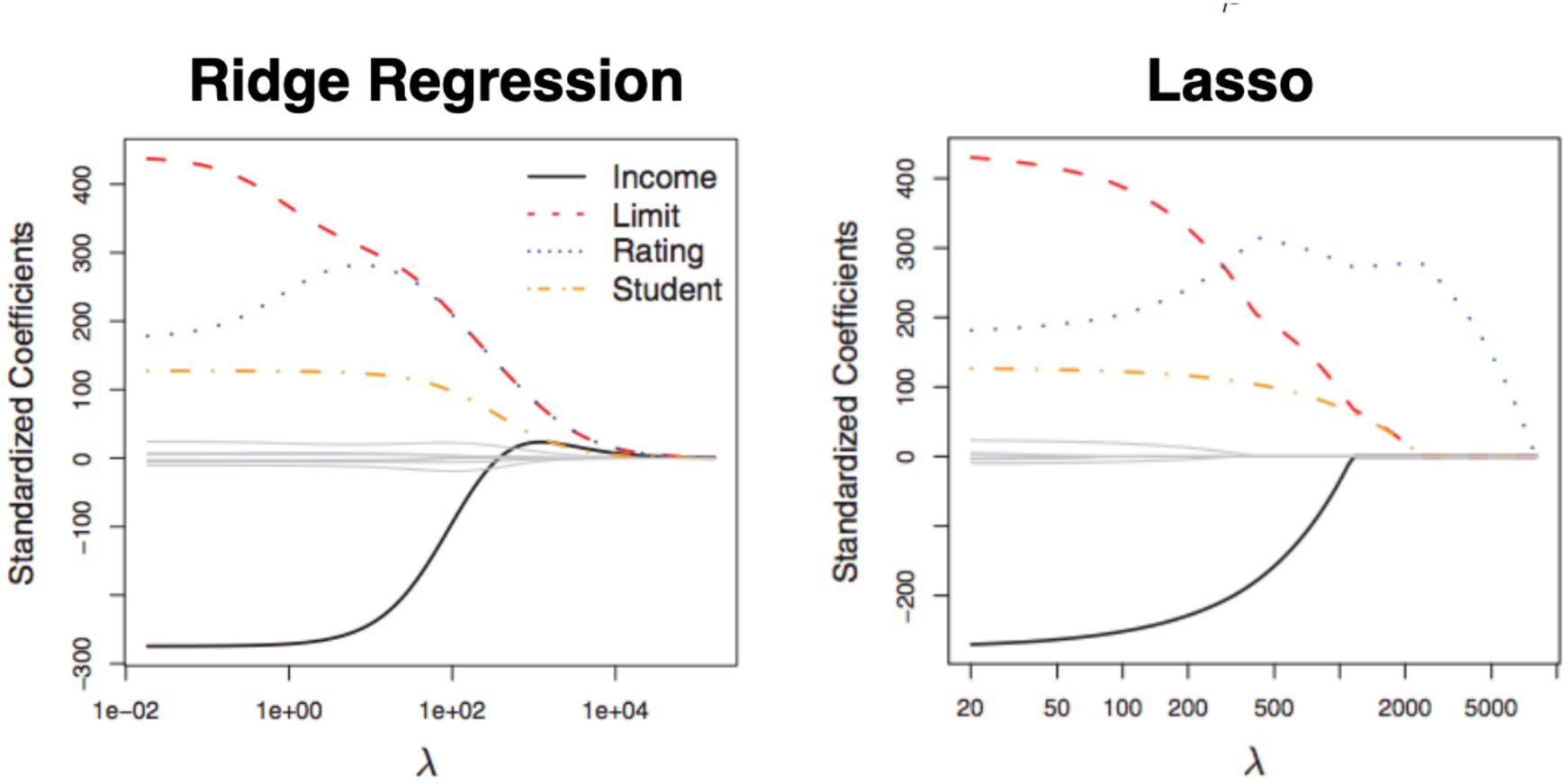
$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6$$

$\beta_2 = 0$, $\beta_4 = 0$, $\beta_5 = 0$

Ridge vs. Lasso Penalty



Ridge vs. Lasso Penalty



Ridge vs. Lasso Penalty

Neither is universally better than the other.

