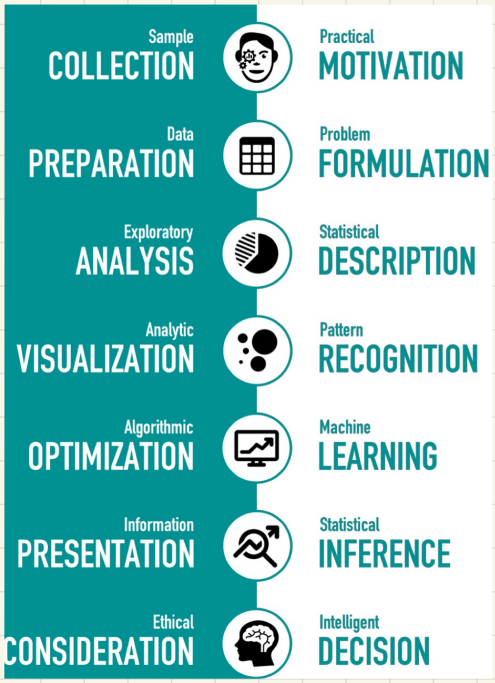
Steps to Data Science



Data Science Problem

Predicting: 1) Numeric → How much / How many sales in the next year?
answers

2) Classes → Boolean type / Is it type A / type B

Detecting: 1) Structure

2) Anomalty → Anything that is unusual / faulty / suboptimal

Decision: 1) Action → for AI and the "result" of data

Regression

- linear regression
- Tree model
- Neural Network

Structure Detection:

- Clustering
  - Works to find multiple groups of people / data
  1) Distance: Eudlicean, Jaccard
  2) k-Means Algorithm
  3) Heirarchical model for clustering

Anomalty Detection

1) Cluster Analysis
2) Nearest Neighbour
3) Support Vector

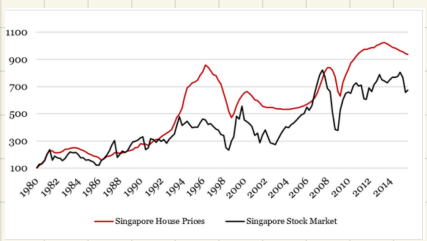
Structured Data Types

Mixed Data

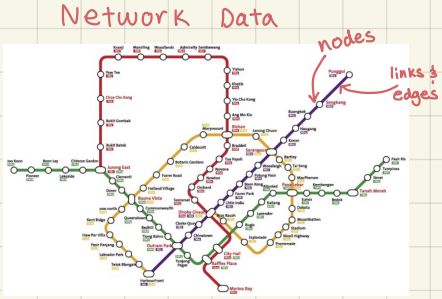
Price	Doors	Seats	Condition
230.1	4	2	unacc
44.5	5more	more	good
17.2	5more	more	vgood
151.5	2	2	unacc
180.8	2	2	unacc
8.7	4	more	acc
57.5	5more	2	unacc
8.6	4	4	acc
199.8	2	more	acc
66.1	4	2	unacc
214.7	3	4	unacc
23.8	3	4	unacc

Numeric
Categorical

Time-Series Data



Network Data



Unstructured Data Types (text, image, voice, video)

Text

#Cashless payments could soon be a way of life for students in @NTUsg, from the way they #pay to the way they attend classes http://tmsk.sg/aM

Eyeing a Smart Campus: Here's #NTUsg's new leadership team at their first town hall session with the NTU community. They shared how smart technologies will be used at NTU to improve learning and living experiences.

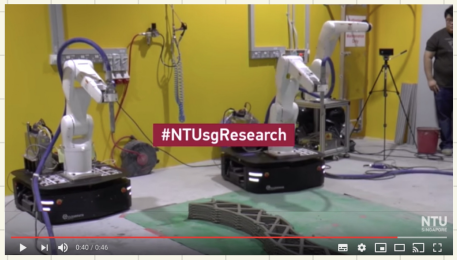
#NTUsg partners Volvo to develop #autonomous #electricbuses in Singapore. NTU is the first university in the world to work with Volvo on self-driving technology for buses. #NTUsgResearch

- Context specific
- Words/emotion

image

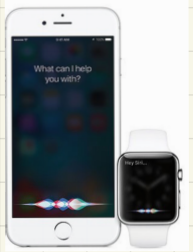
- Pixels ~> Object?

video



- images + frames
- identify object as 3D printer

voice

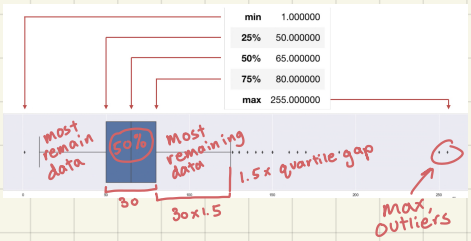


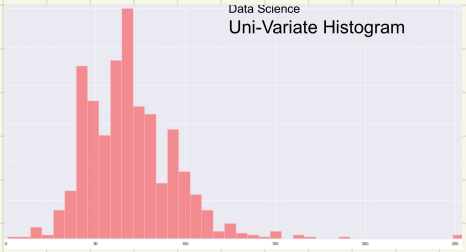
- waves to info

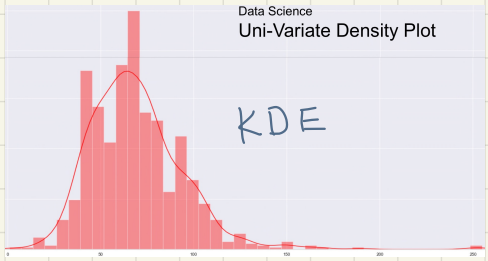
Uni-Variate Statistics

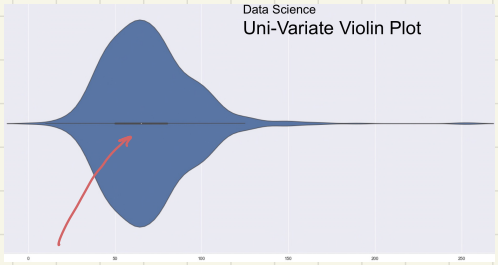
count	total
mean	average
std	δ
min	median
25%	low $\frac{1}{4}$
50%	median
75%	high $\frac{1}{4}$
max	Highest

