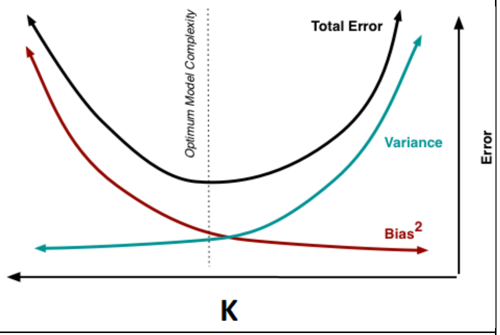
	Supervised Learning (Data is labelled)	Unsupervised
Categorical classification	K-nearest neighbor	
	Decision Trees	
	Naïve Bayes Classifier	
	Logistic Regression	
Continuous	Linear Regression	K-means clustering
		Association Rules

K-nearest neighbor (KNN)

- making predictions about a categorical outcome y based on a num of predictors x
- obj is assigned to the class most common among its k nearest neighbors
- to determine the KNN, use Euclidean dist, sqrt($(x_1-x_0)^2 + (y_1-y_0)^2$)
- y usually set to 1 or 0 depending on the classification
- fitted outcome val/predicted class membership = average y value
-predicted outcome val = y < 0.5, assign to class 0 or if y > 0.5, class 1
- By taking larger k, variance will decrease, but since we are taking data points further away, may lead to greater bias in calculating ave. y
- Prediction error for a model = bias^2 + variance + irreducible error
- Essentially, variance = opp of precision, bias = opp of accuracy



bias-variance tradeoff

bias: unaccuracy, variance: un-precision

True Positive: Predict true when true
True -ve: Predict false when false
False +ve: Predict true when false
False -ve: Predict false when true
Good classifier have large TP and TN and small/zero FP and FN

Confusion matrix

		Predicted class	
		+ve	-ve
Actual class	+ve	TP	FN
	-ve	FP	TN

Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$ x 100%
Precision = $\frac{TP}{TP+FP}$
Ideally, TPR = 1 and (FPR and FNR = 0)
threshold, $\sigma\uparrow$: predicted +ve $\downarrow$, TPR & FPR $\downarrow$ or same
$\sigma\uparrow$: predicted -ve $\uparrow$, TNR & FNR $\uparrow$ or same

True +ve rate (TPR) = $\frac{TP}{TP+FN}$
True -ve rate (TNR) = $\frac{TN}{TN+FP}$
False +ve rate (FPR/false alarm rate/type I error rate) = $\frac{FP}{FP+TN}$
False -ve rate (FNR/miss rate/type II error rate) = $\frac{FN}{TP+FN}$

N-Fold Cross-Validation

Entire dataset randomly split into N datasets of ≈ equal size
N-1 of these datasets are treated as training dataset, while remaining one would be test dataset and measure model
Repeat process but change test set each time for the remaining N-1 time (1st: a as test, 2nd: b as test... 10th: j as test)
The N models errors are averaged across the N folds

Decision Trees

Decision/Prediction tree uses a tree structure to specify seq of decisions and consequences
At each node/test point, test a particular var/attribute and pick a specific child/branch and traverse down tree
Eventually when leaf is reached, prediction is made
Depth: min no. of steps to reach node from root.

Building a decision tree
1. Choose the most informative attribute
- Use entropy-based methods (used by ID3, Iterative Dichotomiser 3 and C4.5) or Gini index
Purity of node = prob of corresponding class

Entropy Mtd

Entropy mtd selects most informative var based on
1. Entropy = impurity of var/attribute
2. Information gain = purity of var/attribute
Entropy is a measure of unpredictability
If Y is binary and only takes on values 0 or 1,
$D_Y = -[P(Y = 1)\log_2 P(Y = 1) + P(Y = 0)\log_2 P(Y = 0)]$
When coin is biased, less "uncertainty: in predicting outcome of next toss, so entropy is lower
When coin is fair, entropy is max
Ideally in decision tree, entropy should be low so that our predictions have less "uncertainty"

Given variable Y and set of possible categorical values it can take, $(y_1, y_2, ..., y_k)$
Entropy of Y = $D_Y = -\sum_{j=1}^{K} P(Y = y_j)\log_2 P(Y = y_j)$
where $P(Y = y_j)$ denotes purity/probability of class $Y = y_j$, and $\sum_{j=1}^{K} P(Y = y_j) = 1$
Suppose we have feature X and split values (x_1, x_2).
Conditional entropy given X and split points (x_1, x_2) = $D_{Y|X}$ = $\sum_{i=1}^{2} P(X = x_i) D(Y|X = x_i)$ = $-\{\sum_{i=1}^{2} P(X = x_i) \sum_{j=1}^{K} P(Y = y_j|X = x_i)\log 2[P(Y = y_j|X = x_i)]\}$
So reduction in entropy = information gain = $D_Y - D_{Y|X}$
So chose var and how to split according to most info gain

