Introduction to Data Mining

- An attempt at knowledge discovery
- Searching for patterns and structure in a sea of data
- Uses techniques from many disciplines, such as statistical analysis and machine learning
  - These techniques are not our main interest in this slides-set.

Knowledge Discovery in Databases (KDD)

- Data mining is actually one step of a larger process known as **knowledge discovery in databases** (KDD).
- The KDD process model comprises six phases
  - Data selection
  - Data cleansing
  - Enrichment
  - Data transformation or encoding
  - **Data mining**
  - Reporting and displaying discovered knowledge

Goals of Data Mining and Knowledge Discovery (PICO)

- **Prediction**:
  - Determine how certain attributes will behave in the future.
- **Identification**:
  - Identify the existence of an item, event, or activity.
- **Classification**:
  - Partition data into classes or categories.
- **Optimization**:
  - Optimize the use of limited resources.
- **Types of Discovered Knowledge**
  - Association Rules
  - Classification Hierarchies
  - Sequential Patterns
  - Patterns Within Time Series
  - Clustering

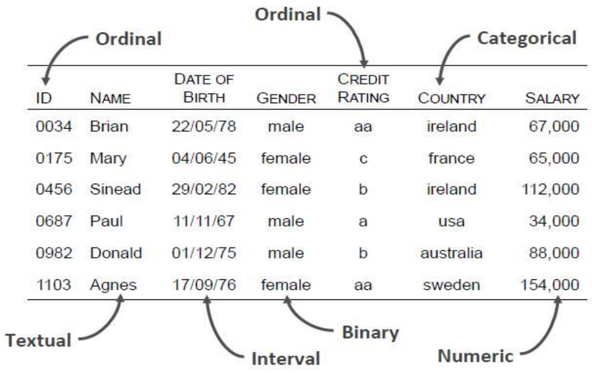
Goals of Data Mining

- **Association**
  - Finding patterns in data that associate instances of that data to related instances
    - Example: what types of books does a customer buy
- **Classification**
  - Finding patterns in data that can be used to classify that data (and possibly the people it describes)
    - Example “high-end buyers” and “low-end” buyers
  - This classification might then be used for **Prediction**
    - Which bank customers will default on their mortgages?
  - Categories for classification are known in advance

Goals (con't)

- **Clustering**
  - Finding patterns in data that can be used to classify that data (and possibly the people it describes) into categories determined by a similarity measure
    - Example: Are cancer patients clustered in any geographic area (possibly around certain power plants)?
  - Categories are *not* known in advance, unlike is the classification problem

Different Types of Data


Ordinal
Ordinal
Categorical
Textual
Interval
Binary
Numeric

ID	NAME	DATE OF BIRTH	GENDER	CREDIT RATING	COUNTRY	SALARY
0034	Brian	22/05/78	male	aa	ireland	67,000
0175	Mary	04/06/45	female	c	france	65,000
0456	Sinead	29/02/82	female	b	ireland	112,000
0687	Paul	11/11/67	male	a	usa	34,000
0982	Donald	01/12/75	male	b	australia	88,000
1103	Agnes	17/09/76	female	aa	sweden	154,000

Predictive Data Analytics

- Predictive data analytics models are reliant on the data that is used to build them—the **analytics base table** (**ABT**).
- The first step in designing an ABT is to decide on the **prediction subject**.
- An effective way in which to design ABTs is to start by defining a set of **domain concepts** in collaboration with the business, and then designing **features** that express these concepts in order to form the actual ABT.
- Features (both descriptive and target) are concrete numeric or symbolic representations of domain concepts.
- It is useful to distinguish between **raw features** that come directly from existing data sources and **derived features** that are constructed by manipulating values from existing data sources.
- Common manipulations used in this process include aggregates, flags, ratios, and mappings, although any manipulation is valid.

Predictive Data Analytics encompasses the business and data processes, and computational models that enable a business to make **data-driven decisions**.

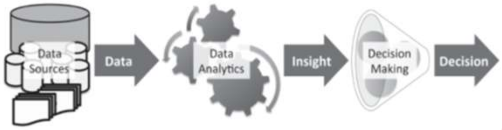
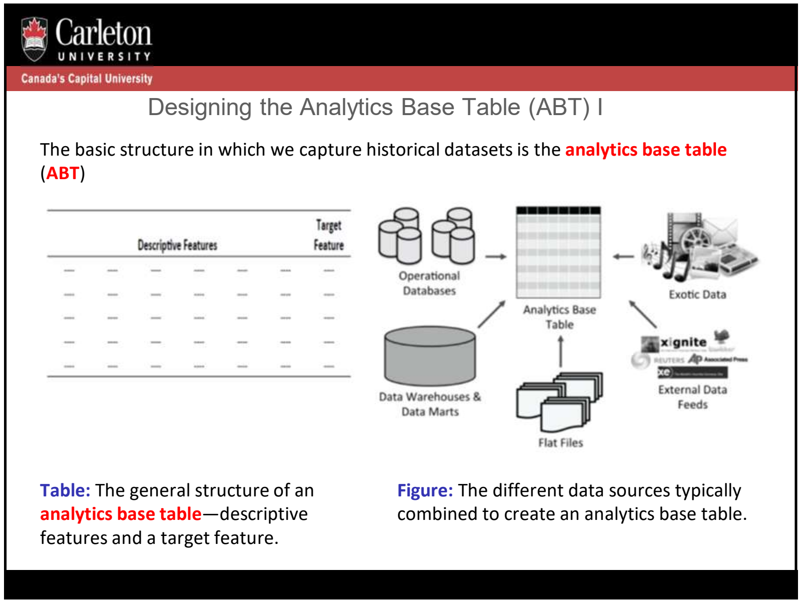


Figure: Predictive data analytics moving from **data** to **insights** to **decisions**.

Example Applications:

- Price Prediction
- Fraud Detection
- Dosage Prediction
- Risk Assessment
- Propensity modelling
- Diagnosis
- Document Classification
- . . .



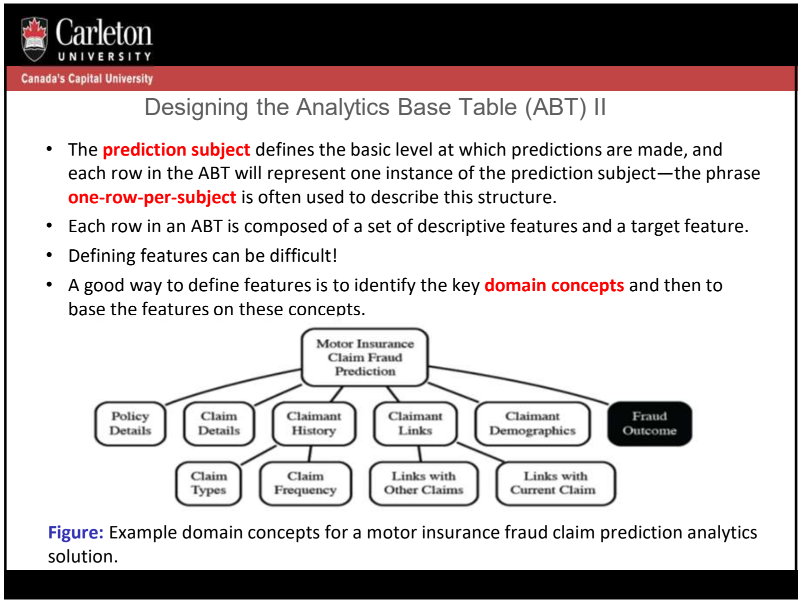
Designing the Analytics Base Table (ABT) I

The basic structure in which we capture historical datasets is the **analytics base table** (**ABT**)

Table: The general structure of an **analytics base table**—descriptive features and a target feature.

Figure: The different data sources typically combined to create an analytics base table.

9

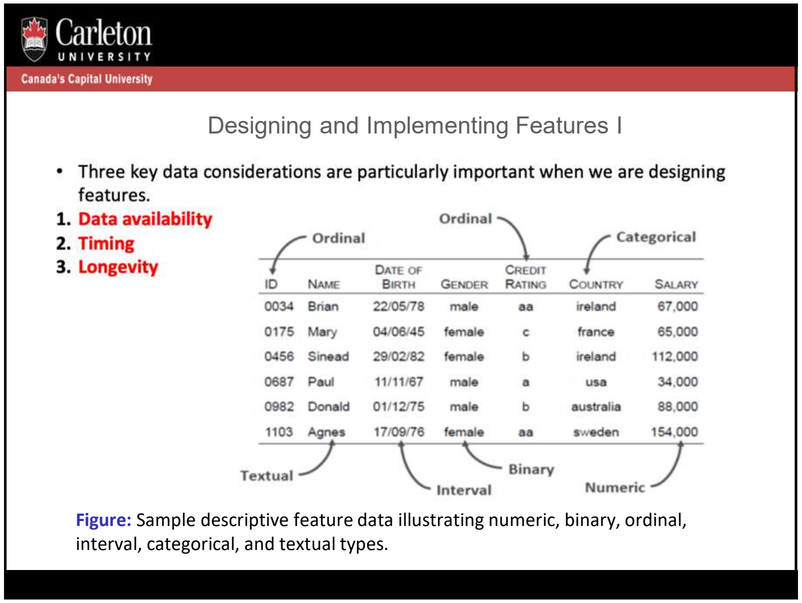


Designing the Analytics Base Table (ABT) II

- The **prediction subject** defines the basic level at which predictions are made, and each row in the ABT will represent one instance of the prediction subject—the phrase **one-row-per-subject** is often used to describe this structure.
- Each row in an ABT is composed of a set of descriptive features and a target feature.
- Defining features can be difficult!
- A good way to define features is to identify the key **domain concepts** and then to base the features on these concepts.

Figure: Example domain concepts for a motor insurance fraud claim prediction analytics solution.

10



Designing and Implementing Features I

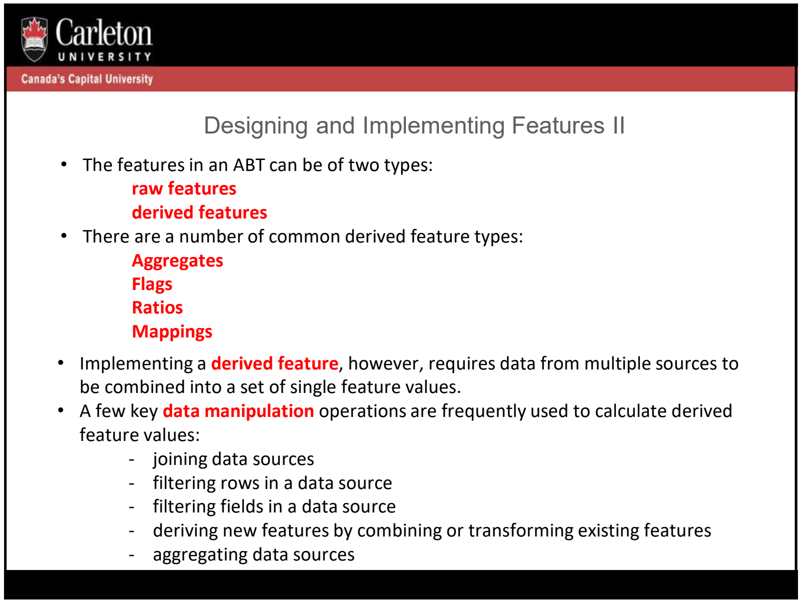
- Three key data considerations are particularly important when we are designing features.

1. **Data availability**
2. **Timing**
3. **Longevity**

ID	NAME	DATE OF BIRTH	GENDER	CREDIT RATING	COUNTRY	SALARY
0034	Brian	22/05/78	male	aa	ireland	67,000
0175	Mary	04/06/45	female	c	france	65,000
0456	Sinead	29/02/82	female	b	ireland	112,000
0687	Paul	11/11/67	male	a	usa	34,000
0982	Donald	01/12/75	male	b	australia	88,000
1103	Agnes	17/09/76	female	aa	sweden	154,000

Figure: Sample descriptive feature data illustrating numeric, binary, ordinal, interval, categorical, and textual types.

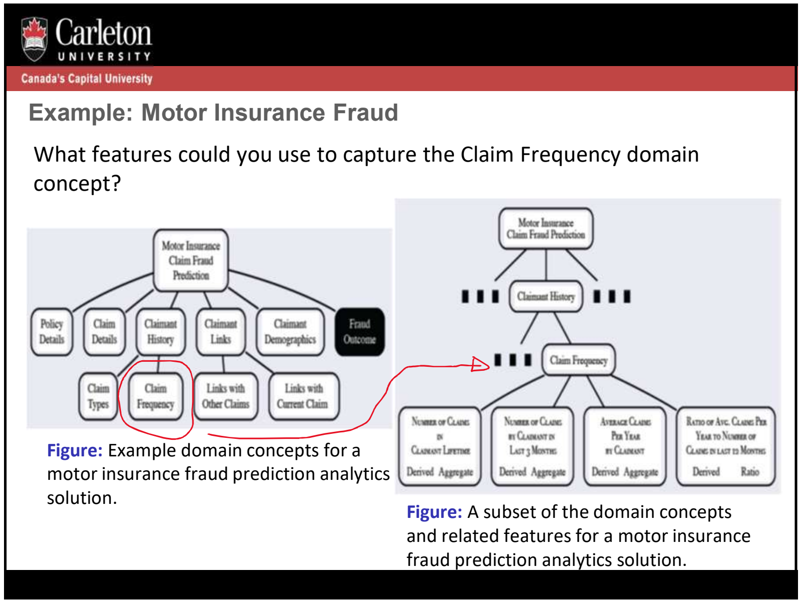
11



Designing and Implementing Features II

- The features in an ABT can be of two types:
  - **raw features**
  - **derived features**
- There are a number of common derived feature types:
  - **Aggregates**
  - **Flags**
  - **Ratios**
  - **Mappings**
- Implementing a **derived feature**, however, requires data from multiple sources to be combined into a set of single feature values.
- A few key **data manipulation** operations are frequently used to calculate derived feature values:
  - joining data sources
  - filtering rows in a data source
  - filtering fields in a data source
  - deriving new features by combining or transforming existing features
  - aggregating data sources

12



Example: Motor Insurance Fraud

What features could you use to capture the Claim Frequency domain concept?