Box Plot







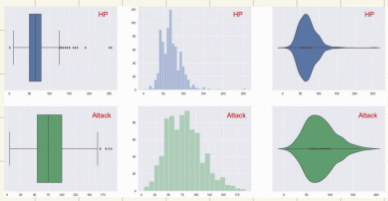


Box Plot here

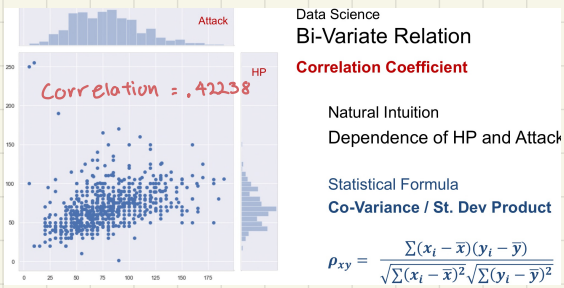
Normal Distribution

Bi-Variate Exploration

	HP		Attack
count	800.000000	count	800.000000
mean	69.258750	mean	79.001250
std	25.534669	std	32.457366
min	1.000000	min	5.000000
25%	50.000000	25%	55.000000
50%	65.000000	50%	75.000000
75%	80.000000	75%	100.000000
max	255.000000	max	190.000000



-> Two Variables ∴ Statistical relationship



Data Science

Bi-Variate Relation

Correlation Coefficient

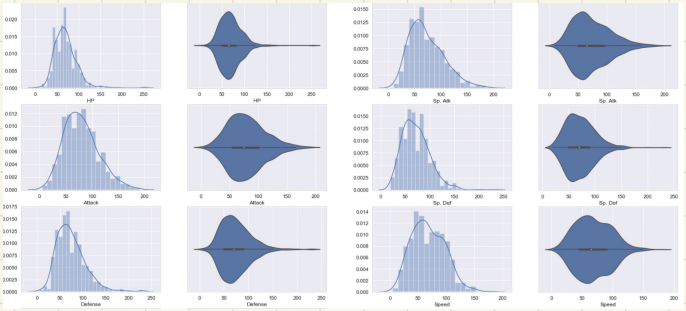
Natural Intuition

Dependence of HP and Attack

Statistical Formula

Co-Variance / St. Dev Product

$$\rho_{xy} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2}\sqrt{\sum(y_i - \bar{y})^2}}$$



	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed
HP	1.000000	0.422386	0.239622	0.362380	0.378718	0.175952
Attack	0.422386	1.000000	0.438687	0.396362	0.263990	0.381240
Defense	0.239622	0.438687	1.000000	0.223549	0.510747	0.015227
Sp. Atk	0.362380	0.396362	0.223549	1.000000	0.506121	0.473018
Sp. Def	0.378718	0.263990	0.510747	0.506121	1.000000	0.259133
Speed	0.175952	0.381240	0.015227	0.473018	0.259133	1.000000

Machine learning

→ Machine takes hints from data set, then get tested by test-case

Supervised Learning
- Regression
- Classification

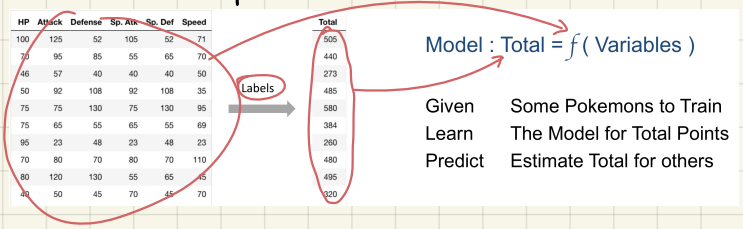
Un-Supervised Learning
- Clustering
- Anomaly Detection

① Learning
② Testing

} Optimally learn from the data

Numeric Problem Solutions

1) Regression : Take predictors(data), pass through function, and return predicted total

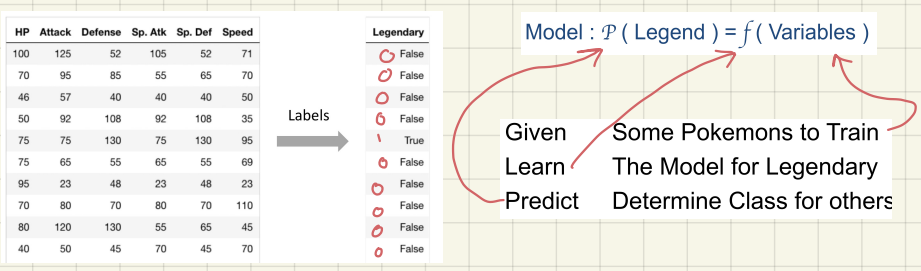


Supervised Learning
- Test during training
- Test during final Exam

2) Prediction ō Classes : Classification

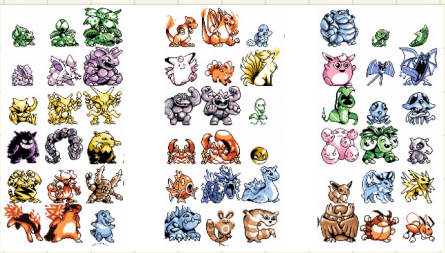
★ Is it type A or type B?