Thus, decision tree algo recursively calculate conditional entropy for (i) each variable X and (ii) its different split points
Then var and its split points selected based on largest info gain until
(i) All leaf nodes in tree satisfy min purity threshold
(ii) Tree cannot be further split with preset min purity threshold
(iii) Any other stopping cond satisfied (e.g. max depth of tree)

Another mtd is Gini index
Given variable Y and set of possible categorical values it can take, $(y_1, y_2, ..., y_k)$
Gini index of Y = $G_Y = \sum_{j=1}^{K} P(Y = y_j)[1-p(Y = y_j)]$
where $P(Y = y_j)$ denotes purity/probability of class $Y = y_j$, and $\sum_{j=1}^{K} P(Y = y_j) = 1$

Decision or prediction surface corresponds to a leaf node of the tree, and it can be reached by traversing from the root of the tree following by a series of decisions according to the value of an attribute.
The decision surface can only be axis-aligned for the decision tree. (knn don't need be axis-aligned)

Naive Bayes theorem

Bayes theorem gives relationship btw probabilities of 2 events and their conditional probs
Naive Bayes classifier assumes that features are indep. of other features (they dont affect each other)
Input variables are normally categorical, but some algo accept continuous variables
Discretization of continuous variables: convert continuous vars to categorical vars
Output usually class label and its probability

Conditional prob of C occuring, given A has occurred: P(C|A) = $\frac{P(A \cap C)}{P(A)} = \frac{P(A|C)*P(C)}{P(A)}$, where P(A∩C) is prob that A and C occur
P(C|A) usually more diff to compute than P(A|C) and P(C)
E.g. 1% of pop has disease. Lab test gives 95% TP, 6% FP
Let C = {have disease}, A = {+ve test result}
P(C) = 0.01, P(¬C) = 0.99, P(A|C) = 0.95, P(A|¬C) = 0.06
P(C|A) = $\frac{P(A|C)*P(C)}{P(A)} = \frac{P(A|C)P(C)}{P(A \cap C)+P(A \cap \neg C)} = \frac{P(A|C)P(C)}{P(C)P(A|C)+P(\neg C)P(A|\neg C)} = \frac{0.95*0.01}{0.01*0.95+0.99*0.06} \approx 0.1379$

Bayes' Theorem: More generally, suppose categorical outcome var Y can take on vals $\{y_1, y_2 \ldots y_k\}$ and feature vars X = $\{X_1, X_2, \ldots, X_m\}$, then $P(Y = y_j|X) = \frac{P(X_1 = x_1,\ X_2 = x_2,\ \ldots\ X_m = x_m|Y = y_j)P(Y = y_j)}{P(X_1 = x_1,\ X_2 = x_2,\ \ldots\ X_m = x_m)}$ for j = 1...k

naïve Bayes theorem: 1. Assume conditional independence of feature vars. i.e. $P(X_1 = x_1, X_2 = x_2, \ldots, X_m = x_m|Y = y_j) = P(X_1 = x_1|Y = y_j)P(X_2 = x_2|Y = y_j) \ldots P(X_m = x_m|Y = y_j) = \prod_{i=1}^{m} P(X_i = x_i|Y = y_j)$
2. Ignore denominator since it is constant for all values of Y
Combining, $P(Y = y_j|X) \propto P(Y = y_j) * \prod_{i=1}^{m} P(X_i = x_i|Y = y_j)$ for j = 1,2,...,k

E.g.

j	1	2	3
$P(Y = y_j)$	0.5	0.3	0.2
$P(x_1\|y_j)$	0.4	1/15	0.5
$P(x_2\|y_j)$	0.2	1/3	0.25
$P(x_3\|y_j)$	0.4	0.6	0.25

X = {1, 2, not 3}
$P(Y = 1|X) \propto P(Y = 1) * P(x_1|y_1) * P(x_2|y_1) * P(\neg x_3|y_1) = 0.5 * 0.4 * 0.2 * (1 - 0.4) \approx 0.024$
$P(Y = 2|X) \propto 0.3 * 1/15 * 1/3 * (1 - 0.6) \approx 0.0027$
$P(Y = 3|X) \propto 0.2 * 0.5 * 0.25 * (1 - 0.25) \approx 0.0188$
Thus, given X, we predict object to be class 1