Ridge regression will perform better when the response is a function of many predictors, all with coefficients roughly of similar magnitude.

Lasso will perform better when a small number of predictors have substantial coefficients, and the remaining are small or equal zero.

Lasso Regression

Pro: Computationally feasible, reduces variance in linear regression when $p > n$ or n is not much larger than p, allows for a unique solution when $p > n$, performs feature selection (offers interpretability)

Con: Not as good as ridge when all predictors have significant and roughly similarly sized coefficients

Elastic Net

Elastic Net is used in regression models and combines both L1 (Lasso) and L2 (Ridge) penalties to achieve benefits from both types of regularization.

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda_2\sum_{j=1}^{p}\beta_j^2 + \lambda_1\sum_{j=1}^{p}|\beta_j|$$

Decision Trees

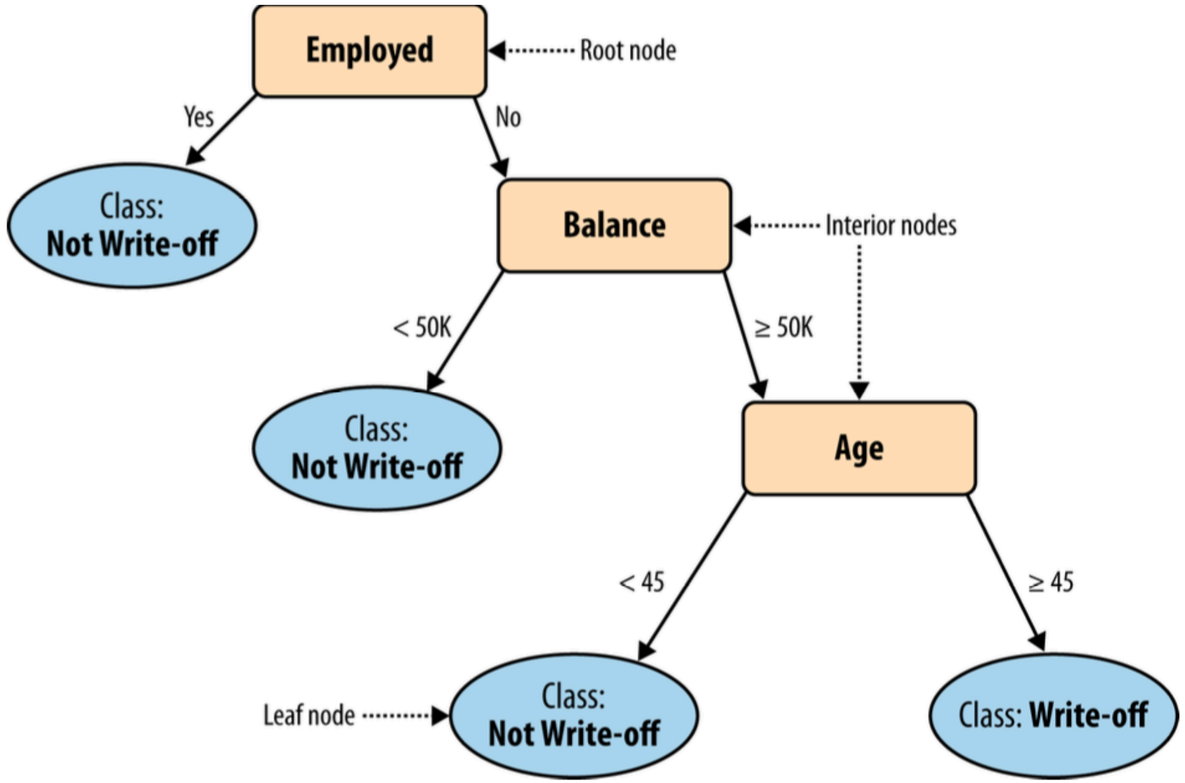
Moving Beyond Linearity

So far, our focus was on linear models:

- They are relatively simple to describe and implement, and have advantages over other approaches in terms of interpretation and inference
- However, they have significant limitations in terms of predictive power. This is because the linearity assumption is almost always an approximation, and sometimes a poor one