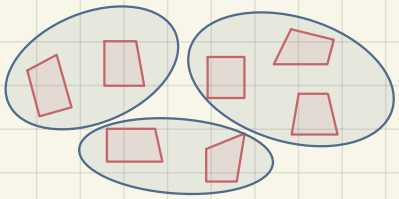
Can you predict if it is A/B?



3) Detecting Structure : Clustering

- Group datapoints into smaller clusters
- Structure





Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed
505	100	125	52	105	52	71
440	70	95	85	55	65	70
273	46	57	40	40	40	50
485	50	92	108	92	108	35
580	75	75	130	75	130	95
384	75	65	55	65	55	69
260	95	23	48	23	48	23
480	70	80	70	80	70	110
495	80	120	130	55	65	45
320	40	50	45	70	45	70

Grouping depends on "Distance"

Given	Distance on all Pokemons
Find	Optimal Groups in the Data
Justify	Interpretation of the Groups

The reason behind the groups, "non-random"

4) Anomaly Detection : Is it weired?

Unsupervised learning for both

1) Did not tell machine what to look out for
2) No training provided, no external influence
3) Natural Development

Uni Variate Regression

Total	HP
505	100
440	70
273	46
485	50
580	75
384	75
260	95
480	70
495	80
320	40

→

	HP
count	800.000000
mean	69.258750
std	25.534669
min	1.000000
25%	50.000000
50%	65.000000
75%	80.000000
max	255.000000

	Total
count	800.00000
mean	435.10250
std	119.96304
min	180.00000
25%	330.00000
50%	450.00000
75%	515.00000
max	780.00000

Bi-Variate Exploration

Statistical Summary

HP	Hit Points of a Pokemon
Total	Total Points of a Pokemon

Machine Learning Questions
- What is the mutual relationship?
- Can we predict Total given HP?

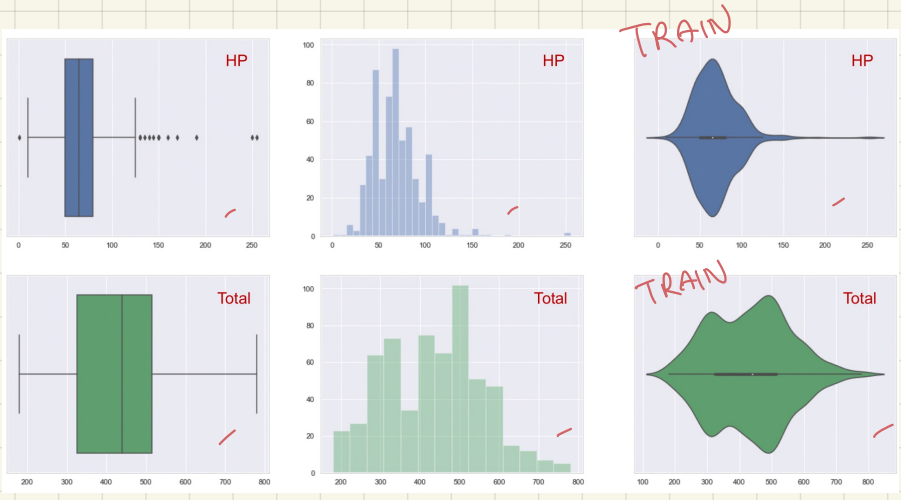
→ Extract 75 % random counts to make training set

Cannot Use same data

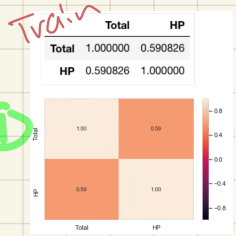
→ 25% for test set

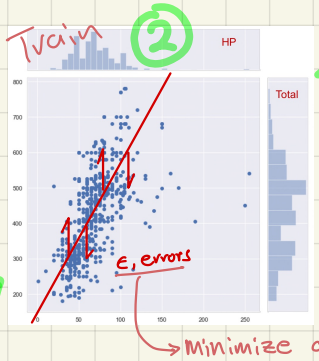
Train	Used to train the model
Test	Used to test the model

The Objectives
- Learn the relationship from Train
- Try to predict the Total on Test



1) Take corr between HP & Total





Statistical Modeling 3

Hypothesize a Linear Model

$$Total = a \times HP + b + \epsilon$$

ϵ: error; a, b: Parameter

Steps in Linear Regression

Guess parameters ***a*** and ***b*** in the model
Predict the values of ***Total*** in Train Data
Calculate the ***Errors*** compared to actual
Tune the parameters to minimize ***Errors***