To prevent probability from tending to 0, can compute log: $\log P(Y = y_j) + \sum_{i=1}^{m} log\ P(X_i = x_i|Y = y_j)$

Linear Regression

Assume relationship btw input variables and outcome are linear
$y_i \approx ß_0 + ß_1 x_i$
Best fit line: vertical dist btw each data point to line is minimised
Residual: dist btw line and pt: $e_i = y_i - (ß_0 + ß_1 x_i)$
Residual Sum of Squares (RSS) : RSS = $e_1^2 + e_2^2 + \ldots + e_i^2$
Least squares: Process of minimizing RSS: partial differentiate wrt to $ß_0$,then $ß_1$

More generally, RSS = $\sum_{i=1}^{n}[y_i - (ß_0 + ß_1 x_i)]^2$
$\frac{\partial \text{RSS}}{\partial ß_0} = \sum_{i=1}^{n} 2[y_i - (ß_0 + ß_1 x_i)](-1) = 0$
$nß_0 + ß_1\sum_{i=1}^{n} x_i - \sum_{i=1}^{n} y_i = 0 \Rightarrow ß_0 + ß_1\bar{x} - \bar{y} = 0$ – (1)
$\frac{\partial \text{RSS}}{\partial ß_1} = \sum_{i=1}^{n} 2[y_i - (ß_0 + ß_1 x_i)](-x_i) = 0$
$ß_0\sum_{i=1}^{n} x_i + ß_1\sum_{i=1}^{n} {x_i}^2 - \sum_{i=1}^{n} x_i y_i = 0$ – (2)
$(\bar{y} - ß_1\bar{x})\sum_{i=1}^{n} x_i + ß_1\sum_{i=1}^{n} {x_i}^2 = \sum_{i=1}^{n} x_i y_i$
$\widehat{ß}_1 = \{\sum_{i=1}^{n} x_i y_i - \bar{y}\sum_{i=1}^{n} x_i\} / \{\sum_{i=1}^{n} {x_i}^2 - \bar{x}\sum_{i=1}^{n} x_i\}$ and
$\widehat{ß}_0 = \bar{y} - \widehat{ß}_1\bar{x}$ (^ represent estimated values)

Logistic Regression

Logistic regression can be applied to outcome that have multiple values (not just binary)
Logistic fn: $f(z) = \frac{exp(z)}{1+exp(z)} = 1 - \frac{1}{1+exp(z)}$
Note as $z \to \infty$, $f(z) \to 1$. As $z \to -\infty$, $f(z) \to 0$
In logistic regression, z is linear fn of input variables, i.e. $z = ß_0 + ß_1X_1 + ß_2X_2 + \ldots + ß_pX_p$
$P(Y = 1|X_1, X_2, \ldots, X_p) = \frac{exp(ß_0 + ß_1X_1 + ß_2X_2 + \ldots + ß_pX_p)}{1+exp(ß_0 + ß_1X_1 + ß_2X_2 + \ldots + ß_pX_p)}$
To estimate ß₀, ß₁..., use maximum likelihood estimation (MLE), which maximise L(p)
Likelihood fn of p given observed data: $L(p) = \prod_{i=1}^{n} p^{y_i}(1-p)^{1-y_i}$, where y_i is outcome (0 or 1) for each ith data point, and p = P(Y = 1), 1-p = P(Y = 0)
Easier to work with logarithm of L:
$\ln L(p) = \sum_{i=1}^{n} y_i ln(p) + \sum_{i=1}^{n}(1-y_i)ln(1-p)$
Let $n_1 = \sum_{i=1}^{n} y_i$ = (num of y = 1),
$n_0 = \sum_{i=1}^{n}(1-y_i)$ = (num of y = 0). And $n_1 + n_0 = n$
$\ln L(p) = n_1 \ln(p) + n_0 \ln(1 - p)$
To get max ln L(p): $\frac{d}{dp}[\ln L(p)]$: $\frac{n_1}{p} - \frac{n_0}{1-p} = 0 \Rightarrow n_1(1-p) = n_0 p \Rightarrow \hat{p}_{mle} = \frac{n_1}{n_1+n_0} = \frac{n_1}{n}$. By right, need check if max using 2nd derivative < 0

