Gini Index : $1 - \sum_{i=0}^{c-1} p_i(t)^2$

Entropy = $-\sum_{i=0}^{c-1} p_i(t) \log_2(p_i(t))$

Classification error: $1 - \max[p_i(t)]$

* $p_i(t)$ is probability of class i at node t, c is total # of classes.

* Gain = P − M, P → impurity before split, M → weighted impurity after split

* $\log_2(x) = \ln(x)/\ln(2)$

Impurity: $-\sum_{i=0}^{c-1} p_i(t) \log_2(p_i(t))$

$= -\left(\left(\frac{3}{10}\log_2\left(\frac{3}{10}\right)\right) + \left(\frac{7}{10}\log_2\left(\frac{7}{10}\right)\right)\right)$

$= 0.88 = P$ (Parent Impurity)

Hunt's Algorithm with Entropy Impurity:

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No 0
2	No	Married	100K	No 0
3	No	Single	70K	No 0
4	Yes	Married	120K	No 0
5	No	Divorced	95K	Yes 1
6	No	Married	60K	No 0
7	Yes	Divorced	220K	No 0
8	No	Single	85K	Yes 1
9	No	Married	75K	No 6
10	No	Single	90K	Yes 1

① Start with single classification node:

Defaulted	
1	0
3	7

② Split on some attribute:

Home Owner

No →

Def.	
1	0
3	4

Yes → (Pure Node Becomes Leaf)

Def.	
1	0
0	3

③ Calculate weighted Impurity for k children

$M = \sum_{i=0}^{k} \frac{n_i}{n} \text{Entropy}(i) = \frac{7}{10}\text{Entropy}(\text{No}) + \frac{3}{10}\text{Entropy}(\text{yes})$

$= \frac{7}{10}\left(-\left(\frac{3}{7}\log_2\frac{3}{7} + \frac{4}{7}\log_2\frac{4}{7}\right)\right) + \frac{3}{10}\left(-\left(\frac{0}{3}\log_2\frac{0}{3} + \frac{3}{3}\log_2\frac{3}{3}\right)\right)$ (0 by convention)

$= \frac{7}{10}\left(-(-0.05238 - 0.46134)\right) = 0.3596 = M$

④ Next calculate Gain = P − M = 0.88 − 0.3596 = 0.5204

↳ Calculate this metric for each attribute and split on feature with highest gain.

↳ Repeat process until desired model fit

Impurity measures prefer many pure child nodes.

- Gain Ratio:

$$Gain\ Ratio = \frac{Gain_{split}}{Split\ Info} \qquad Split\ Info = -\sum_{i=1}^{k} \frac{n_i}{n} log_2 \frac{n_i}{n}$$

Parent Node, p is split into k partitions (children)
n_i is number of records in child node i

- Adjusts Information Gain by the entropy of the partitioning ($Split\ Info$).
  - Higher entropy partitioning (large number of small partitions) is penalized!

CarType	Family	Sports	Luxury
C1	1	8	1
C2	3	0	7
Gini	0.163		

SplitINFO = 1.52

CarType	{Sports, Luxury}	{Family}
C1	9	1
C2	7	3
Gini	0.468	

SplitINFO = 0.72

-(16/20) log2 (16/20) - (4/20)*log2(4/20)

CarType	{Sports}	{Family, Luxury}
C1	8	2
C2	0	10
Gini	0.167	

SplitINFO = 0.97

Gain Ratio adjusts for this.

Finding best cont. feature split value requires comparing all weighted impurities.

Sorted Values
Split Positions →

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
Annual Income	60	70	75	85	90	95	100	120	125	220
Gini	0.420	0.400	0.375	0.343	0.417	0.400	0.300	0.343	0.375	0.400	0.420
No	0	1	2	3	3	3	3	4	5	6	7
Yes	0	0	0	0	1	2	3	3	3	3	3
<=	0	1	2	3	4	5	6	7	8	9	10
>	10	9	8	7	6	5	4	3	2	1	0
Split Positions	55	65	72	80	87	92	97	110	122	172	230

Decision Tree Advantages :- Inexpensive, Fast, interpretable (small trees), robust to noise

Disadvantages : - space of possible trees very large & greedy approach doesn't always work
- inter-attribute interactions ignored, each decision boundry involves only 1 attribute

Metrics for Performance Evaluation

Confusion matrix

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

Accuracy (most common) : (TP+TN) / (TP+TN+FP+FN)

Precision (accuracy of positive predictions) = TP/(TP+FP)

Recall (Proportion of true positives model could catch) = TP/(TP+FN)

F-measure (?) : 2TP/(2TP+FN+FP)

Accuracy limitations: Class 0 = 9990, class 1 = 0 → If model always predicts class 0; accuracy = 99.9% but this is misleading since model never predicts class 1!