Decision Trees

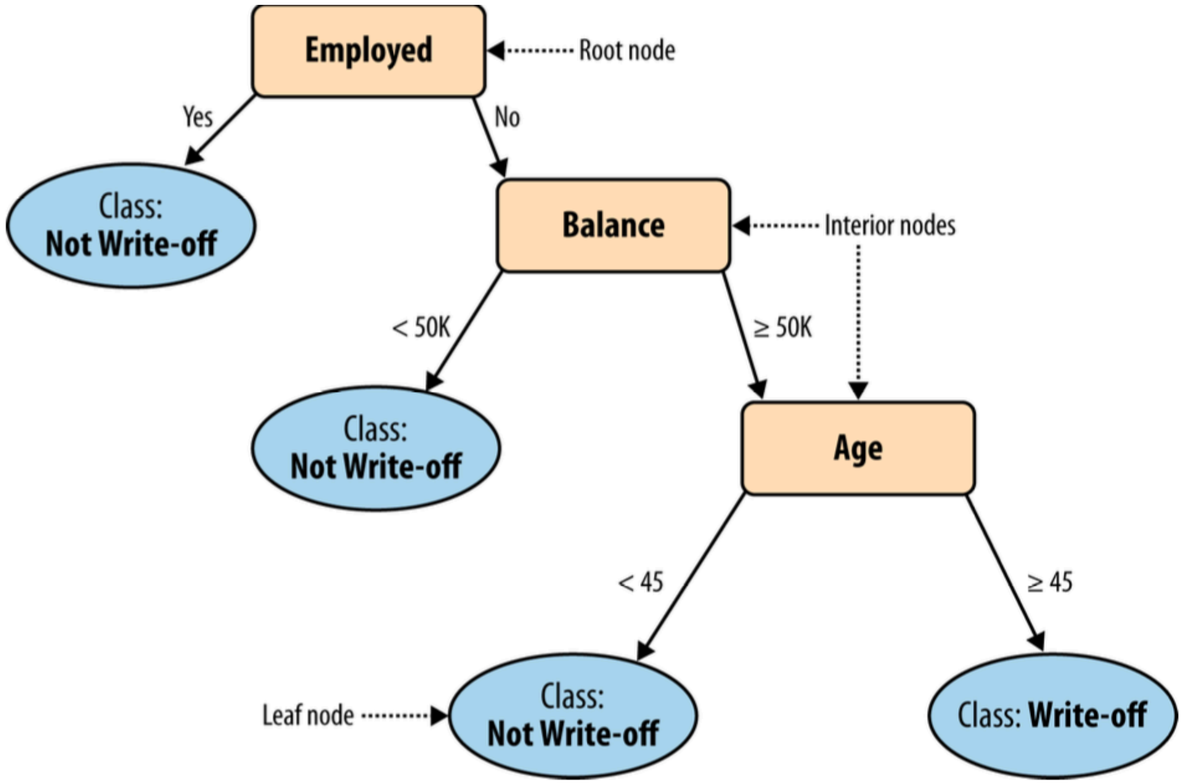
Decision trees work by recursively partitioning the data into subsets based on the values of input features, creating a tree-like structure where each interior node represents a decision based on a feature, and each leaf node represents a predicted output. They can be used for both classification and regression tasks.



Decision Trees

Algorithm for classification (greedy strategy):

1. Start by selecting the most informative attribute as the root node
2. If the remaining examples are all positive (or all negative), then we are done
3. If there are some positive and some negative examples, then choose the best attribute to split them
4. If there are no examples left, it means that no example has been observed for this combination of attribute values → return the most common output value for the parent
5. If there are no attributes left, but both positive and negative examples, it means that these examples have exactly the same description, but different classifications (error in the data or no observed attribute to distinguish them)→ return the most common output value of the remaining examples



Selecting Informative Attributes

The most common criterion for selecting informative attributes (i.e. for splitting the tree) is called information gain and it is based on a purity measure called entropy.

Entropy is a measure of disorder of how mixed (impure) the segment is.