$\pi(X) = P(Y = 1|X) = \frac{exp(ß_0 + ß_1X)}{1+exp(ß_0 + ß_1X)}$, and
$P(Y = 0|X) = 1 - \frac{exp(ß_0 + ß_1X)}{1+exp(ß_0 + ß_1X)} = \frac{1}{1+exp(ß_0 + ß_1X)}$
$L(ß_0, ß_1) = \prod_{i=1}^{n}\left[\frac{exp(ß_0 + ß_1X_i)}{1+exp(ß_0 + ß_1X_i)}\right]^{y_i}\left[\frac{1}{1+exp(ß_0 + ß_1X_i)}\right]^{1-y_i}$
$= \prod_{i=1}^{n}\frac{exp(ß_0 + ß_1X_i)^{y_i}}{1+exp(ß_0 + ß_1X_i)} = \prod_{i=1}^{n}\frac{exp[y_i(ß_0 + ß_1X_i)]}{1+exp(ß_0 + ß_1X_i)}$
$\ln L(ß_0, ß_1) = \sum_{i=1}^{n}\{y_i(ß_0 + ß_1x_i) - \ln(1 + exp(ß_0 + ß_1x_i))\}$
Note $\frac{\partial}{\partial ß_0}\sum_{i=1}^{n} y_i(ß_0 + ß_1X_i) = \sum_{i=1}^{n} y_i$ and
$\frac{\partial}{\partial ß_0}\ln(1 + exp(ß_0 + ß_1x_i)) = \frac{1}{1+exp(ß_0+ß_1X_i)}\frac{\partial}{\partial ß_0}(1 + exp(ß_0 + ß_1x_i)) = \frac{exp(ß_0 + ß_1X_i)}{1+exp(ß_0 + ß_1X_i)} = \pi[X_i]$
$\frac{\partial}{\partial ß_1}\sum_{i=1}^{n} y_i(ß_0 + ß_1X_i) = \sum_{i=1}^{n} x_i y_i$
and $\frac{\partial}{\partial ß_1}\ln(1 + exp(ß_0 + ß_1x_i)) = \frac{1}{1+exp(ß_0+ß_1X_i)}\frac{\partial}{\partial ß_1}(1 + exp(ß_0 + ß_1x_i)) = \frac{X_i*exp(ß_0 + ß_1X_i)}{1+exp(ß_0 + ß_1X_i)} = X_i\pi[X_i]$
Solving $\sum_{i=1}^{n}\left\{y_i - \frac{exp(ß_0 + ß_1X_i)}{1+exp(ß_0 + ß_1X_i)}\right\} = 0$ and $\sum_{i=1}^{n}\left\{x_i y_i - x_i\frac{exp(ß_0 + ß_1X_i)}{1+exp(ß_0 + ß_1X_i)}\right\} = 0$ will give MLE $\widehat{ß}_0$ and $\widehat{ß}_1$
These 2 fn aka score fn and typically solved by numerical mtds (Newton-Raphson algo)

Clustering

Clustering refers to a broad set of techniques for finding clusters in a data set
Each cluster contains observations of a data set that are quite similar to each other, while diff clusters are quite diff from each other
Clustering is an exploratory technique to discover hidden structures of the data
Clustering mtds: K-means clustering, mean-shift clustering, density-based spatial clustering of applications with noise (DBSCAN), expectation-maximization (EM) clustering using Gaussian mixture models (GMM) and agglomerative hierarchical clustering

K-means clustering

For a chosen value of k, identify k clusters based on data proximity to center of clusters