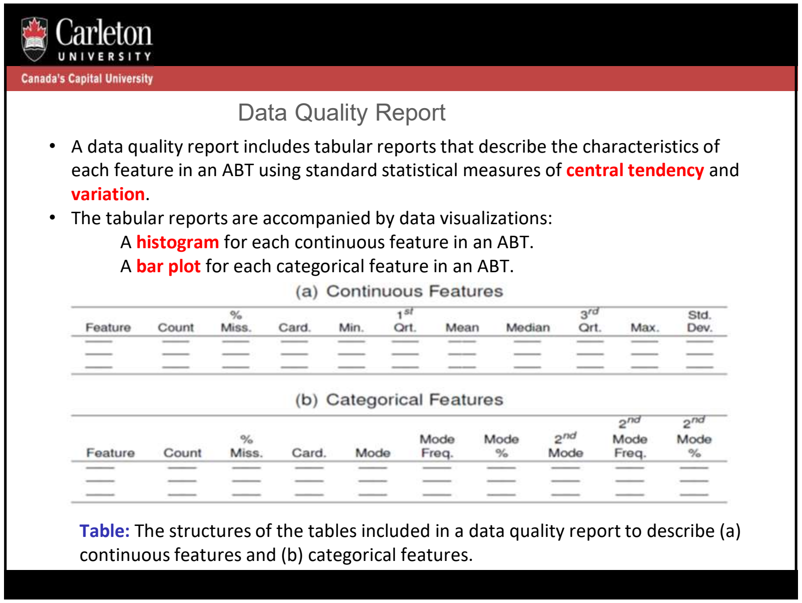
Motor Insurance Claim Fraud Prediction
Policy Details
Claim Details
Claimant History
Claimant Links
Claimant Demographics
Fraud Outcome
Claim Types
Claim Frequency
Links with Other Claims
Links with Current Claim

Figure: Example domain concepts for a motor insurance fraud prediction analytics solution.

Motor Insurance Claim Fraud Prediction
Claimant History
Claim Frequency
Number of Claims in Claimant Lifetime
Derived Aggregate
Number of Claims by Claimant in Last 3 Months
Derived Aggregate
Average Claims Per Year by Claimant
Derived Aggregate
Ratio of Avg. Claims Per Year to Number of Claims in Last 12 Months
Derived Ratio

Figure: A subset of the domain concepts and related features for a motor insurance fraud prediction analytics solution.

13



Data Quality Report

- A data quality report includes tabular reports that describe the characteristics of each feature in an ABT using standard statistical measures of **central tendency** and **variation**.
- The tabular reports are accompanied by data visualizations:
  A **histogram** for each continuous feature in an ABT.
  A **bar plot** for each categorical feature in an ABT.

(a) Continuous Features

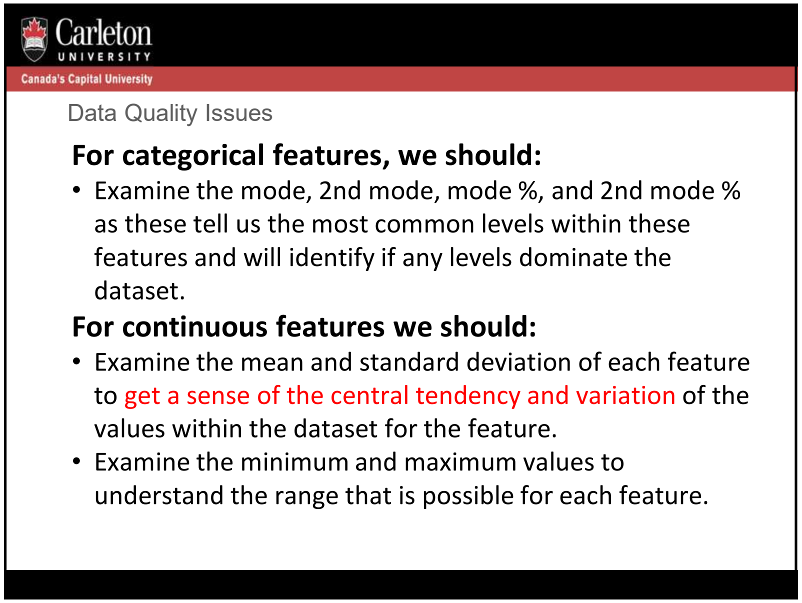
		%			1st			3rd		Std.
Feature	Count	Miss.	Card.	Min.	Qrt.	Mean	Median	Qrt.	Max.	Dev.
—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—

(b) Categorical Features

								2nd	2nd
		%			Mode	Mode	2nd	Mode	Mode
Feature	Count	Miss.	Card.	Mode	Freq.	%	Mode	Freq.	%
—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—

Table: The structures of the tables included in a data quality report to describe (a) continuous features and (b) categorical features.

14



Data Quality Issues

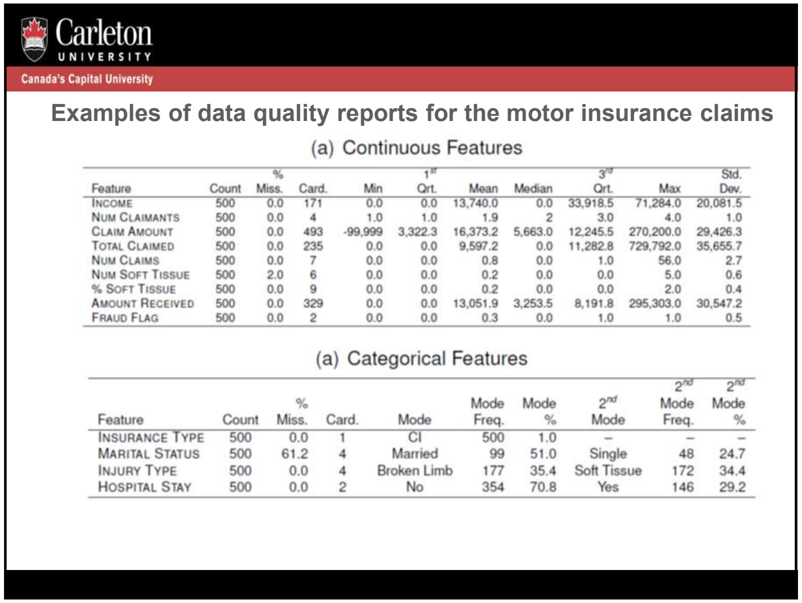
For categorical features, we should:

- Examine the mode, 2nd mode, mode %, and 2nd mode % as these tell us the most common levels within these features and will identify if any levels dominate the dataset.

For continuous features we should:

- Examine the mean and standard deviation of each feature to get a sense of the central tendency and variation of the values within the dataset for the feature.
- Examine the minimum and maximum values to understand the range that is possible for each feature.

15



Examples of data quality reports for the motor insurance claims

(a) Continuous Features

		%			1st			3rd		Std.
Feature	Count	Miss.	Card.	Min	Qrt.	Mean	Median	Qrt.	Max	Dev.
INCOME	500	0.0	171	0.0	0.0	13,740.0	0.0	33,918.5	71,284.0	20,081.5
NUM CLAIMANTS	500	0.0	4	1.0	1.0	1.9	2	3.0	4.0	1.0
CLAIM AMOUNT	500	0.0	493	-99,999	3,322.3	16,373.2	5,663.0	12,245.5	270,200.0	29,426.3
TOTAL CLAIMED	500	0.0	235	0.0	0.0	9,597.2	0.0	11,282.8	729,792.0	35,655.7
NUM CLAIMS	500	0.0	7	0.0	0.0	0.8	0.0	1.0	56.0	2.7
NUM SOFT TISSUE	500	2.0	6	0.0	0.0	0.2	0.0	0.0	5.0	0.6
% SOFT TISSUE	500	0.0	9	0.0	0.0	0.2	0.0	0.0	2.0	0.4
AMOUNT RECEIVED	500	0.0	329	0.0	0.0	13,051.9	3,253.5	8,191.8	295,303.0	30,547.2
FRAUD FLAG	500	0.0	2	0.0	0.0	0.3	0.0	1.0	1.0	0.5

(a) Categorical Features

								2nd	2nd
		%			Mode	Mode	2nd	Mode	Mode
Feature	Count	Miss.	Card.	Mode	Freq.	%	Mode	Freq.	%
INSURANCE TYPE	500	0.0	1	CI	500	1.0	–	–	–
MARITAL STATUS	500	61.2	4	Married	99	51.0	Single	48	24.7
INJURY TYPE	500	0.0	4	Broken Limb	177	35.4	Soft Tissue	172	34.4
HOSPITAL STAY	500	0.0	2	No	354	70.8	Yes	146	29.2

16

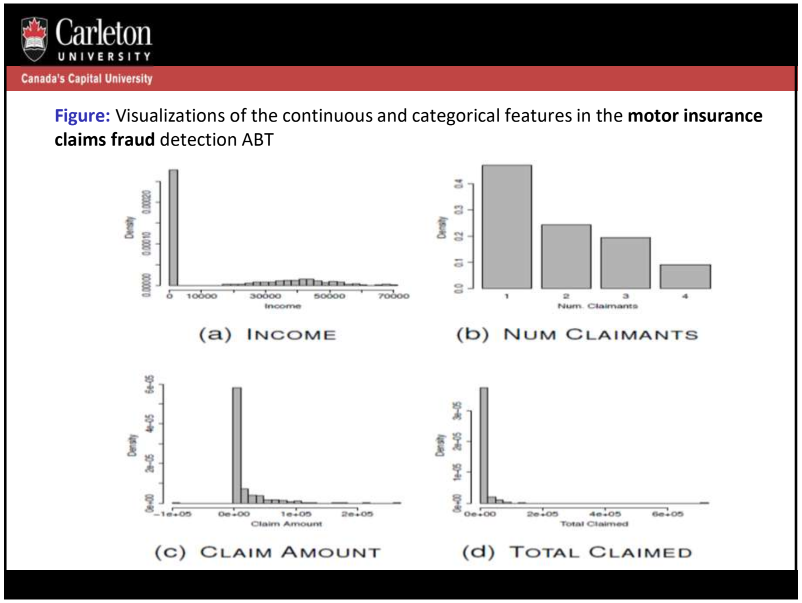


Figure: Visualizations of the continuous and categorical features in the **motor insurance claims fraud** detection ABT

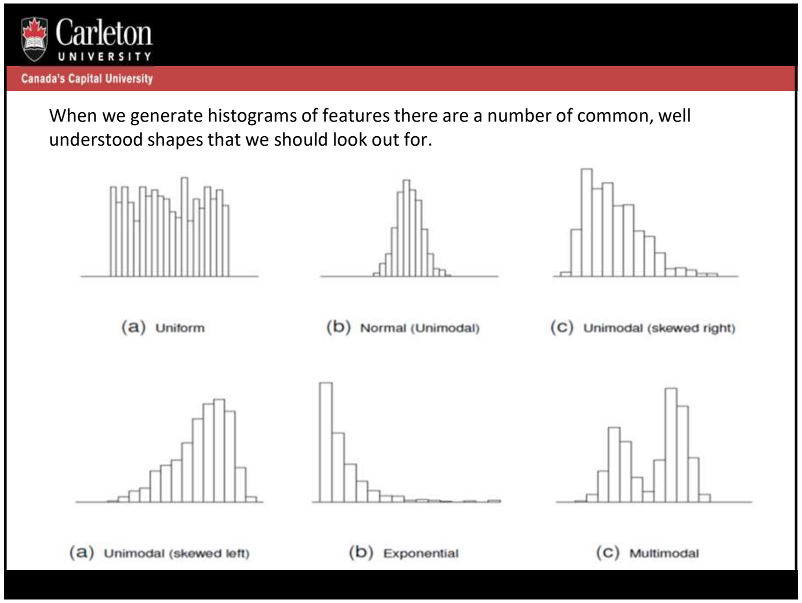
(a) Income

(b) Num Claimants

(c) Claim Amount

(d) Total Claimed

17



When we generate histograms of features there are a number of common, well understood shapes that we should look out for.