		PREDICTED CLASS	
	C(i|j)	Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	C(Yes|Yes)	C(No|Yes)
	Class=No	C(Yes|No)	C(No|No)

C(i|j): cost of classifying class j as class i

Cost Matrix	PREDICTED CLASS		
	C(i|j)	+	-
ACTUAL CLASS	+	-1	100
	-	1	0

Will be define ba probl

Model M_1	PREDICTED CLASS		
		+	-
ACTUAL CLASS	+	150	40
	-	60	250

Accuracy = 80%
Cost = 3910

Model M_2	PREDICTED CLASS		
		+	-
ACTUAL CLASS	+	250	45
	-	5	200

Accuracy = 90%
Cost = 4255

Model Evaluation:

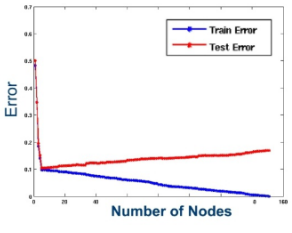
↳ Holdout: Reserve k% for training, (100−k)% testing

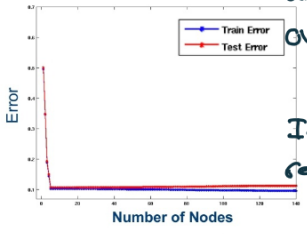
↳ random subsampling: repeated holdout

↳ Cross-Validation: Train on k−1 partitions, test on remaining partition

Final performance is always Average of all runs

Model Overfitting





(Using twice the number of train data instances)

- Increasing the size of training data reduces the difference between training and testing errors at a given size of model

underfitting is when model is too simple, high train & test error

overfitting is when model is too complex, does not generalize well

↳ high test error, low train error

Increasing training set size, reducing model complexity and reducing # of features are all ways to deal with overfitting

Model Selection : use small validation set to estimate generalization error.

↳ choose simpler of two models with similar generalization

↳ We can use pessimistic error to estimate generalization error of decision tree taking error into account

↳ we can stop splitting early (Pre-pruning)

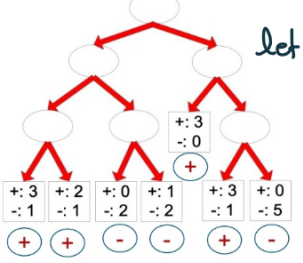
↳ if impurity measures don't improve

↳ if error(T) becomes acceptable

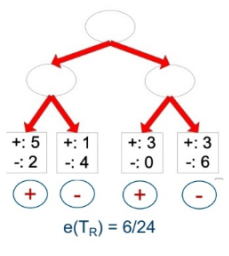
↳ Post-prune → grow tree to max size → trim nodes bottom-up

if error(T) improves, replace subtree w most frequent record class

↳ or replace subtree w most frequent branch



$e(T_L) = 4/24$



$e(T_R) = 6/24$

let $\Omega = 1$

$err_{gen}(T_L) = 4/24 + 1*7/24 = 11/24 = 0.458$

$err_{gen}(T_R) = 6/24 + 1*4/24 = 10/24 = 0.417$

$$err_{gen}(T) = err(T) + \Omega \times \frac{k}{N_{train}}$$

- $err(T)$: error rate on all training records
- Ω: trade-off hyper-parameter (similar to α)
  - Relative cost of adding a leaf node
- k: number of leaf nodes
- N_{train}: total number of training records

Nearest Neighbors Classifiers: To Classify an unknown record ① compute distance to other records. ② Identify k-nearest neighbors. ③ use class labels of nearest neighbors to classify unknown record, use a context dependent distance measure and take a weighted ($w = 1/d^2$) majority vote of the neighbors. If k is too small → sensitive to noise, too large → missclassification. May need feature standardization. Requires presence of all attributes, missing values need to be filled in, irrelevant features add noise, redundant features skew classifaction to certain class.