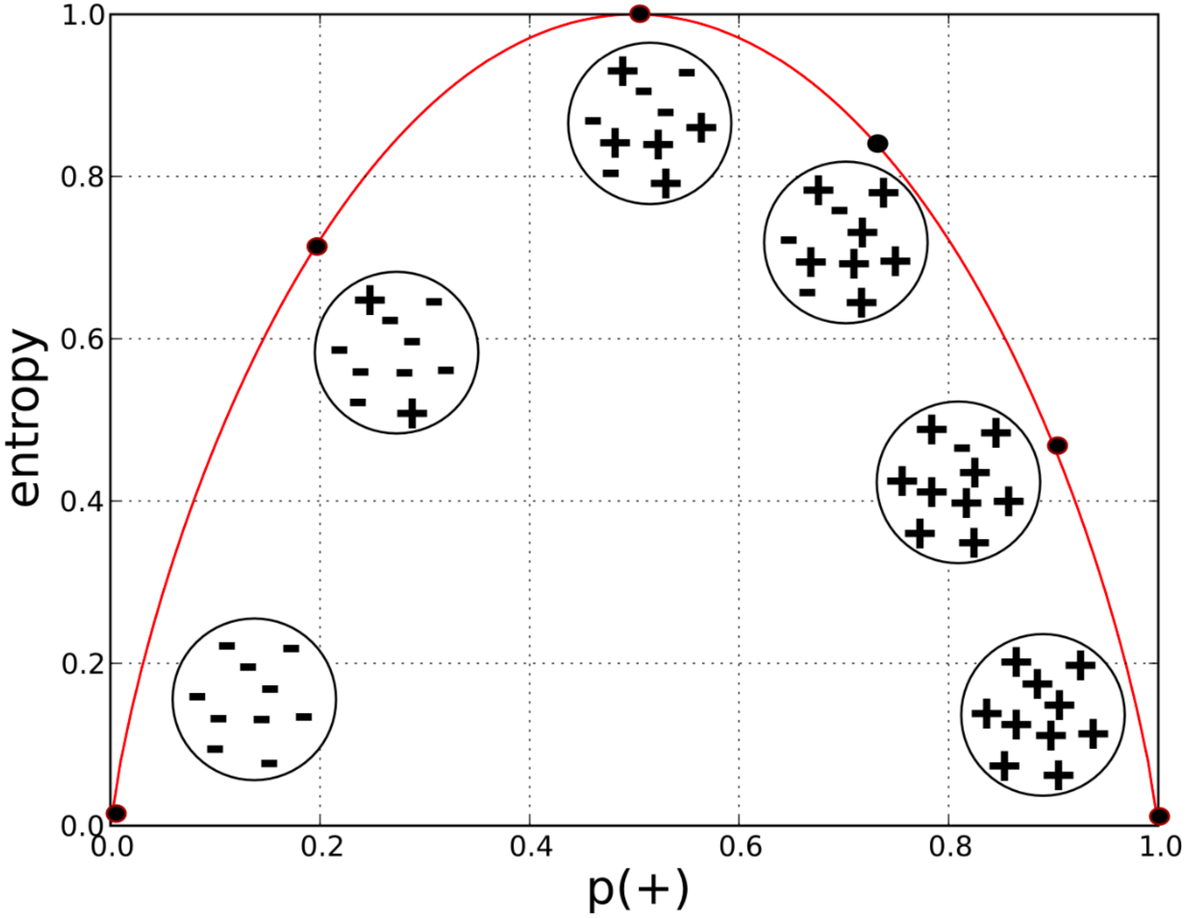
$$entropy = - p_1 \log (p_1) - p_2 \log (p_2) - \cdots$$

Example: Consider a set *S* of 10 people with 7 of the *non-write-off* class and 3 of the *write-off* class:

$$p(\text{non-write-off}) = 7 / 10 = 0.7$$

$$p(\text{write-off}) = 3 / 10 = 0.3$$

$$\begin{aligned} entropy(S) &= -[0.7 \times \log_2 (0.7) + 0.3 \times \log_2 (0.3)] \\ &\approx -[0.7 \times -0.51 + 0.3 \times -1.74] \\ &\approx 0.88 \end{aligned}$$



Entropy of a two-class set as a function of p(+). For example, a mixed up segment with lots of write-offs and lots of non-write-offs would have high entropy.