(a) Uniform

(b) Normal (Unimodal)

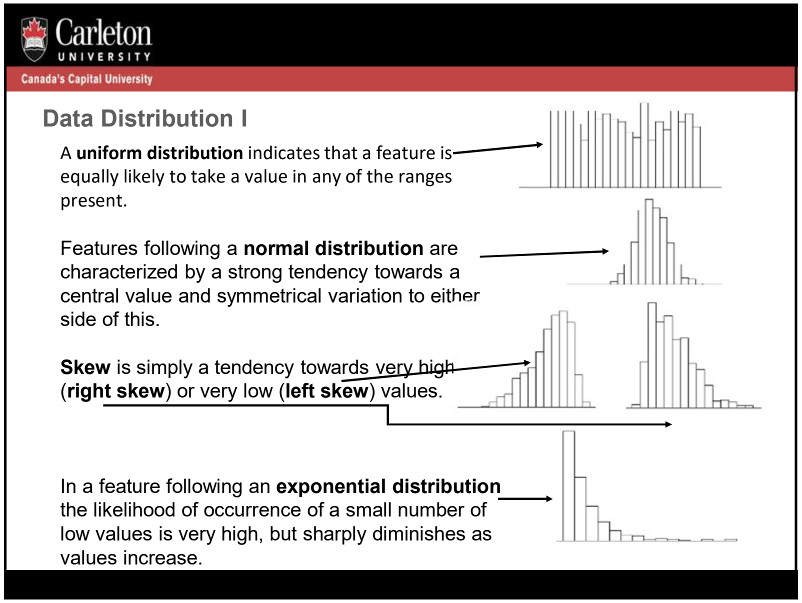
(c) Unimodal (skewed right)

(a) Unimodal (skewed left)

(b) Exponential

(c) Multimodal

18



Data Distribution I

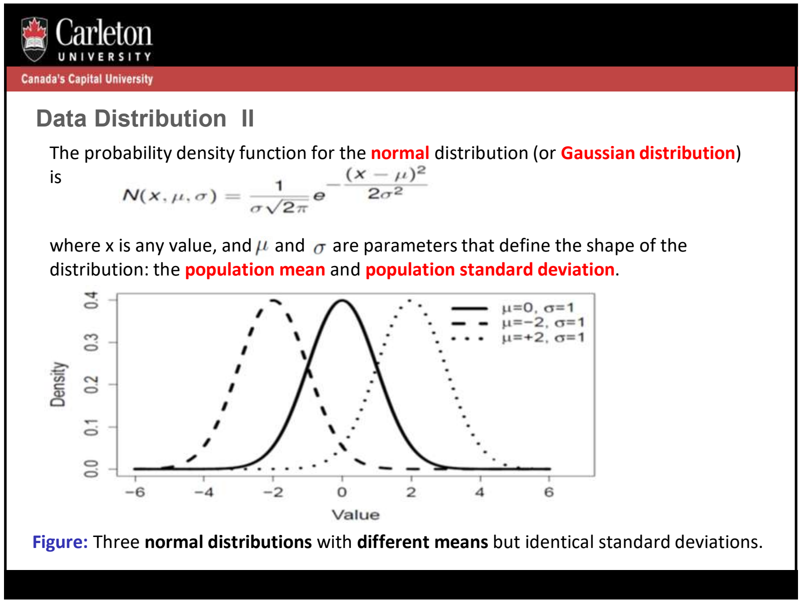
A **uniform distribution** indicates that a feature is equally likely to take a value in any of the ranges present.

Features following a **normal distribution** are characterized by a strong tendency towards a central value and symmetrical variation to either side of this.

Skew is simply a tendency towards very high (**right skew**) or very low (**left skew**) values.

In a feature following an **exponential distribution** the likelihood of occurrence of a small number of low values is very high, but sharply diminishes as values increase.

19



Data Distribution II

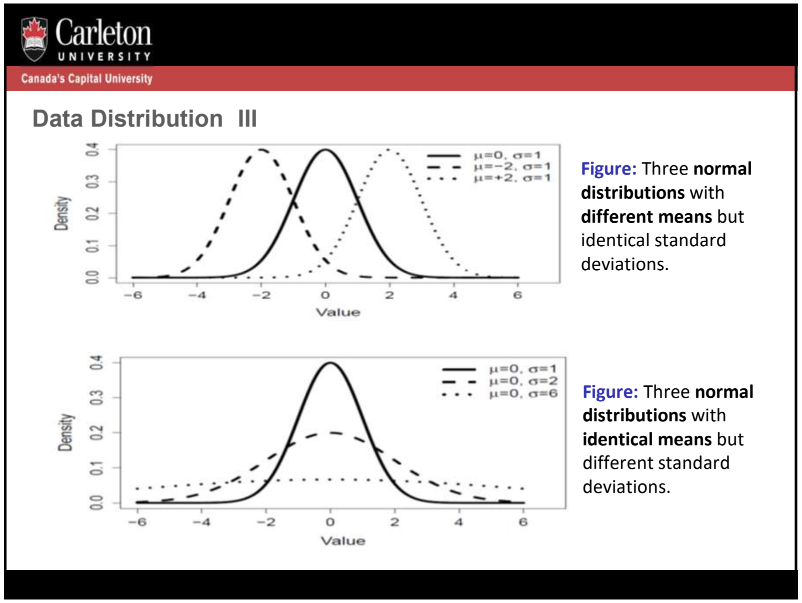
The probability density function for the **normal** distribution (or **Gaussian distribution**) is

$$N(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where x is any value, and μ and σ are parameters that define the shape of the distribution: the **population mean** and **population standard deviation**.

Figure: Three **normal distributions** with **different means** but identical standard deviations.

20



Carleton UNIVERSITY

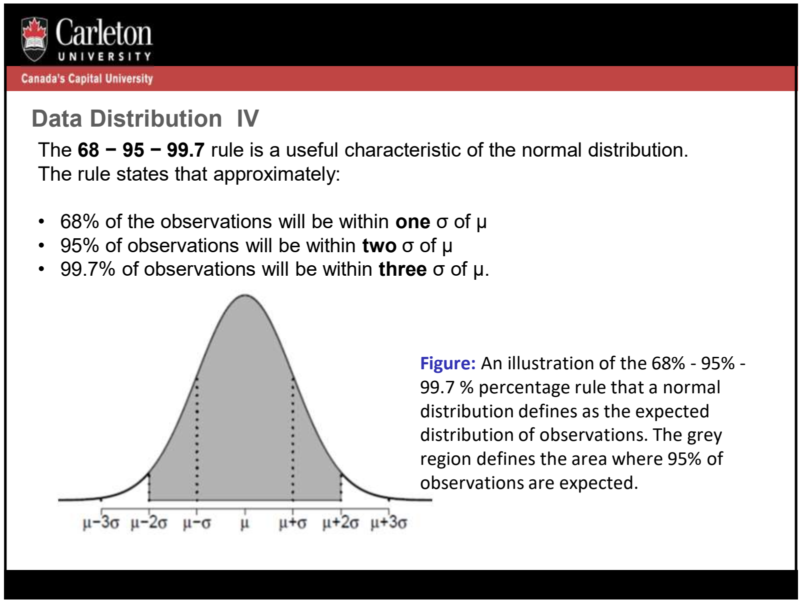
Canada's Capital University

Data Distribution III

Figure: Three **normal distributions** with **different means** but identical standard deviations.

Figure: Three **normal distributions** with **identical means** but different standard deviations.

21



Carleton UNIVERSITY

Canada's Capital University

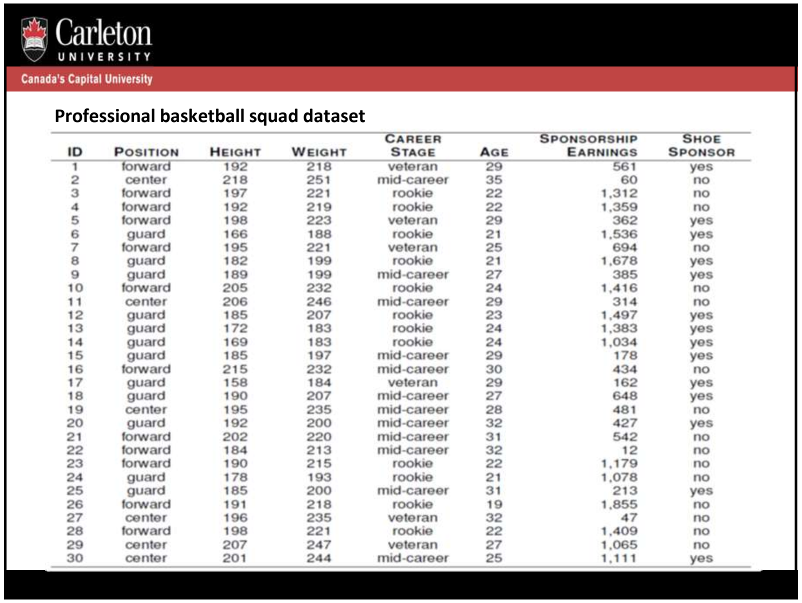
Data Distribution IV

The **68 − 95 − 99.7** rule is a useful characteristic of the normal distribution. The rule states that approximately:

- 68% of the observations will be within **one** σ of μ
- 95% of observations will be within **two** σ of μ
- 99.7% of observations will be within **three** σ of μ.

Figure: An illustration of the 68% - 95% - 99.7 % percentage rule that a normal distribution defines as the expected distribution of observations. The grey region defines the area where 95% of observations are expected.

22



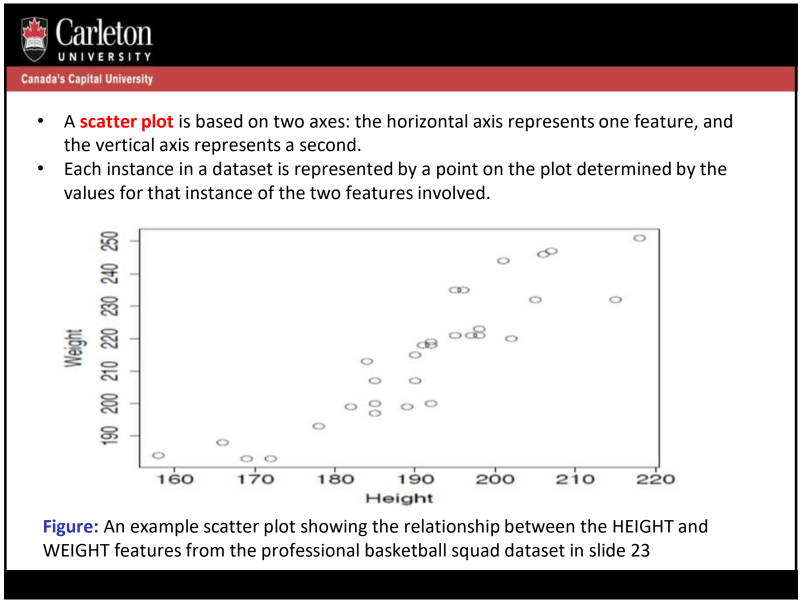
Carleton UNIVERSITY

Canada's Capital University

Professional basketball squad dataset

ID	POSITION	HEIGHT	WEIGHT	CAREER STAGE	AGE	SPONSORSHIP EARNINGS	SHOE SPONSOR
1	forward	192	218	veteran	29	561	yes
2	center	218	251	mid-career	35	60	no
3	forward	197	221	rookie	22	1,312	no
4	forward	192	219	rookie	22	1,359	no
5	forward	198	223	veteran	29	362	yes
6	guard	166	188	rookie	21	1,536	yes
7	forward	195	221	veteran	25	694	no
8	guard	182	199	rookie	21	1,678	yes
9	guard	189	199	mid-career	27	385	yes
10	forward	205	232	rookie	24	1,416	no
11	center	206	246	mid-career	29	314	no
12	guard	185	207	rookie	23	1,497	yes
13	guard	172	183	rookie	24	1,383	yes
14	guard	169	183	rookie	24	1,034	yes
15	guard	185	197	mid-career	29	178	yes
16	forward	215	232	mid-career	30	434	no
17	guard	158	184	veteran	29	162	yes
18	guard	190	207	mid-career	27	648	yes
19	center	195	235	mid-career	28	481	no
20	guard	192	200	mid-career	32	427	yes
21	forward	202	220	mid-career	31	542	no
22	forward	184	213	mid-career	32	12	no
23	forward	190	215	rookie	22	1,179	no
24	guard	178	193	rookie	21	1,078	no
25	guard	185	200	mid-career	31	213	yes
26	forward	191	218	rookie	19	1,855	no
27	center	196	235	veteran	32	47	no
28	forward	198	221	rookie	22	1,409	no
29	center	207	247	veteran	27	1,065	no
30	center	201	244	mid-career	25	1,111	yes

23



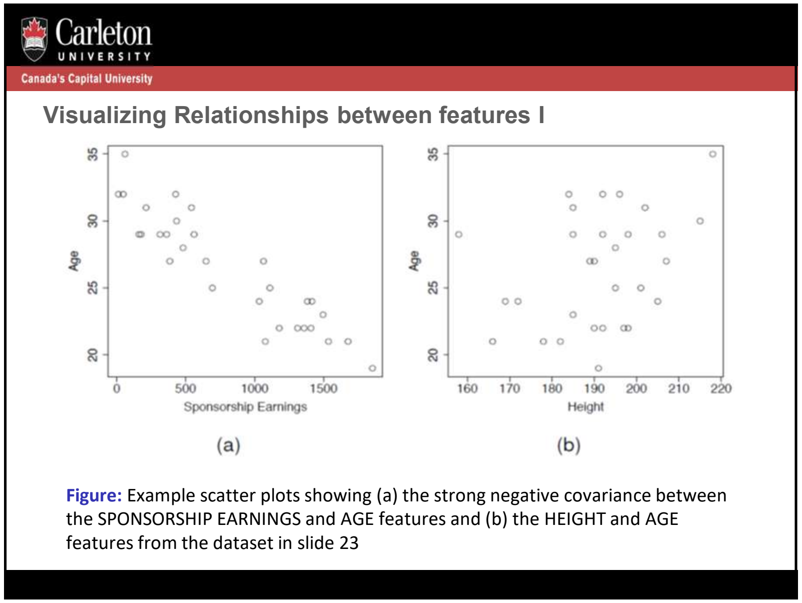
Carleton UNIVERSITY

Canada's Capital University

- A **scatter plot** is based on two axes: the horizontal axis represents one feature, and the vertical axis represents a second.
- Each instance in a dataset is represented by a point on the plot determined by the values for that instance of the two features involved.