1. Input k random centroid (center of cluster)

	Center is the mean of each clusters's n-dimensional vector of attributes	2. For each data point, assign them to nearest cluster using euclidean distance 3. Calculate new centroid for each cluster (avg all cluster points) 4. Repeat step 2 and 3 until centroids converges (< threshold)
	Value of k chosen by calculating Within Sum of Squares (WSS) For M data points $z_1, z_2, ..., z_M$ SS = sum of squares of one cluster = $dist(z_i, D)^2$ = dist of data pt that belongs to cluster with centroid D to D WSS = $\sum_{i=1}^{M} dist(z_i, D_i)^2$ = sum of all SS (lower better)	Plot value of WSS against k Choose k where WSS_{k+1} does not decrease significantly compared to WSS_k This proces of finding appropriate k = finding "elbow" of WSS curve
Association Rules	Descriptive (not predictive) method for unsupervised learning Relationships between features are called rules Association rules often used in mining customer transactions and is aka market basket analysis	Each transaction can be viewed as a basket of items: aka itemset Itemset containing k items: k-itemset {item1, item2, ..., item k} Computation of association rules typically based on itemsets We will use Apriori algo to generate association rules
Apriori Algo	Given an itemset L, support of L = % of transactions that contains L Frequent itemset: set that has items appearing tgt often enough (i.e. support ≥ minimum support criterion) If itemset is frequent, any subset of this frequent itemset must also be frequent This property is known as Apriori property / downward closure property i.e. 60% of transactions contains {x1, x2}, then ≥ 60% of transactions must contain {x1} or {x2}	Apriori algo uses bottom-up iterative approach by first determining all the 1-itemsets, and identifying which of them are frequent based on the min support criterion. In next iteration, those 1-itemsets that are frequent are then paired into 2-itemset, and check if ≥ min support At each iteration, algo checks if support criterion is met, if it is, then itemset is expanded, else Algo stops once there is no itemset that meet min support threshold or itemset has reach a predefined length, N Apriori algo will output all frequent k-itemsets
Candidate Rules	Based on the frequent k-itemsets, collection of candidate rules are formed, e.g. frequent itemset {x1, x2} may suggest candidate rules {x1} → {x2} and {x2} → {x1}	Measure such as confidence, lift and leverage is used to evaluate appropriateness of these candidate rules
Confidence and Relationship	Confidence is defined as the measure of certainty/ trustworthiness associated with each discovered rule Confidence for X → Y is % of transactions that contain both X and Y out of all transactions containing X, i.e. Confidence(X → Y) = $\frac{Support(X \wedge Y)}{Support(X)}$ i.e. if X occurs, prob that X and Y occurs	Relationship is interesting if confidence ≥ predefined threshold / min confidence Higher confidence ⇒ rule is more interesting / trustworthy Problems with confidence: given a rule X → Y, confidence only considers antecedent (X) and cooccurrence of X and Y; it does not consider the consequent of the rule (Y) To address this issue, we use lift and leverage
Lift and Leverage	Lift measures how many times more often X and Y occur tgt than expected if they are indep of each other Lift is a measure of how X and Y are really related rather than coincidentally happening tgt: Lift(X → Y) = $\frac{Support(X \wedge Y)}{Support(X) * Support(Y)}$ Lift is 1 if X and Y are indep of each other Lift > 1 indicates there is usefulness to the rule	Leverage has similar idea with lift, but uses diff instead of a ratio Leverage(X → Y) = Support(X∧Y) - Support(X) * Support(Y) Leverage measures diff in prob of X and Y occurring tgt compared to if they are indep of each other Leverage = 0 if X and Y are indep of each other Larges leverage value indicates stronger r/s btw X and Y

	Market basket analysis refers to a specific implementation of association rules that many companies used	library(rgl)
		plot3d(x1, x2, x3, col = kout$cluster)

Map Reduce	Big data has volume, variety and velocity 1. Structured data: data containing a defined data types, formats and structures. Usually stored in databases/ spreadsheets. Mainly collected from transactions or online analytic processing 2. Semi-structured data: Textual data files with discernible pattern that enables easy parsing 3. Quasi-structured dataL Textual data with erratic data formats that can be formatted with effort, tool and time 4. Unstructured data: data that has no inherent structure	MapReduce break a large tasks into smaller tasks, run tasks in parallel, and consolidate the outputs of the individual tasks into the final output. Map: Applies an operation to a piece of data and provides some intermediate output (turn data into key-value pair) Key-value pairs from each Map tasks are collected by a master controller and sorted by key. The key-value pairs are divided among the Reduce tasks by their keys Reduce: Consolidates the intermediate outputs from the map steps and provides the final output. Reduce tasks work on one key at a time, and combine all value associated with that key in some way.
	Let matrix M n x n, m_{ij} = element in row i, col j. Let col vector v be length n. Mv = $\begin{pmatrix} m_{11} & m_{12} & \cdots & m_{1n} \\ m_{21} & m_{22} & \cdots & m_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n1} & m_{n2} & \cdots & m_{nn} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$ $x_i = \sum_{j=1}^{n} m_{ij} v_j$	Entries of M are stored as triplets (i, j, m_{ij}) During map step, from each triplet, it produces the key-value pair (i, $m_{ij}v_j$). Reduce function sums all the values associated with given key i. Result will be (i, x_i) Final result will be Mv when sorted according to i