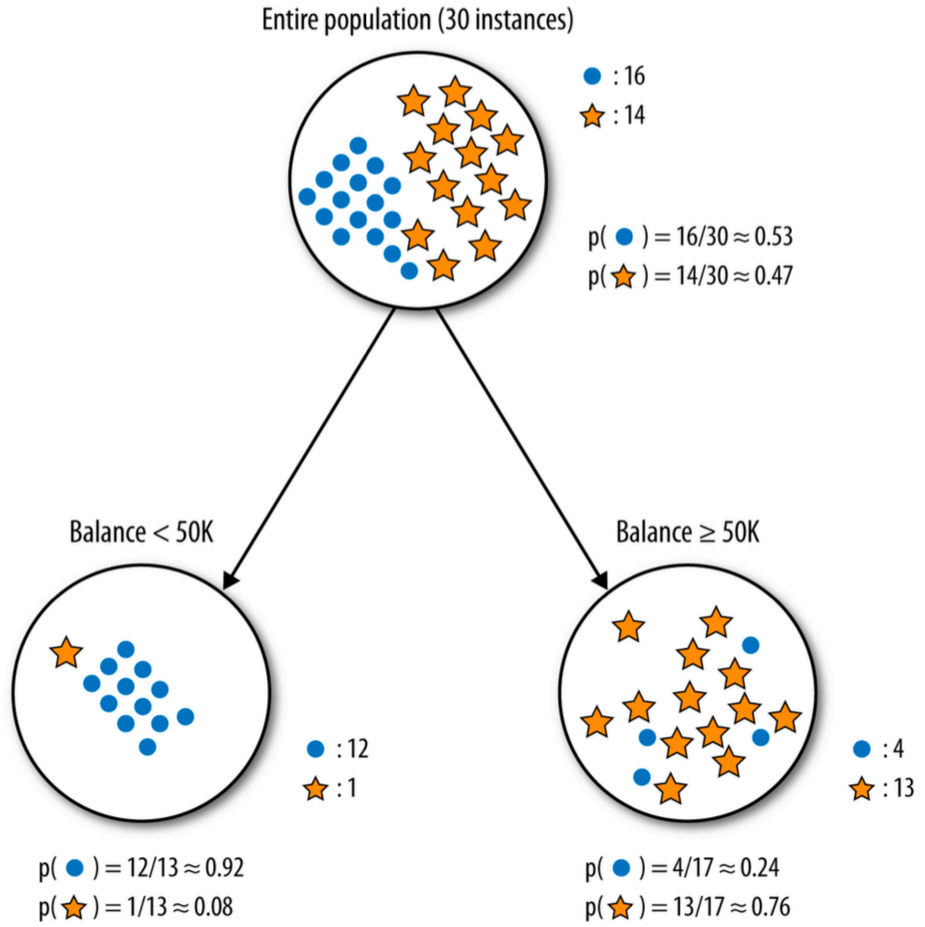
Selecting Informative Attributes

Information gain measures the *change* in entropy due to any amount of new information being added; in the context of supervised segmentation, we consider the information gained by splitting the set on all values of a single attribute.

$$IG(parent, children) = entropy(parent) - [p(c_1) \times entropy(c_1) + p(c_2) \times entropy(c_2) + \cdots]$$

The entropy of the *parent* is:

$$\begin{aligned} entropy(parent) &= -[p(\bullet) \times \log_2 p(\bullet) + p(\star) \times \log_2 p(\star)] \\ &\approx -[0.53 \times -0.9 + 0.47 \times -1.1] \\ &\approx 0.99 \quad \text{(very impure)} \end{aligned}$$



Splitting the "write-off" sample into two segments, based on splitting the Balance attribute (account balance) at 50K.