Figure: An example scatter plot showing the relationship between the HEIGHT and WEIGHT features from the professional basketball squad dataset in slide 23

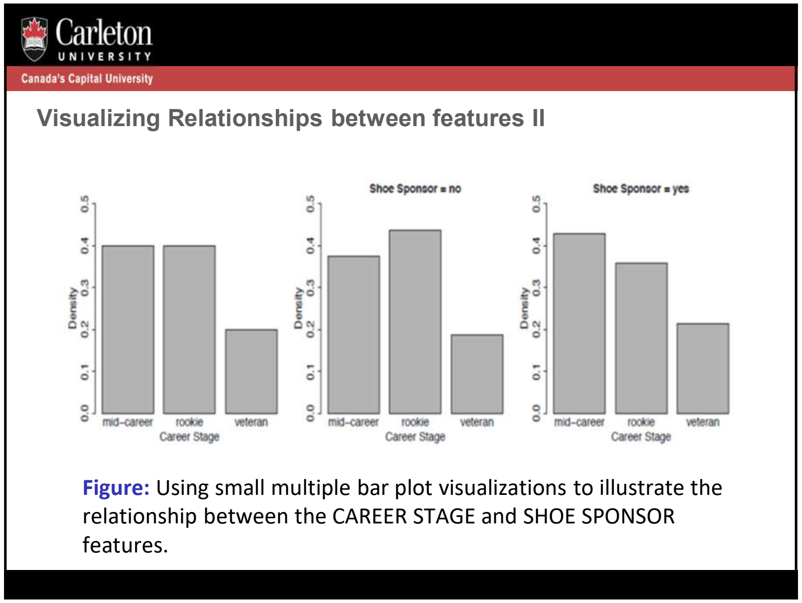
24



Visualizing Relationships between features I

Figure: Example scatter plots showing (a) the strong negative covariance between the SPONSORSHIP EARNINGS and AGE features and (b) the HEIGHT and AGE features from the dataset in slide 23

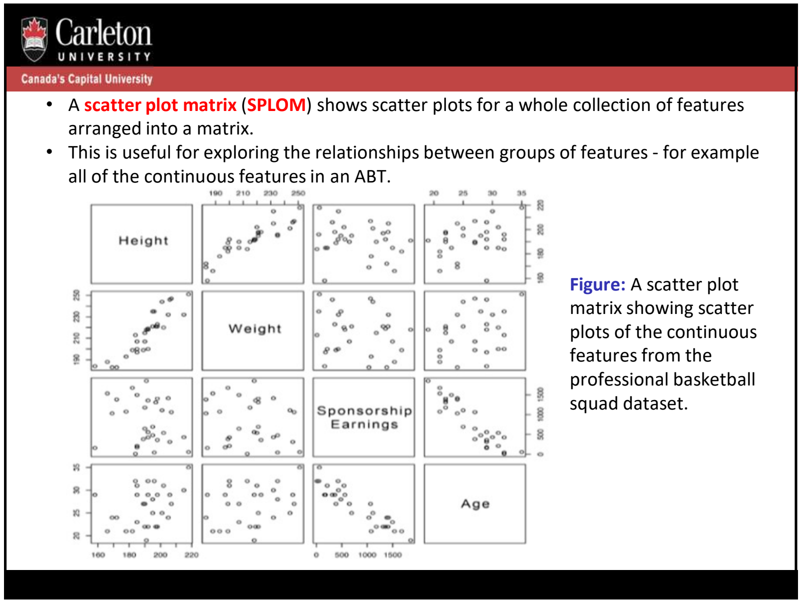
25



Visualizing Relationships between features II

Figure: Using small multiple bar plot visualizations to illustrate the relationship between the CAREER STAGE and SHOE SPONSOR features.

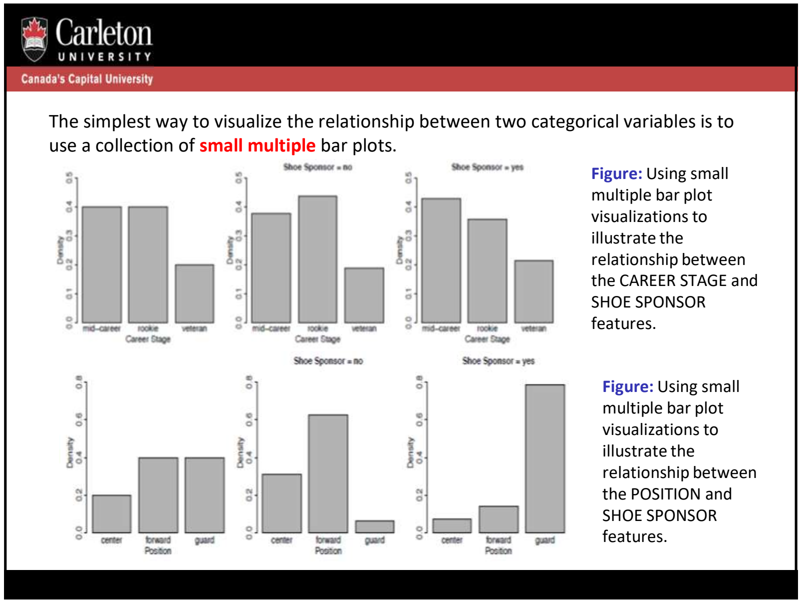
26



- A **scatter plot matrix** (**SPLOM**) shows scatter plots for a whole collection of features arranged into a matrix.
- This is useful for exploring the relationships between groups of features - for example all of the continuous features in an ABT.

Figure: A scatter plot matrix showing scatter plots of the continuous features from the professional basketball squad dataset.

27

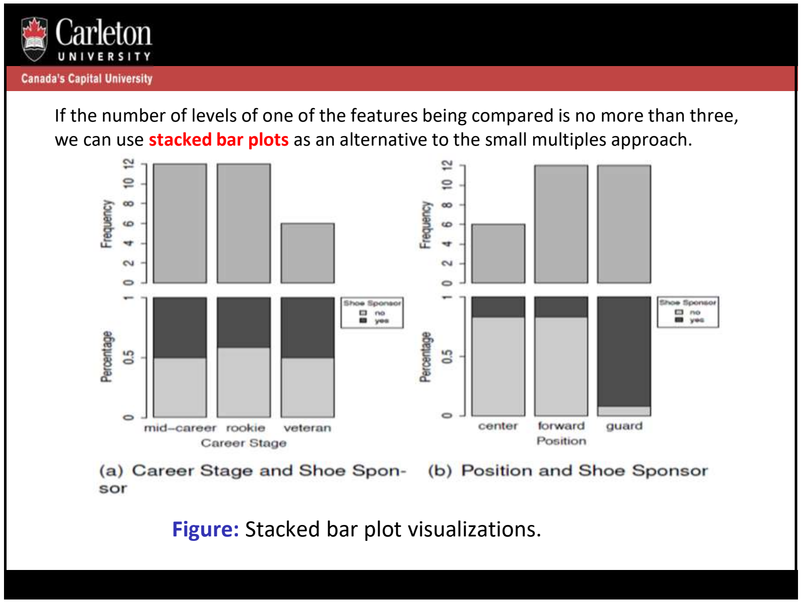


The simplest way to visualize the relationship between two categorical variables is to use a collection of **small multiple** bar plots.

Figure: Using small multiple bar plot visualizations to illustrate the relationship between the CAREER STAGE and SHOE SPONSOR features.

Figure: Using small multiple bar plot visualizations to illustrate the relationship between the POSITION and SHOE SPONSOR features.

28

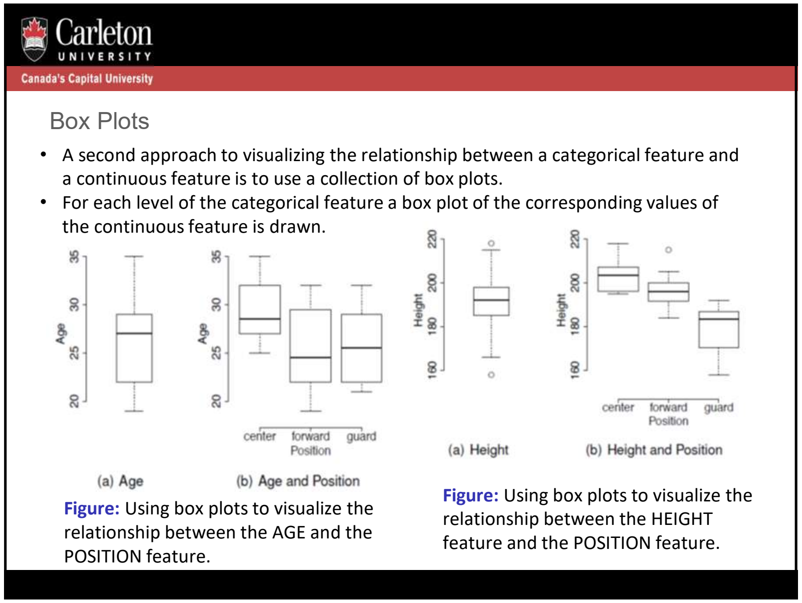


If the number of levels of one of the features being compared is no more than three, we can use **stacked bar plots** as an alternative to the small multiples approach.

(a) Career Stage and Shoe Sponsor (b) Position and Shoe Sponsor

Figure: Stacked bar plot visualizations.

29



Box Plots

- A second approach to visualizing the relationship between a categorical feature and a continuous feature is to use a collection of box plots.
- For each level of the categorical feature a box plot of the corresponding values of the continuous feature is drawn.

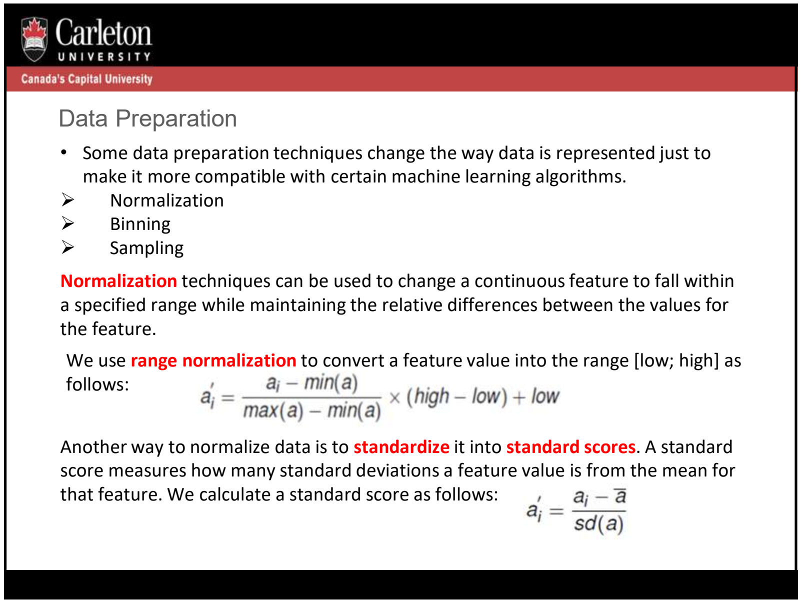
(a) Age (b) Age and Position

Figure: Using box plots to visualize the relationship between the AGE and the POSITION feature.

(a) Height (b) Height and Position

Figure: Using box plots to visualize the relationship between the HEIGHT feature and the POSITION feature.

30



Data Preparation

- Some data preparation techniques change the way data is represented just to make it more compatible with certain machine learning algorithms.
  - ➢ Normalization
  - ➢ Binning
  - ➢ Sampling

Normalization techniques can be used to change a continuous feature to fall within a specified range while maintaining the relative differences between the values for the feature.

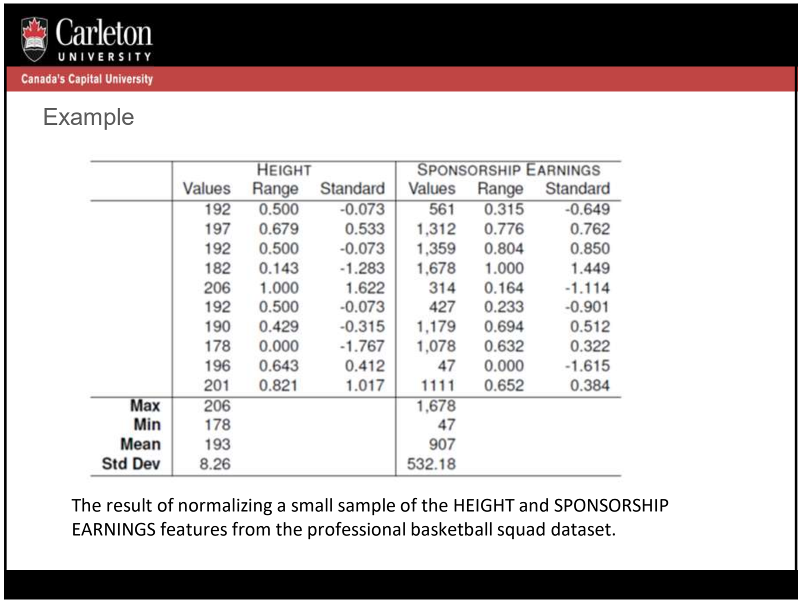
We use **range normalization** to convert a feature value into the range [low; high] as follows:

$$a_i^{'} = \frac{a_i - min(a)}{max(a) - min(a)} \times (high - low) + low$$

Another way to normalize data is to **standardize** it into **standard scores**. A standard score measures how many standard deviations a feature value is from the mean for that feature. We calculate a standard score as follows:

$$a_i^{'} = \frac{a_i - \overline{a}}{sd(a)}$$

31

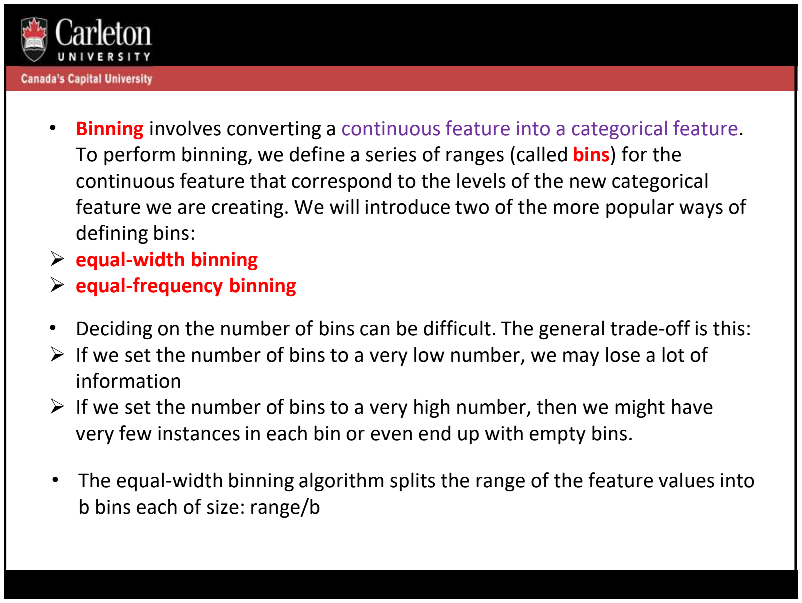


Example

	HEIGHT			SPONSORSHIP EARNINGS		
	Values	Range	Standard	Values	Range	Standard
	192	0.500	-0.073	561	0.315	-0.649
	197	0.679	0.533	1,312	0.776	0.762
	192	0.500	-0.073	1,359	0.804	0.850
	182	0.143	-1.283	1,678	1.000	1.449
	206	1.000	1.622	314	0.164	-1.114
	192	0.500	-0.073	427	0.233	-0.901
	190	0.429	-0.315	1,179	0.694	0.512
	178	0.000	-1.767	1,078	0.632	0.322
	196	0.643	0.412	47	0.000	-1.615
	201	0.821	1.017	1111	0.652	0.384
Max	206			1,678		
Min	178			47		
Mean	193			907		
Std Dev	8.26			532.18		