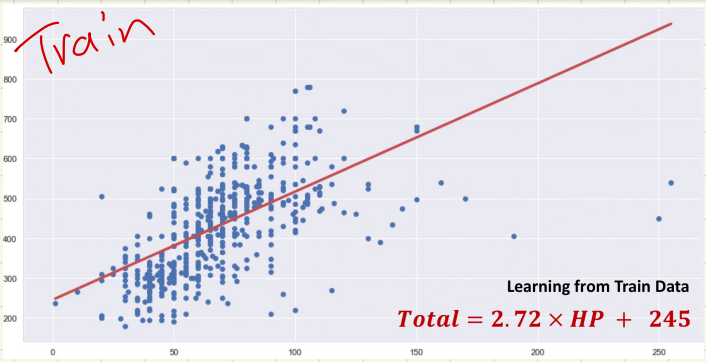
Residual Sum of Squares

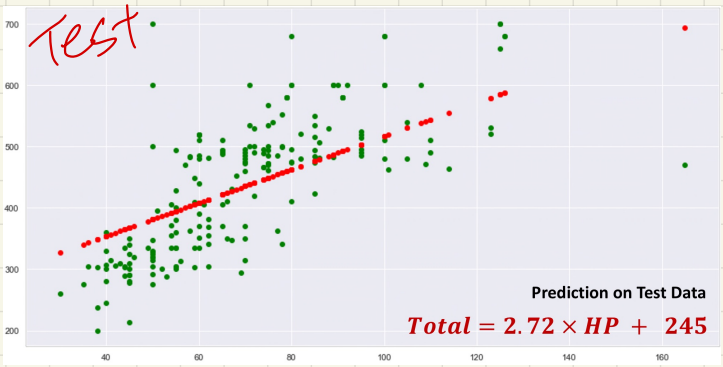
Cost Function to Minimize

$$J(a,b) = \sum (Total - a \times HP - b)^2$$

Total: actual values from training data

$a \times HP - b$: Prediction from estimated line





How to test if the values are good?

Goodness of fit $Total = a \times HP + b + \epsilon$

Explained Variance (R^2)

$$R^2 = 1 - \frac{\sum(Total - a \times HP - b)^2 = RSS}{\sum(Total - \overline{Total})^2 = TSS}$$

total sum of Square

The higher the R^2 the better the Model

Mean Squared Error (MSE)

$$MSE = \frac{1}{n}\sum (Total - a \times HP - b)^2$$

The lower the MSE the better the Model

R^2, explained Variance

$R^2 = 1 - \frac{MSE}{VAR}$, $0 \le R^2 \le 1$

The higher, the better

Residual sum of squares $\times \left(\frac{1}{n}\right)$

: find diff between predicted & actual points
: Square to remove -ve
: Sum & divide by all to average

Also, $\frac{1}{n}(TSS)$, $\frac{1}{n}(\sum(total - \overline{total})^2$ = Variance

Binary Classification

- 2 points only

Total	Legendary
505	False
440	False
273	False
485	False
580	True
384	False
260	False
480	False
495	False
320	False

Data Science

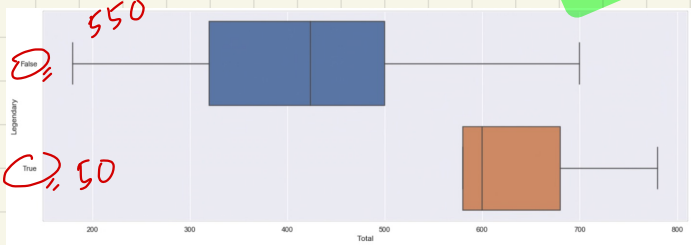
Bi-Variate Exploration

Statistical Summary

Legendary	True or False → Continious
Total	Total Points → Categorical

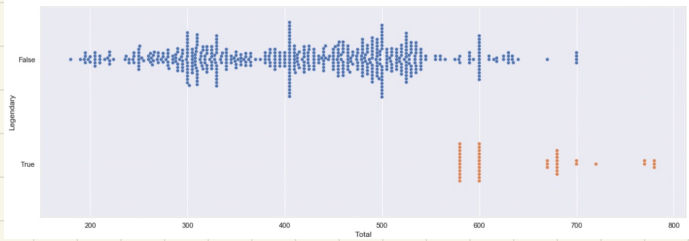
Machine Learning Questions

- What is the mutual relationship?
- Can we predict Legendary by Total?



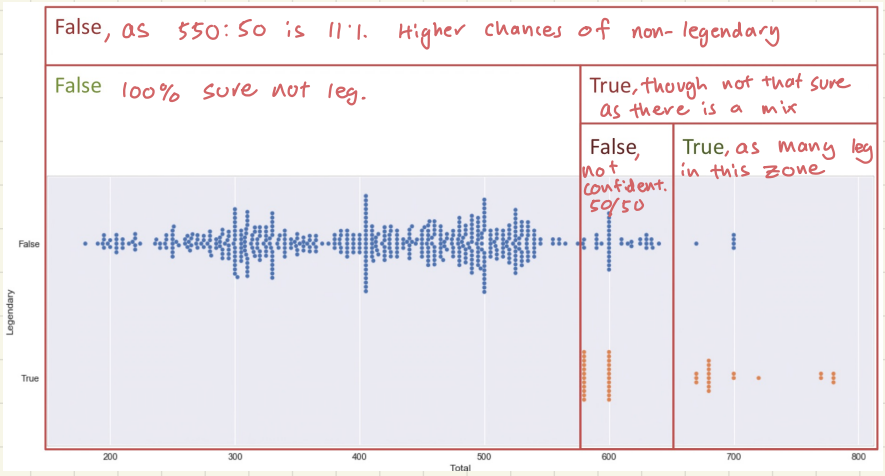
BoxPlot

Train Dataset
75% of the Data

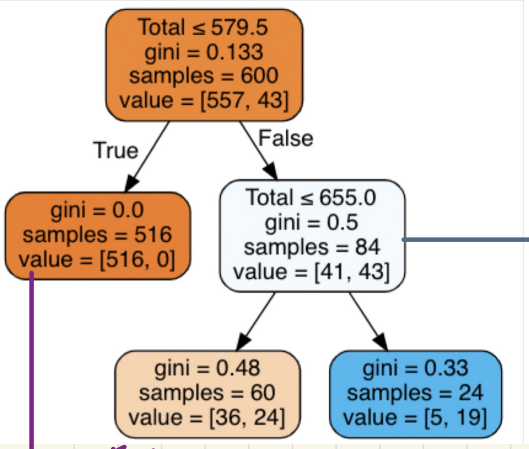


SwarmPlot

Train Dataset
75% of the Data



Decision Tree



Binary Classification

Decision Tree

Partitions made in the Data Space methodically represented using consecutive Binary Decisions