Selecting Informative Attributes

The entropy of the *left* child is:

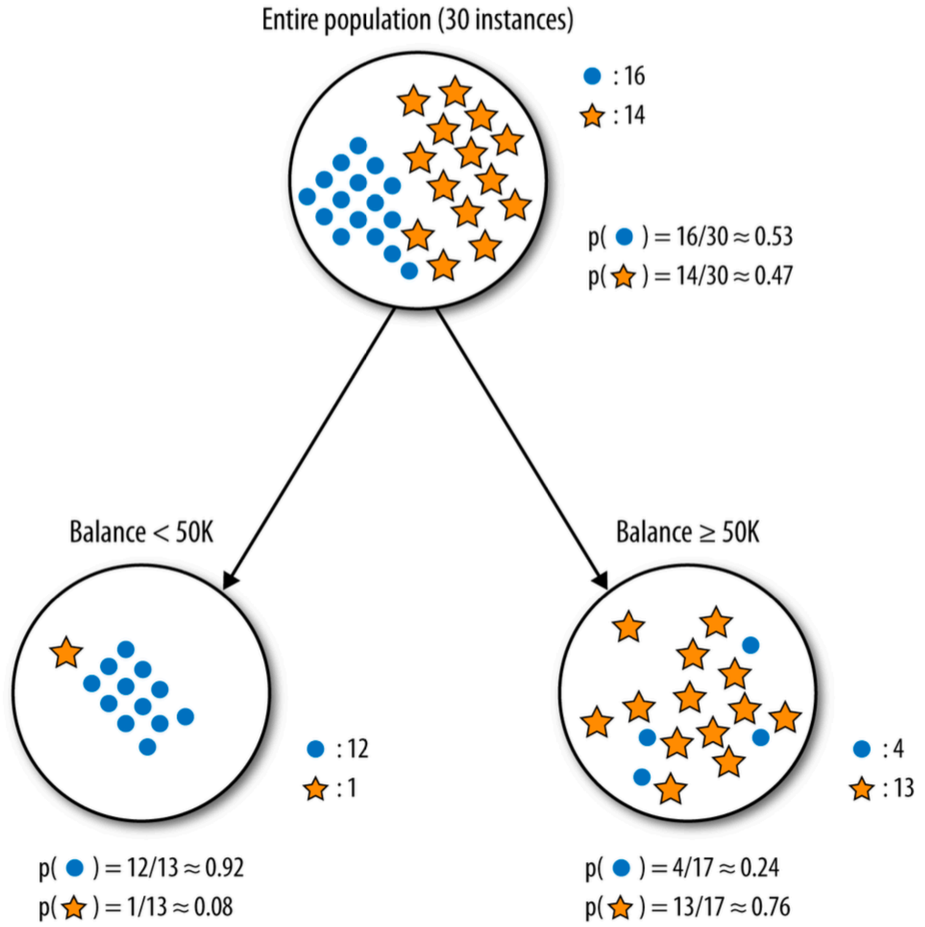
$$\begin{aligned} entropy(Balance < 50K) &= -[p(\bullet) \times \log_2\ p(\bullet) + p(\star) \times \log_2\ p(\star)] \\ &\approx -[0.92 \times (-0.12) + 0.08 \times (-3.7)] \\ &\approx 0.39 \end{aligned}$$

The entropy of the *right* child is:

$$\begin{aligned} entropy(Balance \geq 50K) &= -[p(\bullet) \times \log_2\ p(\bullet) + p(\star) \times \log_2\ p(\star)] \\ &\approx -[0.24 \times (-2.1) + 0.76 \times (-0.39)] \\ &\approx 0.79 \end{aligned}$$

The information gain of this split is:

$$\begin{aligned} IG &= entropy(parent) - [p(\text{Balance} < 50\text{K}) \times entropy(\text{Balance} < 50\text{K}) \\ &\qquad + p(\text{Balance} \geq 50\text{K}) \times entropy(\text{Balance} \geq 50\text{K})] \\ &\approx 0.99 - [0.43 \times 0.39 + 0.57 \times 0.79] \\ &\approx 0.37 \end{aligned}$$