The result of normalizing a small sample of the HEIGHT and SPONSORSHIP EARNINGS features from the professional basketball squad dataset.

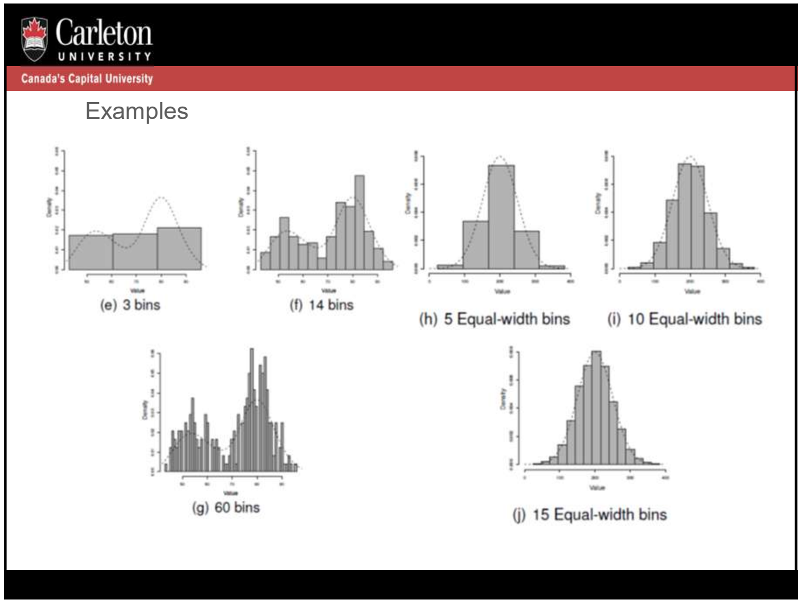
32



- **Binning** involves converting a continuous feature into a categorical feature. To perform binning, we define a series of ranges (called **bins**) for the continuous feature that correspond to the levels of the new categorical feature we are creating. We will introduce two of the more popular ways of defining bins:
- ➢ **equal-width binning**
- ➢ **equal-frequency binning**

- Deciding on the number of bins can be difficult. The general trade-off is this:
- ➢ If we set the number of bins to a very low number, we may lose a lot of information
- ➢ If we set the number of bins to a very high number, then we might have very few instances in each bin or even end up with empty bins.

- The equal-width binning algorithm splits the range of the feature values into b bins each of size: range/b



Examples

(e) 3 bins

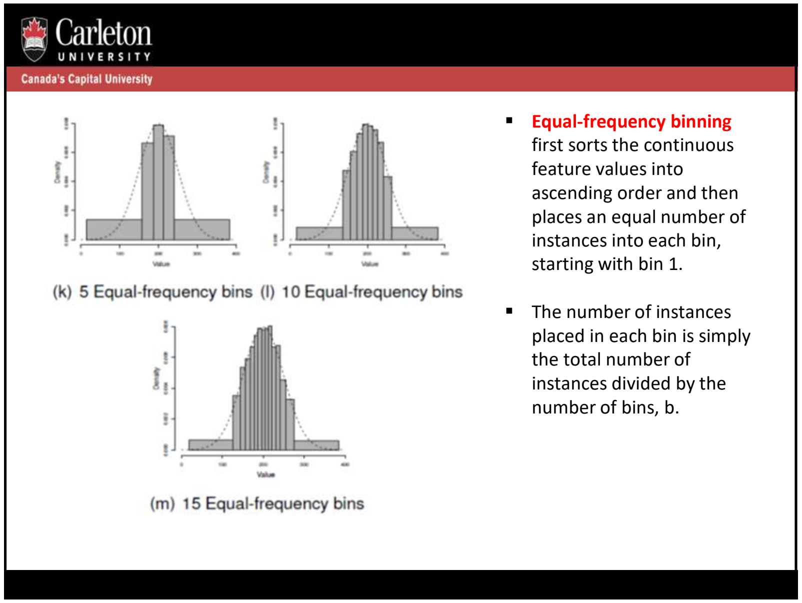
(f) 14 bins

(h) 5 Equal-width bins

(i) 10 Equal-width bins

(g) 60 bins

(j) 15 Equal-width bins

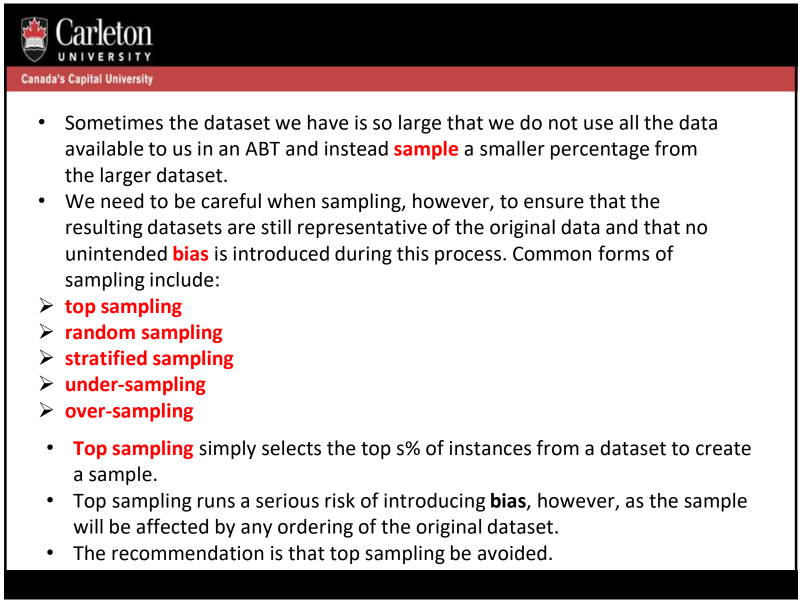


(k) 5 Equal-frequency bins (l) 10 Equal-frequency bins

(m) 15 Equal-frequency bins

- **Equal-frequency binning** first sorts the continuous feature values into ascending order and then places an equal number of instances into each bin, starting with bin 1.

- The number of instances placed in each bin is simply the total number of instances divided by the number of bins, b.



- Sometimes the dataset we have is so large that we do not use all the data available to us in an ABT and instead **sample** a smaller percentage from the larger dataset.
- We need to be careful when sampling, however, to ensure that the resulting datasets are still representative of the original data and that no unintended **bias** is introduced during this process. Common forms of sampling include:
- ➢ **top sampling**
- ➢ **random sampling**
- ➢ **stratified sampling**
- ➢ **under-sampling**
- ➢ **over-sampling**

- **Top sampling** simply selects the top s% of instances from a dataset to create a sample.
- Top sampling runs a serious risk of introducing **bias**, however, as the sample will be affected by any ordering of the original dataset.
- The recommendation is that top sampling be avoided.

- The recommended default, **random sampling** randomly selects a proportion of s% of the instances from a large dataset to create a smaller set. Random sampling is a good choice in most cases as the random nature of the selection of instances should avoid introducing bias.

- **Stratified sampling** is a sampling method that ensures that the relative frequencies of the levels of a specific **stratification feature** are maintained in the sampled dataset.
- To perform stratified sampling:

1. the instances in a dataset are divided into groups (or strata), where each group contains only instances that have a particular level for the stratification feature
2. s% of the instances in each stratum are randomly selected these selections are combined to give an overall sample of s% of the original dataset.

- In contrast to stratified sampling, sometimes we would like a sample to contain different relative frequencies of the levels of a particular feature to the distribution in the original dataset. To do this, we can use **under-sampling** or **over-sampling**.

- **Under-sampling** begins by dividing a dataset into groups, where each group contains only instances that have a particular level for the feature to be under-sampled.
- The number of instances in the smallest group is the under-sampling target size.
- Each group containing more instances than the smallest one is then randomly sampled by the appropriate percentage to create a subset that is the under-sampling target size.
- These under-sampled groups are then combined to create the overall under-sampled dataset.

- **Over-sampling** addresses the same issue as under-sampling but in the opposite way around.
- After dividing the dataset into groups, the number of instances in the largest group becomes the over-sampling target size.
- From each smaller group, we then create a sample containing that number of instances using **random sampling with replacement**.
- These larger samples are combined to form the overall over-sampled dataset.

Dimensionality Reduction

- Many ML problems involve thousands or millions (even tens of millions) of features for each training instance.
- This can make training extremely slow and good solution difficult to find.
- This is often referred to as *curse of dimensionality*.
- Reducing dimensionality may cause some information loss, even though it will speed up training.
- It is also useful for data visualization

The Curse of Dimensionality I

- We are generally used to 3D and our intuition fails us when we imagine high-dimensional space.
- For example, a basic 4D hypercube is very hard to picture in our minds, let alone 200D ellipsoid!

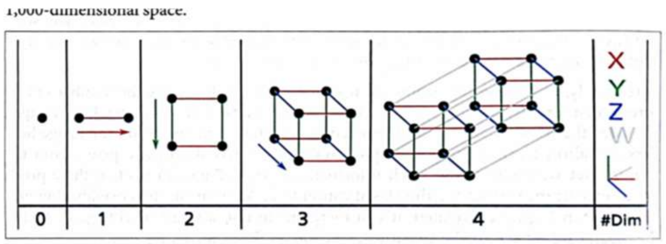


Figure 8-1. Point, segment, square, cube, and tesseract (0D to 4D hypercubes)[2]

The Curse of Dimensionality II

- For example:
  - If you pick 2 points randomly in a unit square, the avg distance between these points will be roughly 0.52.
  - If you pick 2 random points in a unit 3D cube, the avg distance will be roughly 0.66.
  - But 2 points pick randomly in a 1,000,000D hypercube, the avg distance will be about 408.25!
- The question then is: how can 2 points be so far away when they both lie in the same unit hypercube?
- Well, there is plenty of space in high-dimension and, as a result, high-dimensional datasets are at risk of being very sparse.
- In theory, the solution to curse of dimensionality is to increase the size of the training set to reach a sufficient density of training instances. However, in practice, this may not work because of the exponential growth in density with higher number of dimensions.

PCA – Principal Component Analysis

- PCA is the oldest and most popular dimensionality reduction algorithm.
- PCA first identifies the hyperplane that lies closest to the data, then projects the data onto it (see fig below)

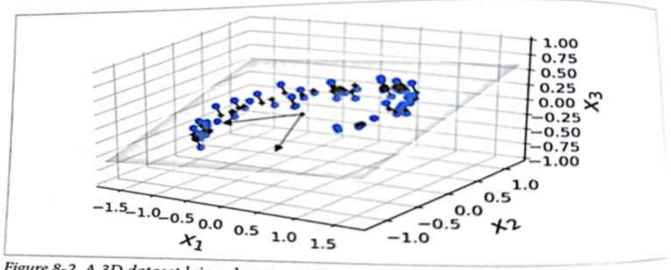


Figure 8-2. A 3D dataset lying close to a 2D subspace

Preserving the Variance

- The right hyperplane is needed for projecting the training set onto a lower dimensional hyperplane.
- For example, a simple 2D dataset is represented on the left of fig below with three different axes – 1D hyperplanes.

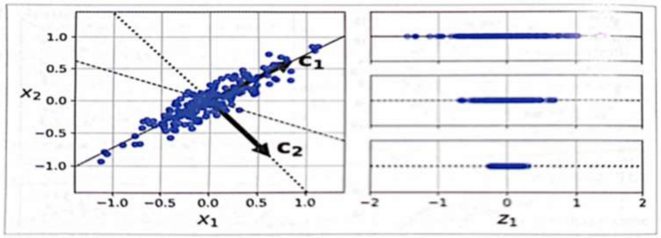


Figure 8-7. Selecting the subspace to project on

- On the right – the result of projecting onto each of these axes. The solid preserves the maximum variance while projection onto dotted lines preserves very little variance and dashed line preserves intermediate amount of variance.