Decision of Partition depends on the Gini Index (metric of misclassification)

$$gini = \frac{x}{n}\left(1 - \frac{x}{n}\right) + \frac{y}{n}\left(1 - \frac{y}{n}\right)$$

Dark red = High no confidence

low confi

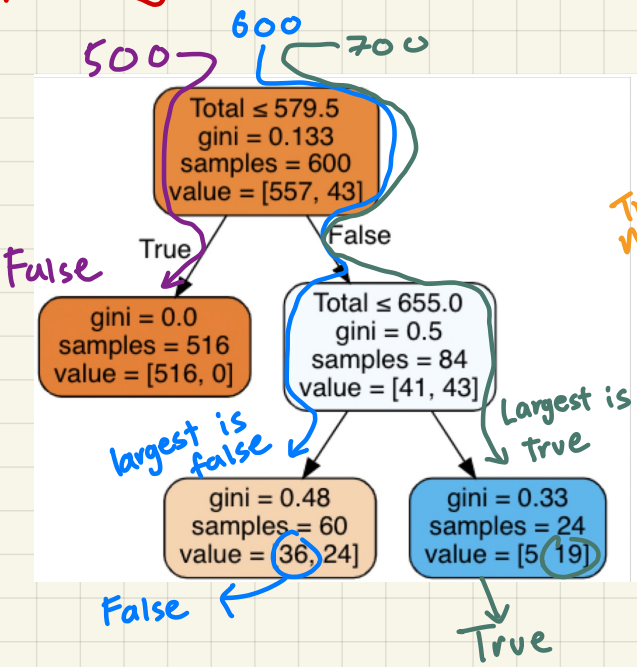
low confi

Dark blue = High confidence

$\frac{516}{516}\left(1-\frac{516}{516}\right) + \frac{0}{516}\left(1-\frac{0}{516}\right) = 0$; 100 % confident

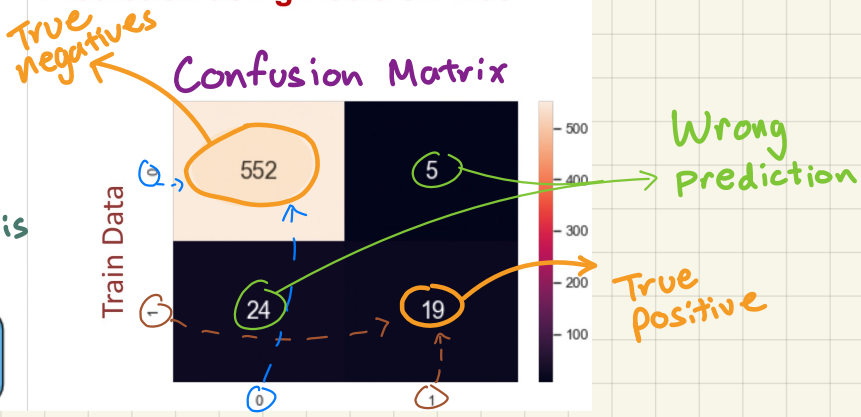
$\frac{41}{84}\left(1-\frac{41}{84}\right) + \frac{43}{84}\left(1-\frac{43}{84}\right) = 0.5$; 50/50 chance

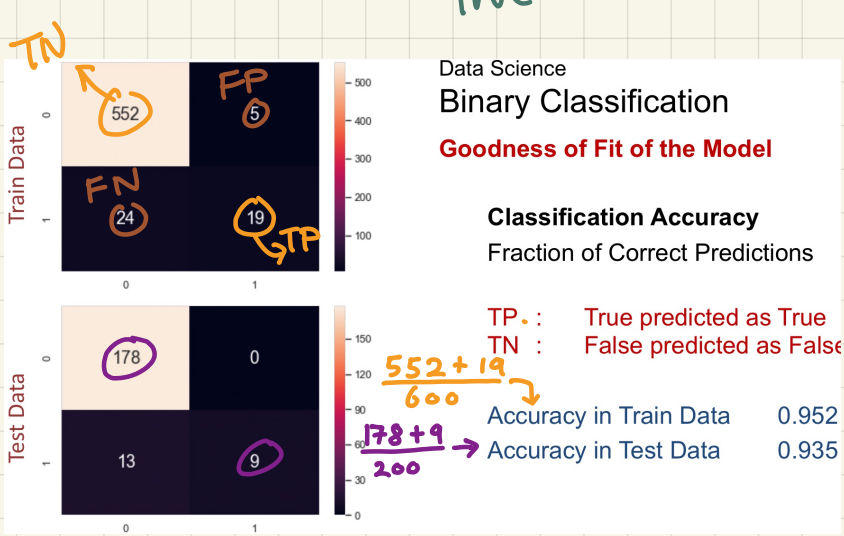
* Only last nodes Gini matter



Binary Classification

Prediction using Decision Tree





Data Science

Binary Classification

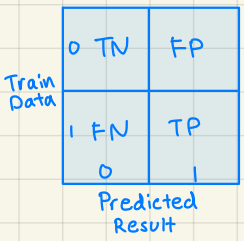
Goodness of Fit of the Model

Classification Accuracy

Fraction of Correct Predictions

TP : True predicted as True
TN : False predicted as False

Accuracy in Train Data	0.952
Accuracy in Test Data	0.935

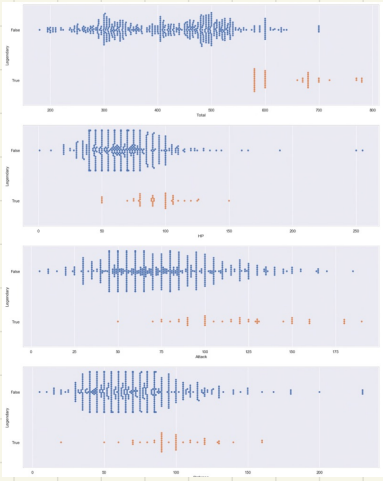


FN : True predicted as False
FP : False predicted as True

FP / TN+FP

FN / TP+FN

Train	$fpr = \frac{5}{557}$,	$fnr = \frac{24}{43}$
Test	$fpr = \frac{0}{178}$,	$fnr = \frac{13}{22}$