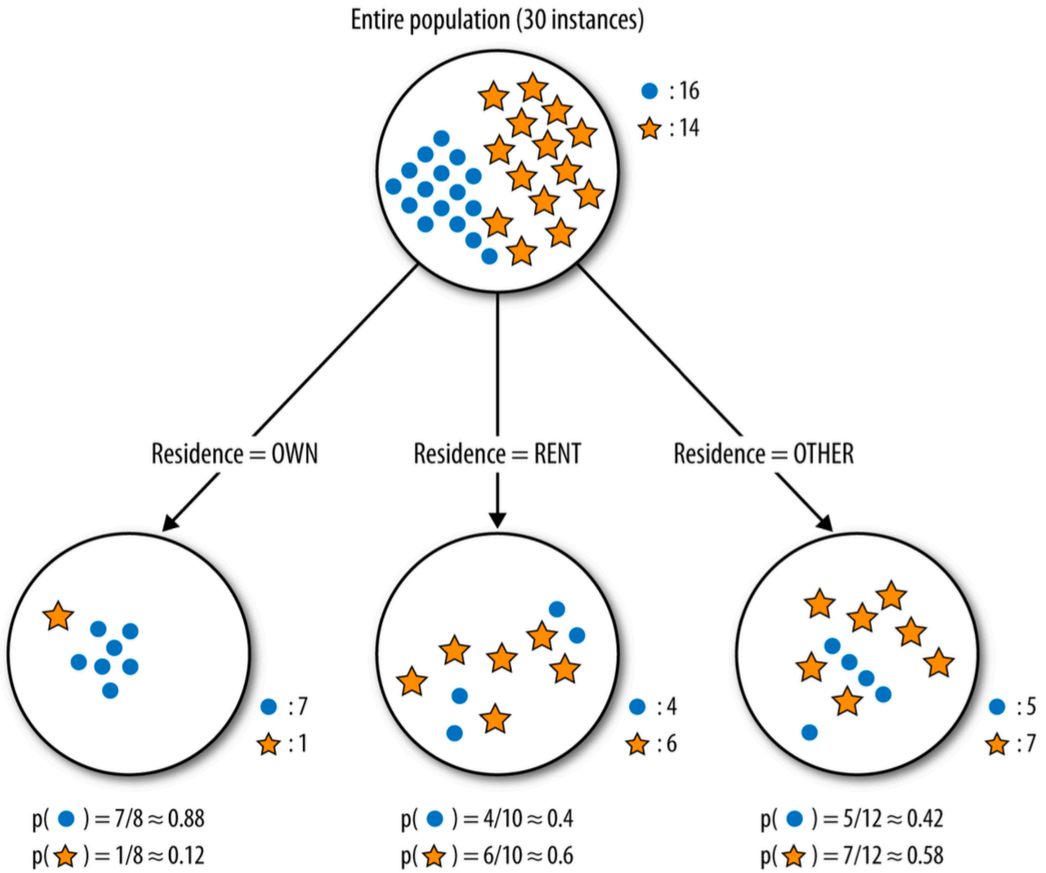


Splitting the "write-off" sample into two segments, based on splitting the Balance attribute (account balance) at 50K.