Different types of PCA

- Scikit-Learn uses a stochastic algorithm called **Randomized PCA** that can quickly find an approximation for the first set of d principal components.
- PCA requires the whole training data to reside in memory in order for the algorithm to run**.** **Incremental PCA** was developed to allow for splitting the training set into mini-batches.
- **Kernel PCA (kPCA)** – is used to perform complex nonlinear projection for dimensionality reduction.

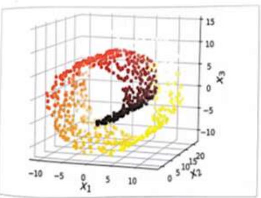


Figure 8-4. Swiss roll dataset

Original Swiss roll Dataset

(equivalent to simply using the PCA class), an RBF kernel, and a sigmoid kernel.

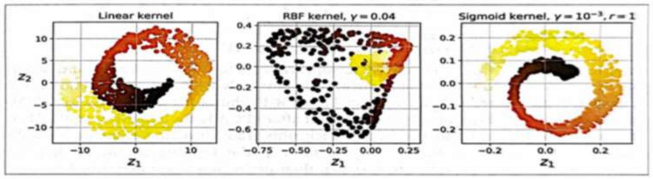


Figure 8-10. Swiss roll reduced to 2D using kPCA with various kernels

Kernel PCA and Reconstructing Pre-Image Error

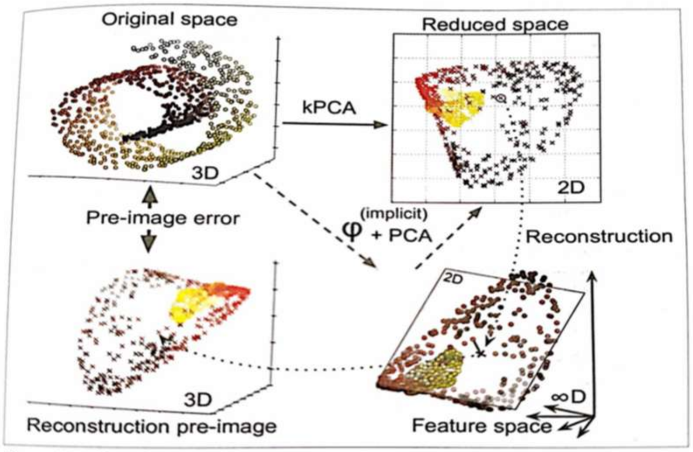


Figure 8-11. Kernel PCA and the reconstruction pre-image error

Other Dimensionality Reduction Techniques

- Random Projections
- Multidimensional Scaling (MDS)
- Isomap
- t-Distribution Stochastic Neighbor Embedding (t-SNE)
- Linear Discriminant Analysis (LDA), etc.

Figure 8-13 shows the results of a few of these techniques.

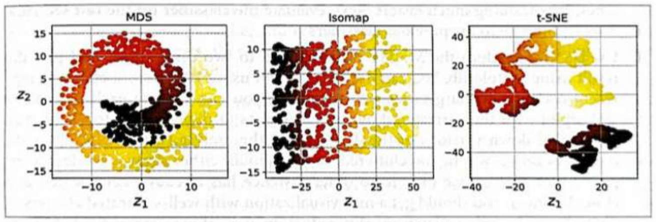
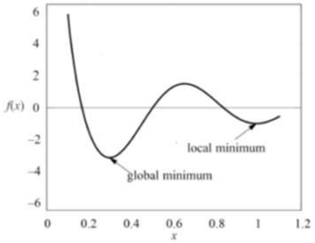


Figure 8-13. Using various techniques to reduce the Swill roll to 2D

Notation and Definition I

- A **scalar** is a simple numerical value, like 15 or−3.25.
- Variables or constants that take scalar values are denoted by an italic letter, like *x* or *a*.
- A **vector** is an ordered list of scalar values, called attributes. We denote a vector as a bold character, for example, **x** or **w**. Vectors can be visualized as arrows that point to some directions as well as points in a multi-dimensional space
- **Matrices** are denoted with bold capital letters, such as **A** or **W**.
- A **function** is a relation that associates each element *x* of a set *X*, the **domain** of the function, to a single element *y* of another set *Y*, the **codomain** of the function.
- We say that $f(x)$ has a **local minimum** at $x = c$ if $f(x) \geq f(c)$ for every *x* in some open interval around $x = c$. An **interval** is a set of real numbers with the property that any number that lies between two numbers in the set is also included in the set. An open interval does not include its endpoints and is denoted using parentheses. For example, (0,1) means "all numbers greater than 0 and less than 1". The minimal value among all the local minima is called the **global minimum**.

Notation and Definition II



A local and a global minima of a function

Max and Arg Max

- Given a set of values A = {a_1, a_2, . . . , a_n}, the operator $\max_{a \in A} f(a)$ returns the highest value $f(a)$ for all elements in the set A. On the other hand, the operator arg $\max_{a2A}$ $f(a)$ returns the element of the set *A* that maximizes $f(a)$.
- Sometimes, when the set is implicit or infinite, we can write $\max_a f(a)$ or arg $\max_a$ $f(a)$.
- Operators min and arg min operate in a similar manner.

Bayes' Rule

The conditional probability $\Pr(X = x|Y = y)$ is the probability of the random variable X to have a specific value x given that another random variable Y has a specific value of y. The **Bayes' Rule** (also known as the **Bayes' Theorem**) stipulates that:

$$\Pr(X = x|Y = y) = \frac{\Pr(Y = y|X = x)\Pr(X = x)}{\Pr(Y = y)}.$$

Parameters vs. Hyperparameters

- A **hyperparameter** is a property of a learning algorithm, usually (but not always) having a numerical value. That value influences the way the algorithm works.
- **Hyperparameters** aren't learned by the algorithm itself from data. They have to be set by the data analyst before running the algorithm.
- **Parameters** are variables that define the model learned by the learning algorithm.
- **Parameters** are directly modified by the learning algorithm based on the training data.
- The goal of learning is to find such values of parameters that make the model optimal in a certain sense.

Model Performance I

- One way of getting a good model is to compare different models by calculating a performance metric on the holdout data.
- **Regression Performance Metric:** There are three metrics often used for regression performance - *mean squared error (MSE*), mean absolute error (MAE), and almost correct predictions error rate (ACPER).
- The most used is MSE, defined as, $\mathrm{MSE}(f) \stackrel{\text{def}}{=} \frac{1}{N}\sum_{i=1\ldots N}(f(\mathbf{x}_i) - y_i)^2$ where *f* is the model that takes a feature vector x as input and outputs a prediction, and *i*, ranging from 1 to *N*, denotes the index of an example from a dataset.
- If the data contains outliers, it is better to apply the median absolute error, MdAE: $\mathrm{MdAE} \stackrel{\text{def}}{=} \mathrm{median}\left(\{|f(\mathbf{x}_i) - y_i|\}_{i=1}^{N}\right)$, where $\{|f(\mathbf{x}_i) - y_i|\}_{i=1}^{N}$ denotes the set of absolute error values for all examples, from I = 1 to N, on which the evaluation of the model is performed.
- You may read about how to calculate ACPER metric!

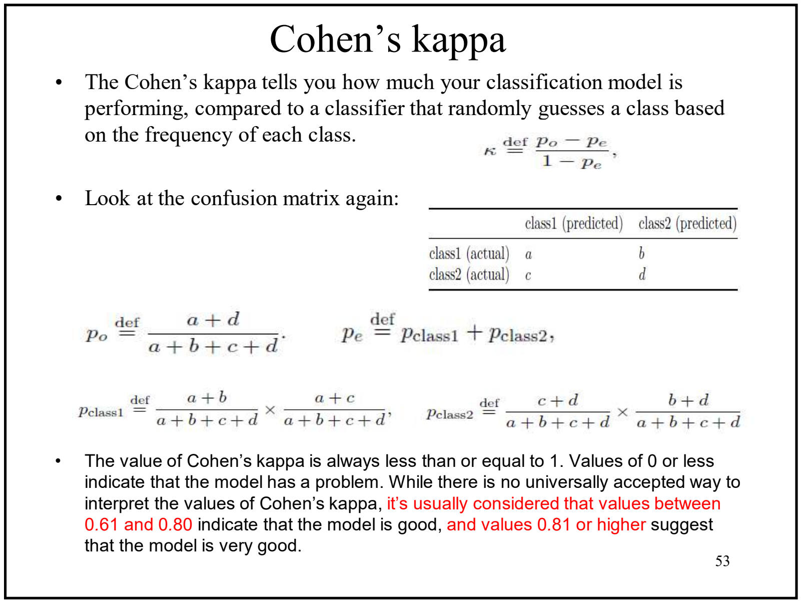
Model Performance II

- Classification Performance Metrics: This is a little bit more complicated, and the metrics used are:
  - Precision-recall
  - Accuracy
  - Cost-sensitive accuracy, and
  - Area under the ROC curve (AUC)
- Confusion Matrix – is a table summarizing how good a classification model is at predicting examples belonging to different classes.

	spam (predicted)	not_spam (predicted)
spam (actual)	23 (TP)	1 (FN)
not_spam (actual)	12 (FP)	556 (TN)

Precision, Recall, F1, and Accuracy

- Precision: $$\text{precision} \stackrel{\text{def}}{=} \frac{\text{TP}}{\text{TP} + \text{FP}}.$$
- Recall: $$\text{recall} \stackrel{\text{def}}{=} \frac{\text{TP}}{\text{TP} + \text{FN}}.$$
- F1: $$\text{F}_1 = \left(\frac{2}{\text{recall}^{-1} + \text{precision}^{-1}}\right) = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$
- Accuracy: $$\text{accuracy} \stackrel{\text{def}}{=} \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$
- Cohen's kappa: $$\kappa \stackrel{\text{def}}{=} \frac{p_o - p_e}{1 - p_e},$$
  - where *po* is called the observed agreement, and *pe* is the expected agreement.



Cohen’s kappa

- The Cohen’s kappa tells you how much your classification model is performing, compared to a classifier that randomly guesses a class based on the frequency of each class.

$$\kappa \overset{\text{def}}{=} \frac{p_o - p_e}{1 - p_e},$$

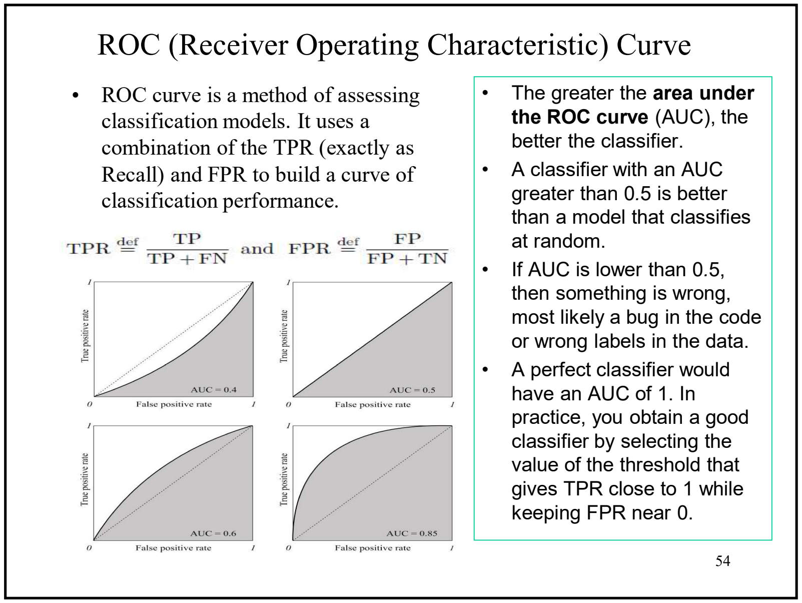
- Look at the confusion matrix again:

	class1 (predicted)	class2 (predicted)
class1 (actual)	a	b
class2 (actual)	c	d

$$p_o \overset{\text{def}}{=} \frac{a+d}{a+b+c+d}. \qquad p_e \overset{\text{def}}{=} p_{\text{class1}} + p_{\text{class2}},$$

$$p_{\text{class1}} \overset{\text{def}}{=} \frac{a+b}{a+b+c+d} \times \frac{a+c}{a+b+c+d}, \qquad p_{\text{class2}} \overset{\text{def}}{=} \frac{c+d}{a+b+c+d} \times \frac{b+d}{a+b+c+d}$$

- The value of Cohen’s kappa is always less than or equal to 1. Values of 0 or less indicate that the model has a problem. While there is no universally accepted way to interpret the values of Cohen’s kappa, it’s usually considered that values between 0.61 and 0.80 indicate that the model is good, and values 0.81 or higher suggest that the model is very good.