Data Science

Binary Classification

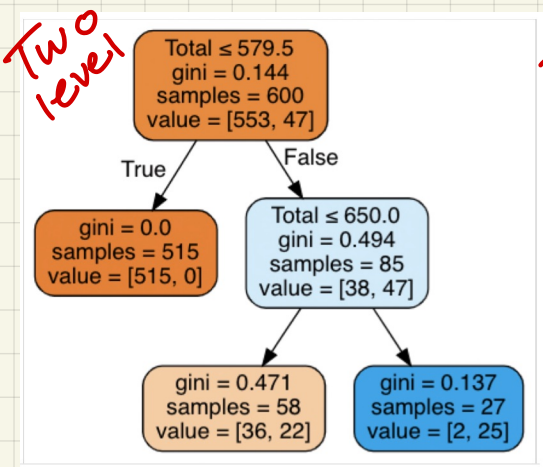
Multi-Variate Decision Tree

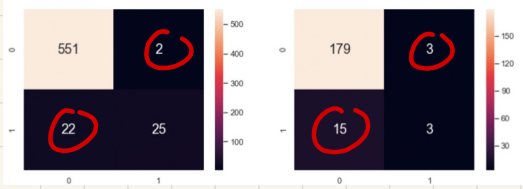
Legendary	True or False
Total	Total Points
HP	Hit Points
Attack	Attack Points
Defense	Defense Points

Swarm Plots per Predictor

Multi-Variate Decision Tree

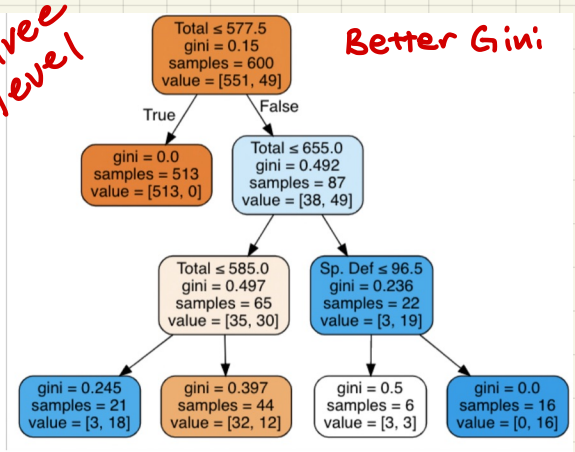
Two level

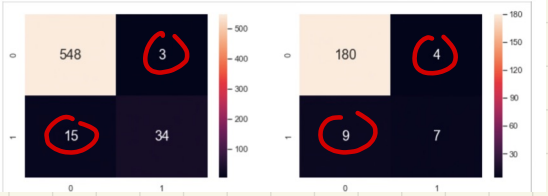




Three level

Better Gini

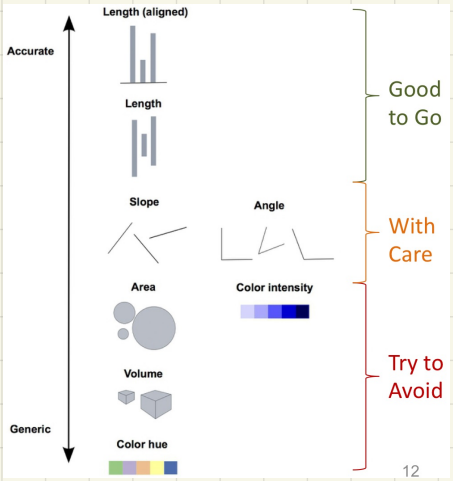


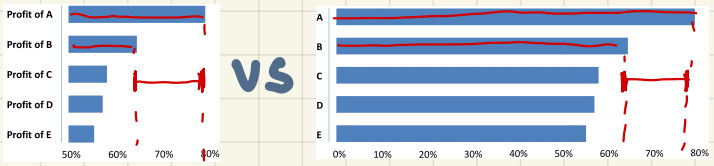


* Error has been reduced by more partition

Data Visualization

1) Convey data through appropraite visual into
2) Establish credibility





Better

2005 Sales Revenue (USD)

Sales Channel	Q1	Q2	Q3	Q4
Direct	383,383	403,939	437,373	538,583
Indirect	283,733	283,833	257,474	258,474
Total	667,116	687,772	694,847	797,057

can't see

Data Ink vs. Non-Data Ink

The data-ink ratio must be as high as possible.

- Tuft

Deviation

Emphasise variations (+/-) from a fixed reference point. Typically the reference point is zero but it can also be a target or a long-term average. Can also be used to show sentiment (positive/neutral/negative).

Example FT uses
Trade surplus/deficit, climate change

Diverging bar

A simple standard bar chart that can handle both negative and positive magnitude values.

Diverging stacked bar

Perfect for presenting survey results which involve sentiment (eg disagree/neutral/agree).

Spine

Splits a single value into two contrasting components (eg male/female).

Surplus/deficit filled line

The shaded area of these charts allows a balance to be shown – either against a baseline or between two series.

Correlation

Show the relationship between two or more variables. Be mindful that, unless you tell them otherwise, many readers will assume the relationships you show them to be causal (i.e. one causes the other).