Selecting Informative Attributes

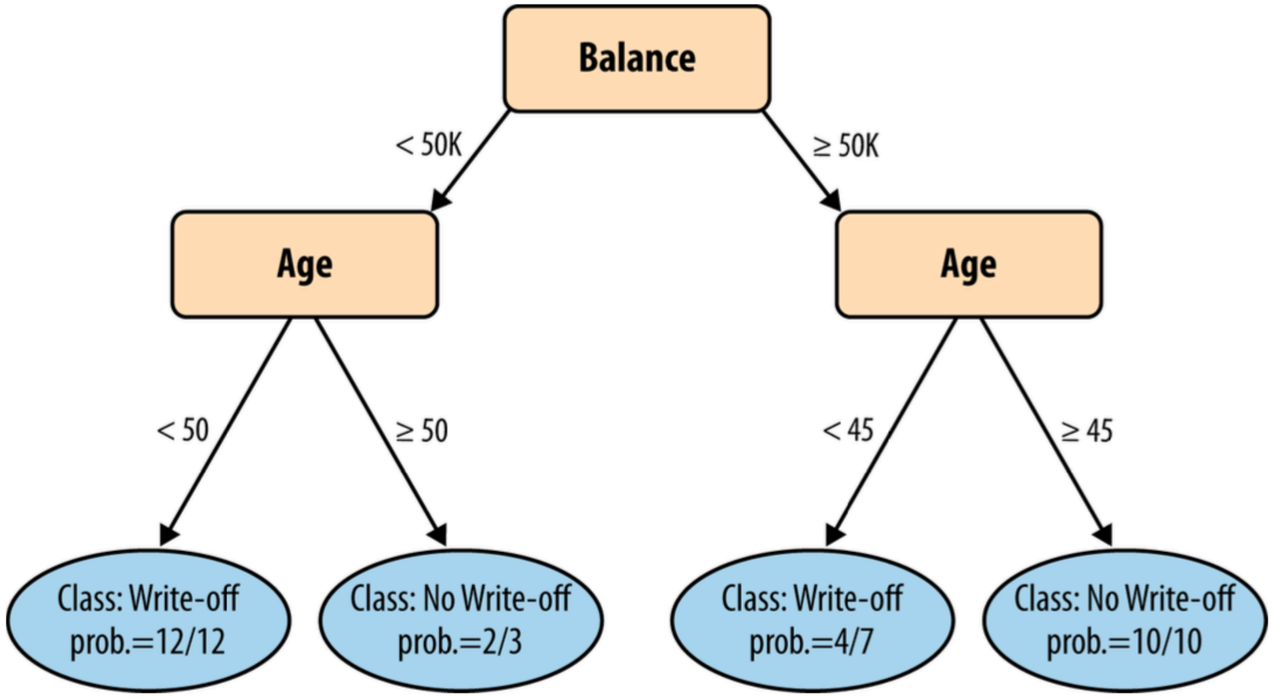
The information gain of this split is:

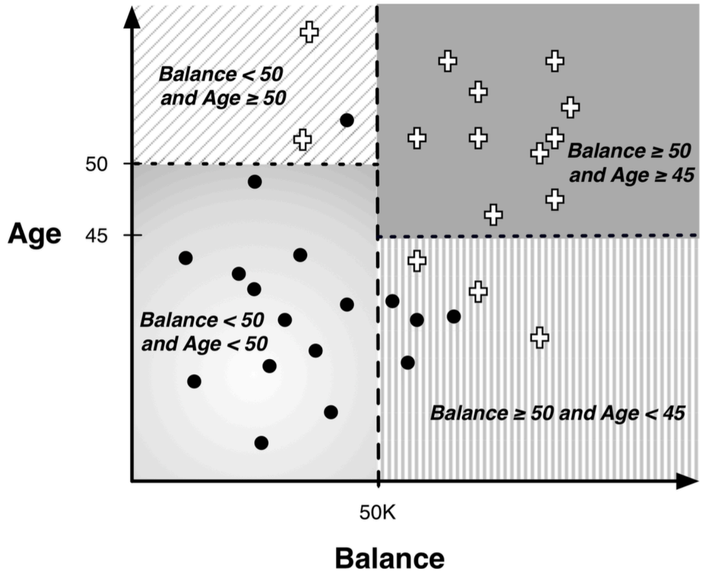
$$entropy(parent) \approx 0.99$$
$$entropy(\text{Residence=OWN}) \approx 0.54$$
$$entropy(\text{Residence=RENT}) \approx 0.97$$
$$entropy(\text{Residence=OTHER}) \approx 0.98$$
$$IG \approx 0.13$$



A classification tree split on the three-valued Residence attribute.

Visualizing Segmentations






```
IF (Balance < 50K) AND (Age < 50) THEN Class=Write-off
IF (Balance < 50K) AND (Age ≥ 50) THEN Class=No Write-off
IF (Balance ≥ 50K) AND (Age < 45) THEN Class=Write-off
IF (Balance ≥ 50K) AND (Age < 45) THEN Class=No Write-off
```

A classification tree and the partitions it imposes in instance space. The black dots correspond to instances of the class Write-off, the plus signs correspond to instances of class non-Write-off. The shading shows how the tree leaves correspond to segments of the population in instance space. Additionally, this can be represented in the form of rules: if we trace down a single path from the root node to a leaf, collecting the conditions as we go, we generate a rule.

Appendix