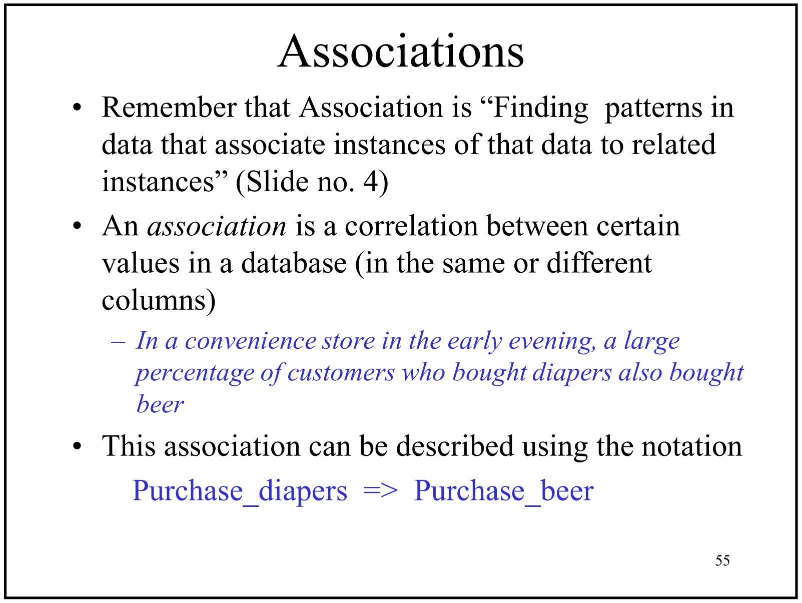


ROC (Receiver Operating Characteristic) Curve

- ROC curve is a method of assessing classification models. It uses a combination of the TPR (exactly as Recall) and FPR to build a curve of classification performance.

$$\text{TPR} \overset{\text{def}}{=} \frac{\text{TP}}{\text{TP}+\text{FN}} \text{ and } \text{FPR} \overset{\text{def}}{=} \frac{\text{FP}}{\text{FP}+\text{TN}}$$

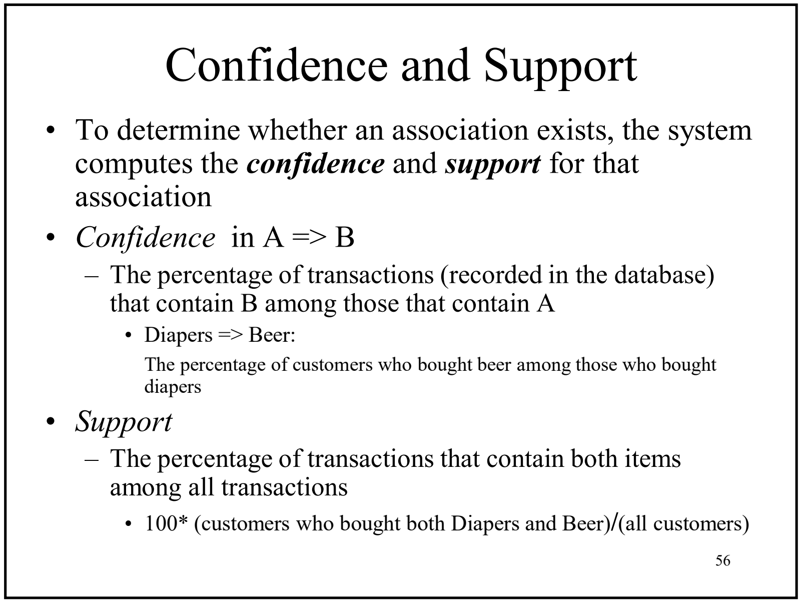
- The greater the **area under the ROC curve** (AUC), the better the classifier.
- A classifier with an AUC greater than 0.5 is better than a model that classifies at random.
- If AUC is lower than 0.5, then something is wrong, most likely a bug in the code or wrong labels in the data.
- A perfect classifier would have an AUC of 1. In practice, you obtain a good classifier by selecting the value of the threshold that gives TPR close to 1 while keeping FPR near 0.



Associations

- Remember that Association is “Finding patterns in data that associate instances of that data to related instances” (Slide no. 4)
- An *association* is a correlation between certain values in a database (in the same or different columns)
  - *In a convenience store in the early evening, a large percentage of customers who bought diapers also bought beer*
- This association can be described using the notation

  Purchase_diapers => Purchase_beer



Confidence and Support

- To determine whether an association exists, the system computes the ***confidence*** and ***support*** for that association
- *Confidence* in A => B
  - The percentage of transactions (recorded in the database) that contain B among those that contain A
    - Diapers => Beer:

      The percentage of customers who bought beer among those who bought diapers
- *Support*
  - The percentage of transactions that contain both items among all transactions
    - 100* (customers who bought both Diapers and Beer)/(all customers)

Ascertain an Association

- To ascertain that an association exists, both the confidence and the support must be above a certain threshold
  - Confidence states that there is a high probability, given the data, that someone who purchased diapers also bought beer
  - Support states that the data shows a large percentage of people who purchased both diapers and beer (so that the confidence measure is not an accident)

A Priori Algorithm for Computing Associations

- Based on this observation:
  - If the support for A => B is larger than *T*, then the support for *A* and *B* must separately be larger than *T*
- Find all items whose support is larger than *T*
  - Requires checking *n* items
  - If there are *m* items with support > T (presumably, m<<n), find all pairs of such items whose support is larger than *T*
  - Requires checking *m*(*m-1*) pairs
- If there are *p* pairs with support > T, compute the confidence for each pair
  - Requires checking *p* pairs

Classification

- *Classification* involves finding patterns in data items that can be used to place those items in certain categories.That classification can then be used to predict future outcomes.
  - *A bank might gather data from the application forms of past customers who applied for a mortgage and classify them as* ***defaulters*** *or* ***non-defaulters****.*
  - *Then when new customers apply, they might use the information on their application forms to predict whether or not they would default*

Example: Loan Risk Evaluation

- Suppose the bank used only three types of information to do the classification
  - Whether or not the applicant was married
  - Whether or not the applicant had previously defaulted
  - The applicant's current income
- The data about previous applicants might be stored in a table called the *training table*

Training Table

Id	*Married*	*PreviousDefault*	*Income*	*Default* (outcome)
C1	Yes	No	50	No
C2	Yes	No	100	No
C3	No	Yes	135	Yes
C4	Yes	No	125	No
C5	Yes	No	50	No
C6	No	No	30	No
C7	Yes	Yes	10	No
C8	Yes	No	10	Yes
C9	Yes	No	75	No
C10	Yes	Yes	45	No

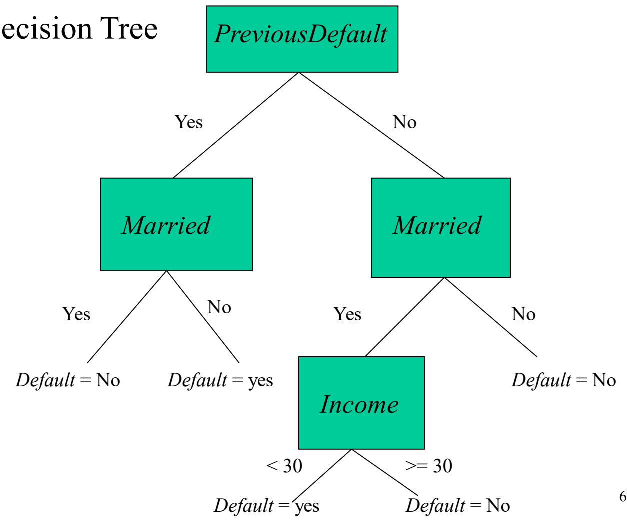
Training Table (cont’d)

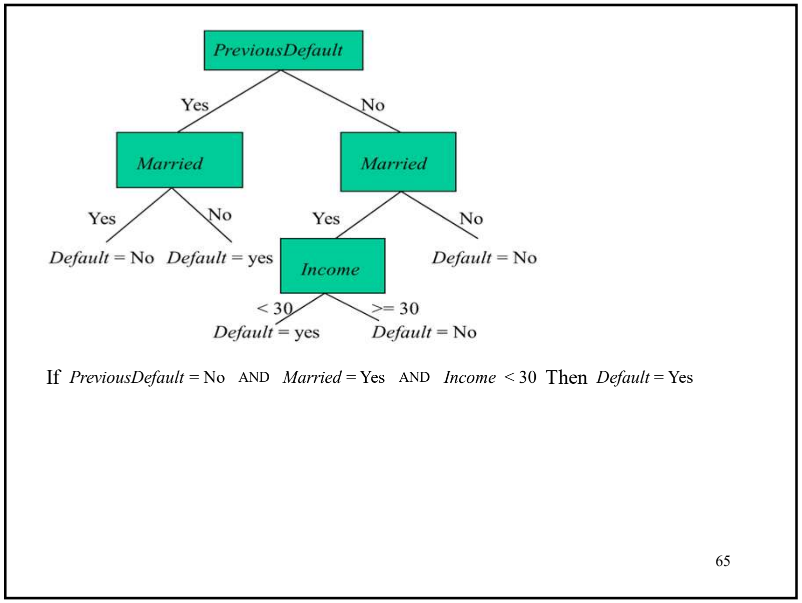
Id	*Married*	*PreviousDefault*	*Income*	*Default* (outcome)
C11	Yes	No	60	Yes
C12	No	Yes	125	Yes
C13	Yes	Yes	20	No
C14	No	No	15	No
C15	No	No	60	No
C16	Yes	No	15	Yes
C17	Yes	No	35	No
C18	No	Yes	160	Yes
C19	Yes	No	40	No
C20	Yes	No	30	No

Classification Using Decision Trees

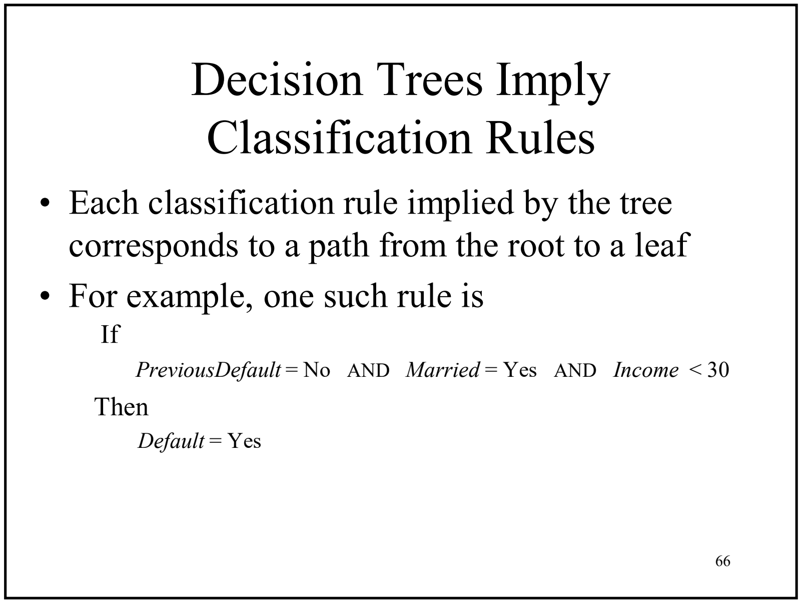
- The goal is to use the information in this table to classify new applicants into defaulters or non defaulters
- One approach is to use the training table to make a decision tree

A Decision Tree





If *PreviousDefault* = No AND *Married* = Yes AND *Income* < 30 Then *Default* = Yes



Decision Trees Imply Classification Rules

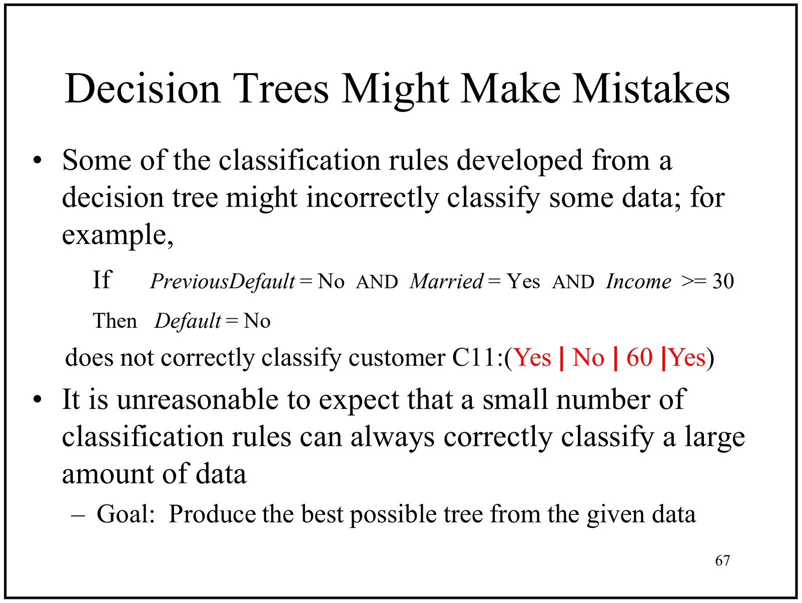
- Each classification rule implied by the tree corresponds to a path from the root to a leaf
- For example, one such rule is

  If

  PreviousDefault = No AND *Married* = Yes AND *Income* < 30

  Then

  Default = Yes



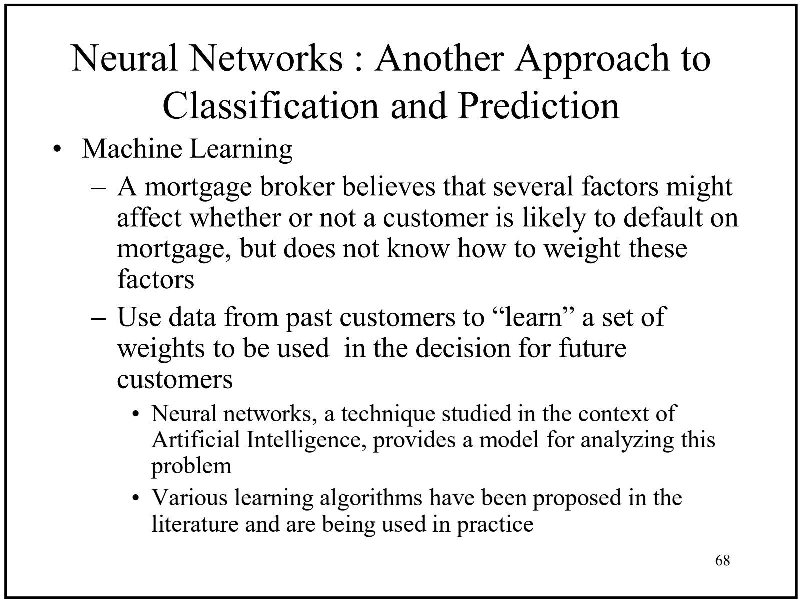
Decision Trees Might Make Mistakes

- Some of the classification rules developed from a decision tree might incorrectly classify some data; for example,