Example FT uses
Inflation and unemployment, income and life expectancy

Scatterplot

The standard way to show the relationship between two continuous variables, each of which has its own axis.

Column + line timeline

A good way of showing the relationship between an amount (columns) and a rate (line).

Connected scatterplot

Usually used to show how the relationship between 2 variables has changed over time.

Bubble

Like a scatterplot, but adds additional detail by sizing the circles according to a third variable.

XY heatmap

A good way of showing the patterns between 2 categories of data, less effective at showing fine differences in amounts.

Ranking

Use where an item's position in an ordered list is more important than its absolute or relative value. Don't be afraid to highlight the points of interest.

Example FT uses
Wealth, deprivation, league tables, constituency election results

Ordered bar

Standard bar charts display the ranks of values much more easily when sorted into order.

Ordered column

See above.

Ordered proportional symbol

Use when there are big variations between values and/or seeing fine differences between data is not so important.

Dot strip plot

Dots placed in order on a strip are a space-efficient method of laying out ranks across multiple categories.

Slope

Perfect for showing how ranks have changed over time or vary between categories.

Lollipop

Lollipops draw more attention to the data value than standard bar/column and can also show rank and value effectively.

Bump

Effective for showing changing rankings across multiple dates. For large datasets, consider grouping lines using colour.

Distribution

Show values in a dataset and how often they occur. The shape (or 'skew') of a distribution can be a memorable way of highlighting the lack of uniformity or equality in the data.

Example FT uses
Income distribution, population (age/sex) distribution, revealing inequality

Histogram

The standard way to show a statistical distribution - keep the gaps between columns small to highlight the 'shape' of the data.

Dot plot

A simple way of showing the change or range (min/max) of data across multiple categories.

Dot strip plot

Good for showing individual values in a distribution, can be a problem when too many dots have the same value.

Barcode plot

Like dot strip plots, good for displaying all the data in a table, they work best when highlighting individual values.

Boxplot

Summarise multiple distributions by showing the median (centre) and range of the data

Violin plot

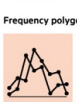
Similar to a box plot but more effective with complex distributions (data that cannot be summarised with simple average).

Population pyramid

A standard way for showing the age and sex breakdown of a population distribution; effectively, back to back histograms.

Cumulative curve

A good way of showing how unequal a distribution is: y axis is always cumulative frequency, x axis is always a measure.

Frequency polygons

For displaying multiple distributions of data. Like a regular line chart, best limited to a maximum of 3 or 4 datasets.

Beeswarm

Use to emphasise individual points in a distribution. Points can be sized to an additional variable. Best with medium-sized datasets

Change over Time

Give emphasis to changing trends. These can be short (intra-day) movements or extended series traversing decades or centuries: Choosing the correct time period is important to provide suitable context for the reader.

Example FT uses
Share price movements, economic time series, sectoral changes in a market

Line

The standard way to show a changing time series. If data are irregular, consider markers to represent data points.

Column

Columns work well for showing change over time - but usually best with only one series of data at a time.

Column + line timeline

A good way of showing the relationship over time between an amount (columns) and a rate (line).

Slope

Good for showing changing data as long as the data can be simplified into 2 or 3 points without missing a key part of story.

Area chart

Use with care – these are good at showing changes to total, but seeing change in components can be very difficult.

Candlestick

Usually focused on day-to-day activity, these charts show opening/closing and high/low points of each day.

Fan chart (projections)

Use to show the uncertainty in future projections - usually this grows the further forward to projection.

Connected scatterplot

A good way of showing changing data for two variables whenever there is a relatively clear pattern of progression.

Calendar heatmap

A great way of showing temporal patterns (daily, weekly, monthly) – at the expense of showing precision in quantity.

Priestley timeline

Great when date and duration are key elements of the story in the data.

Magnitude

Show size comparisons. These can be relative (just being able to see larger/bigger) or absolute (need to see fine differences). Usually these show a 'counted' number (for example, barrels, dollars or people) rather than a calculated rate or per cent.

Example FT uses
Commodity production, market capitalisation, volumes in general

Column

The standard way to compare the size of things. Must always start at 0 on the axis.

Bar

See above. Good when the data are not time series and labels have long category names.

Paired column

As per standard column but allows for multiple series. Can become tricky to read with more than 2 series.

Paired bar

See above.

Marimekko

A good way of showing the size and proportion of data at the same time – as long as the data are not too complicated.

Proportional symbol

Use when there are big variations between values and/or seeing fine differences between data is not so important.

Isotype (pictogram)

Excellent solution in some instances – use only with whole numbers (do not slice off an arm to represent a decimal).

Lollipop

Lollipop charts draw more attention to the data value than standard bar/column – does not have to start at zero (but preferable).

Radar

A space-efficient way of showing value of multiple variables– but make sure they are organised in a way that makes sense to reader.

Parallel coordinates

An alternative to radar charts – again, the arrangement of the variables is important. Usually benefits from highlighting values.

Magnitude

Show size comparisons. These can be relative (just being able to see larger/bigger) or absolute (need to see fine differences). Usually these show a 'counted' number (for example, barrels, dollars or people) rather than a calculated rate or per cent.

Example FT uses
Commodity production, market capitalisation, volumes in general

Column

The standard way to compare the size of things. Must always start at 0 on the axis.

Bar

See above. Good when the data are not time series and labels have long category names.

Paired column

As per standard column but allows for multiple series. Can become tricky to read with more than 2 series.

Paired bar

See above.

Marimekko

A good way of showing the size and proportion of data at the same time – as long as the data are not too complicated.

Proportional symbol

Use when there are big variations between values and/or seeing fine differences between data is not so important.