INTRO: Why data mining?: increased data collection speed and quantity, cheaper & more powerful compute, strong competitive pressure, helpful for scientists (automated analysis, hypothesis formation). **What is data mining?:** Non-trivial extraction of implicit, previously unknown and potentially useful information from data. **KDD** process: [input data] → [preprocessing] → [data mining] → [postprocessing] → [information] . **Why not classical data analysis?:** scalability, high dimensionality, heterogeneous and complex data, data ownership, distributed data, non-traditional analysis. **Origins:** Draws on machine learning/AI, pattern recognition, statistics and database systems. **Data Mining Tasks:** *predictive tasks* (use known features to predict unknown features) → classification (predict discrete features), regression (predict continuous features), outlier detection (predict anomalous records), *descriptive tasks* (find human-interperetable patterns to describe data) → clustering (find groups of related objects), association rules (discover dependency rules between items in records of itemsets). **DATA: What is Data?:** Collection of objects (records) and their attributes (features). **Attribute Types:** Nominal (distinctness $=, \neq$, ID, eye color), Ordinal (order has meaning, $<, >$, rankings, height), Interval (differences have meaning, $+, -$, calendar dates, relative temperature scales), Ratio (ratios are meaningful, $\div, \times$, absolute scales of measure like kelvin or distance). **Data Characteristics:** Dimensionality (# of features), Sparsity (majority empty features), Resolution (patterns depend on scale), size (dataset size). **Types of Data:** Record (tabular, matrix → instances become points in n-D space, Document → each document becomes a word count vector, transaction → records of ID'd itemsets), Graph (networks), Ordered (sequence → braces used to indicate item(s) order, Spatio-Temporal → space & time dimensions like in a regional time series temperature heatmap). **Data Quality Problems:** Noise & Outliers (a noise *object* is an extraneous signal independent of true data, a noisy *feature* is when the extraneous signal modifies original feature values, an *outlier* could be noise or could be analysis target → ex. instance of fraud), Missing Values (incomplete data or inapplicable features → eliminate records, ignore features or estimate values), Duplicate data (ex same person with multiple emails → handling is context dependent). **Similarity & Dissimilarity:** similarity $\in [0, 1]$, dissimilarity $\in [0, \infty]$, **Minkowski** (generalized euclidean) → $d(x, y) = \left(\sum_{k=1}^{n} (x_k - y_k)^r\right)^{1/r}$ standardization may be necessary, **Mahalanobis** (takes distribution into account) → $M(x, y) = (x - y)^T \Sigma^{-1} (x - y)$ where Σ^{-1} is the feature covariance matrix, *Binary Vectors:* **Simple Matching** (# matches/# attributes) → $SMC(x, y) = (f_{11} + f_{00})/(f_{11} + f_{00} + f_{01} + f_{10})$, **Jaccard** (# 11 matches/# nonzero attributes) → $J(x, y) = (f_{11})/(f_{11} + f_{01} + f_{10})$, **Cosine** → $\cos(d_1, d_2) = < d_1, d_2 > / ||d_1||\, ||d_2||$, **Extended Jaccard** (for continuous or count variables) → $EJ(x, y) = (x \cdot y)/(||x||^2 + ||y||^2 - x \cdot y)$. **Preprocessing**: **Aggregation** (combining multiple attributes/objects into a single attribute/object for data reduction, change of scale or stablize (smooth) data), Sampling (randomly selected representative subset of the data considering sample size, with/without replacement or stratified sampling (sampling independently from partitions)), **Dimensionality Reduction** (curse of dimensionality: high dimensionality results in extremely sparse data making density/distance measures less meaningful) → **PCA** (principal component analysis involves finding eigenvalues & eigenvectors of covariance matrix and then projecting data onto eigenvector with largest eigenvalue), **Feature Reduction** (remove redundant/irrelevant features), **Discretization** (continuous variable mapped to ordinal). **EXPLORATION:** Preliminary exploration to better understand data, select best future preprocessing/analysis tools. **EDA:** focus was mainly visualization using clustering and anomaly detection, today these are major areas of interest. **Summary Statistics: Spread** → (mean,std, variance, range, iqr) → $AAD(x) = \frac{1}{m} \sum_{i=1}^{m} |x_i - \bar{x}|$, $MAD(x) = median\,(\{|x_1 - \bar{x}|, \ldots, |x_m - \bar{x}|\})$, $interquartile\ range(x) = x_{75\%} - x_{25\%}$, **frequency** (percentage occurrence), **mode** (most common occurrence), **Percentiles** (p^{th} percentile of an ordinal or continuous feature is the value strictly greater than p percent of all its other values), **Location** (mean is very sensitive to outliers, median or trimmed mean is better location measure). **Visualization:** converting data to visual or tabular formats for summarization, leveraging humans' well developed visual pattern recognition abilities, **Representation** (mapping of information to visual format), **Arrangement** (placement of visual elements within a display), **Selection for visualization** (elimination or de-emphasis of certain objects/features), **Histograms** (visualize distribution of continuous variables using "bins", 2-d for single feature distribution , 3-d for two feature joint distribution), **Box plots** (good for iqr, outliers, percentiles and median), **Scatter Plots** (features determine position, others can be used to change point shape, color, size, etc...), **Contour Plots** (partition spatial grid into regions where continuous feature values are similar, contour lines form region boundaries), **Data Matrix** (sort instances by class along one axis, arrange features along the other axis, calculate some metric for each instance-feature insersection), **Parallel Coordinates** (visualize each instance as a line colored by class, visualize movement of lines through sequential parallel axis observing patterns between classes), **Star Plot** (multiple features radiate from central point as individual axes, a line intersects the axes based on an objects feature values), **Chernoff Faces** (data features are mapped to face features). **On-Line Analytical Processing:** multidimensional array data representation (**Data Cube**) effective for aggregation operations, **1.** Identify and discretize selected features (features → cube dimensions, unique feature values → dimension indices), **2.** group-by unique tuples and sum target variable values or calculate group count, **3.** place target values in cube using the unique feature tuple values as cell indices, **example:** for sales, products and locations cube dimensions, select Product ID and Location, aggregating along sales to get total sales for each product at each location, many more combinations. **Slicing:** subset of cube specifying *one* value for one/more features, **Dicing:** subset cube specifying range of attribute values across dimensions, **Roll-up**: (zoom out), move *up* attribute hierarchy, a kind of aggregation (cities → countries), **Drill-Down:** (zoom in), move *down* attribute hierarchy increasing granularity/resolution, a kind of unravel (quarter → months).