  If *PreviousDefault* = No AND *Married* = Yes AND *Income* >= 30

  Then *Default* = No

  does not correctly classify customer C11:(Yes | No | 60 |Yes)
- It is unreasonable to expect that a small number of classification rules can always correctly classify a large amount of data
  - Goal: Produce the best possible tree from the given data



Neural Networks : Another Approach to Classification and Prediction

- Machine Learning
  - A mortgage broker believes that several factors might affect whether or not a customer is likely to default on mortgage, but does not know how to weight these factors
  - Use data from past customers to “learn” a set of weights to be used in the decision for future customers
    - Neural networks, a technique studied in the context of Artificial Intelligence, provides a model for analyzing this problem
    - Various learning algorithms have been proposed in the literature and are being used in practice

A Model of a Neuron

- Suppose the factors are represented as x_i where each x_i can be 1 or 0, and the weight of each such factor is represented as w_i Then the weighted sum of the factors is compared with a threshold t. If the weighted sum exceeds the threshold

$$\sum_{i=1}^{n} w_i \times x_i \geq t$$

the output is 1 and we predict that the customer will default; otherwise the output is 0 and we predict he would be considered a good risk

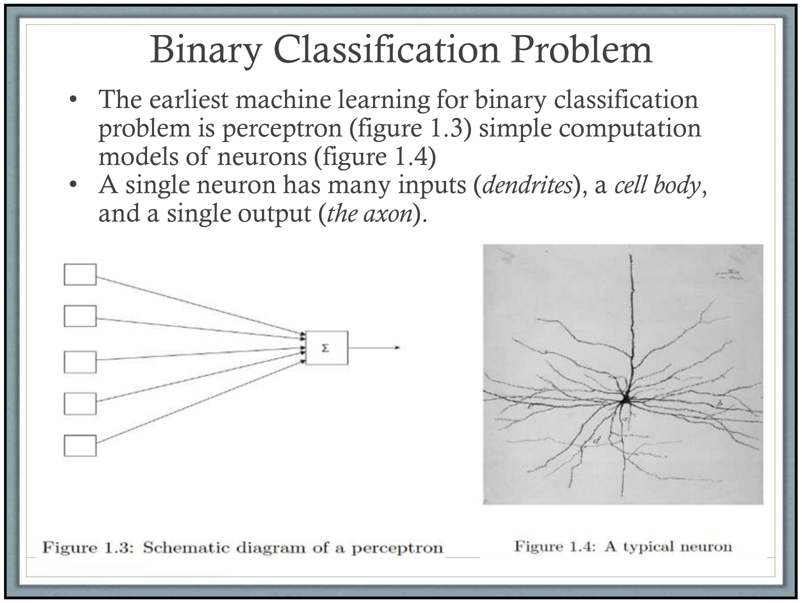
Simplified Model

- The model is simplified if we introduce a new weight w_0, which equals t, and assume there is a new input x_0 which always equals –1. Then the above inequality becomes

$$\sum_{i=0}^{n} w_i \times x_i \geq 0$$

Step-Function Activation

- This model is said to have **step-function activation**
  - Its output is 1 if the weighted sum of the inputs is greater than or equal to 0
  - Its output is 0 otherwise
- Neurons with this activation function are sometimes called **perceptrons**.
- Later we will discuss another activation function



Binary Classification Problem

- The earliest machine learning for binary classification problem is perceptron (figure 1.3) simple computation models of neurons (figure 1.4)
- A single neuron has many inputs (*dendrites*), a *cell body*, and a single output (*the axon*).

Figure 1.3: Schematic diagram of a perceptron

Figure 1.4: A typical neuron

- A perceptron consists of vector of *weights* **w** = [$w_1 \ldots w_m$] one for each input
- A distinguish weight **b** called **bias**
- **w** and **b** are called **parameters** denoted by **Φ with** $\Phi i \in \Phi$ the ith parameter. For a perceptron, $\Phi = \{w \cup b\}$.
- With these parameters the perceptron computes the function $f_\Phi(x)$:

1.1
$$f_\Phi(\mathbf{x}) = \begin{cases} 1 & \text{if } b + \sum_{i=1}^{l} x_i w_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

Let define the dot product of two vectors as:

1.2
$$\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{l} x_i y_i$$

Now, simplifying the notation for perceptron as

1.3
$$f_\Phi(\mathbf{x}) = \begin{cases} 1 & \text{if } b + \mathbf{w} \cdot \mathbf{x} > 0 \\ 0 & \text{otherwise} \end{cases}$$

```
1. set $b$ and all of the $\mathbf{w}$'s to 0.
2. for $N$ iterations, or until the weights do not change
   (a) for each training example $\mathbf{x^k}$ with answer $a^k$
       i. if $a^k - f(\mathbf{x}^k) = 0$ continue
       ii. else for all weights $w_i$, $\Delta w_i = (a^k - f(\mathbf{x}^k))x_i$
```

Figure 1.5: The perceptron algorithm

- ML can be characterized as a *function approximation* problem. Figure 1.5 gives a pseudocode for perceptron algorithm.
- a^k is either 0 or 1 indicating if the image is member of the class or not.
- Lines 2a(i) and 2a(ii): the first, if the output in the perceptron is correct do nothing; If not, the second, change the weight so that the output can be correct the next time around i.e. add $(a_k - f(x^k))x_i^k$ to each parameter w_i

- Perceptron can be extended to multiclass decisions problems (figure 1.6).
- All the perceptrons are trained independently using exactly the same algorithm.
- The answer return is the highest value.

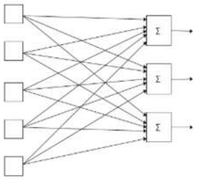


Figure 1.6: Multiple perceptrons for identification of multiple classes

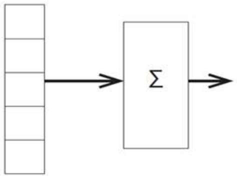


Figure 1.7: NN showing layers

Perceptron Learning Algorithm

- Set the values of each weight (and threshold) to some small random number
- Apply the inputs one at a time and compute the outputs
- If the desired output for some input is d and the actual output is y, change each weight w_i by

$$\Delta w_i = \eta \times x_i \times (d - y)$$

where η is a small constant called the **learning factor**

- Continue until some termination condition is met

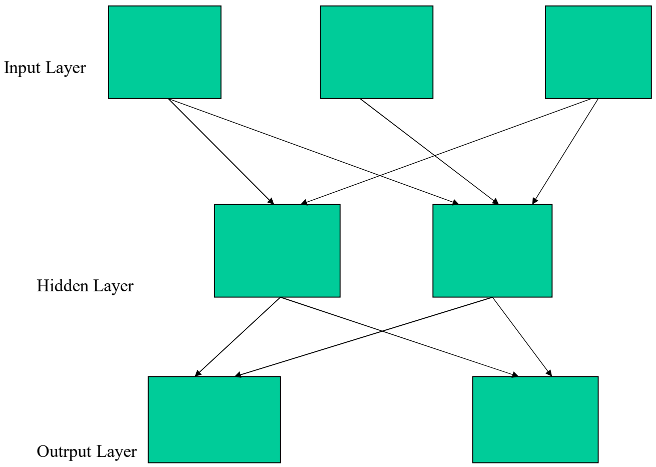
Rationale for Learning Algorithm

- If there is no error, no change in the weights are made
- If there is an error, each weight is changed in the direction to decrease the error
  - For example, if the output is 0 and the desired output is 1, the weights of all the inputs that were 1 are increased and the threshold is decreased.

Correctness and Problems with Perceptron Learning Algorithm

- If the decision can always be made correctly by a single neuron, this algorithm will eventually “learn” the correct weights
- The problem is that, for most applications, the decision cannot be made, even approximately, by a single neuron
- We therefore consider networks of such neurons

Three Level Neural Network



Three-Level Network

- The input level just gathers the inputs and submits them to the other levels (no neurons)
- The middle or hidden level consists of neurons that make intermediate decisions and send them to the output layer
- The output layer makes the final decisions

The Sigmoid Activation Function

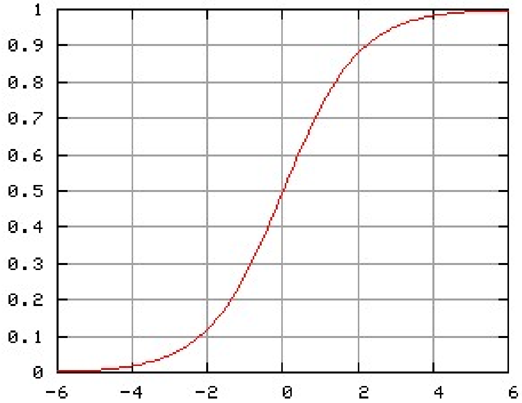
- To mathematically derive a learning algorithm for such a neural network, we must take derivatives
  - But we cannot take derivatives of the step function activation function
- Therefore we must use a continuous activation function
  - A common such activation function is the sigmoid function

    $y = 1/(1+e^{-X})$

    where

    $$X = \sum_{i=0}^{n} w_i \times x_i$$

The Sigmoid Function



Properties of Sigmoid Function

- In some sense the sigmoid function is similar to the step function
  - It has the value *.5* for $x = 0$
  - It becomes asymptotic to *1* for large positive values of *x*
  - It becomes asymptotic to *0* for large negative values of *x*
- However it is continuous and, as can be easily computed, has the derivative

$$\frac{\partial y}{\partial X} = e^{-X} /(1+e^{-X})^2 = y \times (1-y)$$

Learning Algorithm for a Single Sigmoid Neuron

- The idea is to take the derivative of the squared error with respect to each of the weights and change each weight by a small multiple of the negative of that derivative
  - Called the Gradient Descent Approach
  - Move in the direction towards the minimum of the function

$$\Delta w_i = -\eta \times \frac{\partial (d-y)^2}{\partial w_i}$$

Clustering

- Given:
  - a set of items
  - characteristic attributes for the items
  - a similarity measure based on those attributes
- ***Clustering*** involves placing those items into ***clusters***, such that items in the same cluster are close according to the similarity measure
  - Different from Classification: there the categories are known in advance
- For example, cancer patients might have the attribute *location*, and might be placed in clusters with similar locations.

Example: Clustering Students by Age

Student Id	**Age**	**GPA**
S1	17	3.9
S2	17	3.5
S3	18	3.1
S4	20	3.0
S5	23	3.5
S6	26	3.6

K-Means Algorithm

- To cluster a set of items into *k* categories
  1. Pick *k* items at random to be the (initial) centers of the clusters (so each selected item is in its own cluster)
  2. Place each item *in the training set* in the cluster to which it is closest to the center
  3. Recalculate the centers of each cluster as the mean of the items in that cluster
  4. Repeat the procedure starting at Step 2 until there is no change in the membership of any cluster

The Student Example (con't)

- Suppose we want 2 clusters based on *Age*
  - Randomly pick S1 (age 17) and S4 (age 20) as the centers of the initial centers
  - The initial clusters are

    17 17 18   20 23 26
  - The centers of these clusters are

    17.333 and 23
  - Redistribute items among the clusters based on the new centers:

    17 17 18 20   23 26
  - If we repeat the procedure, the clusters remain the same

The Hiearchical or Aglomerative Algorithm

- Number of clusters is not fixed in advance
- Initially select each item in the training set as the center of its own cluster
- Select two clusters to merge into a single center
  - One approach it to pick the clusters whose centers are closest according to some measure (e.g., Euclidian distance)
- Continue until some termination condition is reached (e.g., the number of clusters falls below some limit)

Student Example (con’t)

17   17   18   20   23   26

17 17   18   20   23   26

17 17 18   20   23   26

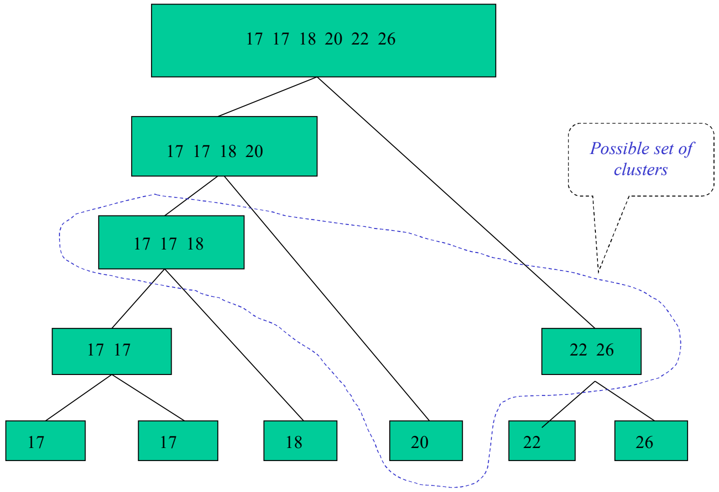
17 17 18 20   23   26

17 17 18 20   23 26 --- K-means Solution

17 17 18 20 23 26

Dendrogram

- One way to <u>manually</u> *analyze* the results of the hierarchical algorithm is with the use of a tree called a ***dendrogram***
- The nodes are clusters in the intermediate stages of the hierarchical algorithm
- The tree is constructed in reverse order of the execution of the hierarchical algorithm, starting with the final (single) cluster



A Dendrogram for the Student Example

Analysis of Dendrogram

- Any set of nodes whose children *partition* all the leaves is a possible clustering
  - For example,

    17 17 18 20 23 26

    is an allowable set of clusters.

    Note: these clusters were not seen at any of the intermediate steps in the hierarchical or K- means algorithms!