Isotype (pictogram)

Excellent solution in some instances – use only with whole numbers (do not slice off an arm to represent a decimal).

Lollipop

Lollipop charts draw more attention to the data value than standard bar/column – does not have to start at zero (but preferable).

Radar

A space-efficient way of showing value of multiple variables– but make sure they are organised in a way that makes sense to reader.

Parallel coordinates

An alternative to radar charts – again, the arrangement of the variables is important. Usually benefits from highlighting values.

Part-to-whole

Show how a single entity can be broken down into its component elements. If the reader's interest is solely in the size of the components, consider a magnitude-type chart instead.

Example FT uses
Fiscal budgets, company structures, national election results

Stacked column/bar

A simple way of showing part-to-whole relationships but can be difficult to read with more than a few components.

Marimekko

A good way of showing the size and proportion of data at the same time – as long as the data are not too complicated.

Pie

A common way of showing part-to-whole data – but be aware that it's difficult to accurately compare the size of the segments.

Donut

Similar to a pie chart – but the centre can be a good way of making space to include more information about the data (eg total).

Treemap

Use for hierarchical part-to-whole relationships; can be difficult to read when there are many small segments.

Voronoi

A way of turning points into areas – any point within each area is closer to the central point than any other centroid.

Arc

A hemicycle, often used for visualising parliamentary composition by number of seats.

Gridplot

Good for showing % information, they work best when used on whole numbers and work well in small multiple layout form.

Venn

Generally only used for schematic representation.

Waterfall

Can be useful for showing part-to-whole relationships where some of the components are negative.

Spatial

Aside from locator maps only used when precise locations or geographical patterns in data are more important to the reader than anything else.

Example FT uses
Population density, natural resource locations, natural disaster risk/impact, catchment areas, variation in election results

Basic choropleth (rate/ratio)

The standard approach for putting data on a map – should always be rates rather than totals and use a sensible base geography.

Proportional symbol (count/magnitude)

Use for totals rather than rates – be wary that small differences in data will be hard to see.

Flow map

For showing unambiguous movement across a map.

Contour map

For showing areas of equal value on a map. Can use deviation colour schemes for showing +/- values

Equalised cartogram

Converting each unit on a map to a regular and equally-sized shape – good for representing voting regions with equal value.

Scaled cartogram (value)

Stretching and shrinking a map so that each area is sized according to a particular value.

Dot density

Used to show the location of individual events/locations – make sure to annotate any patterns the reader should see.

Heat map

Grid-based data values mapped with an intensity colour scale. As choropleth map – but not snapped to an admin/political unit.

Flow

Show the reader volumes or intensity of movement between two or more states or conditions. These might be logical sequences or geographical locations.

Example FT uses
Movement of funds, trade, migrants, lawsuits, information; relationship graphs.

Sankey

Shows changes in flows from one condition to at least one other; good for tracing the eventual outcome of a complex process.

Waterfall

Designed to show the sequencing of data through a flow process, typically budgets. Can include +/- components.

Chord

A complex but powerful diagram which can illustrate 2-way flows (and net winner) in a matrix.

Network

Used for showing the strength and inter-connectedness of relationships of varying types.

AI

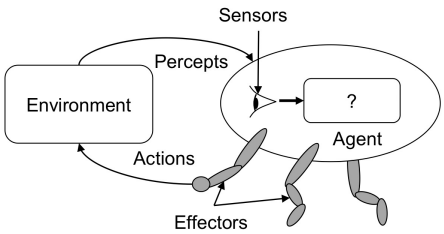
Views of AI fall into four categories:

1. Thinking Humanly
2. Acting Humanly
3. Thinking Rationally
4. Acting Rationally

- Rational behavior: doing the right thing
- The right thing: that which is expected to maximise goal achievement, given the available information
- This course is about designing rational agents
- For any given class of environments and tasks, we seek the agent(s) with the best performance

Agent

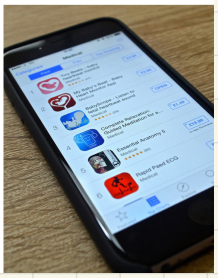
An agent is an entity that
- Perceives through sensors (e.g. eyes, ears, cameras, infrared range sensors)
- Acts through effectors (e.g. hands, legs, motors)



Example: AI CAR, Healthcare

- **Percepts:** video, speed, acceleration, engine status, GPS, radar, …
- **Actions:** steer, accelerate, brake, horn, display, …
- **Goals:** safety, reach destination, maximise profits, obey laws, passenger comfort,…
- **Environment:** Singapore urban streets, highways, traffic, pedestrians, weather, customers, …

- **Percepts**: symptoms, findings, patient's answers, …
- **Actions:** questions, medical tests, treatments,…
- **Goals:** healthy patient, faster recovery, minimise costs, …
- **Environment:** Patient, hospital, clinic, …



Rational Agents

- A rational agent is one that does the right thing
- Rational action: action that maximises the expected value of an objective performance measure given the percept sequence to date
- Rationality depends on
  - performance measure
  - everything that the agent has perceived so far
  - built-in knowledge about the environment
  - actions that can be performed

Autonomous Car

- Do **not** rely entirely on built-in knowledge about the environment (i.e. not entirely pre-programmed)
- Otherwise,
  - The agent will **only** operates successfully when the built-in knowledge are all correct
- Adapt to the environments through experience

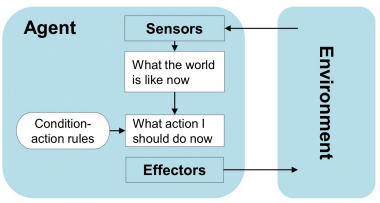
Example: Driverless Car
- Learn to drive in driving center
- Drive at NTU
- Drive on public roads
- Drive in highways
- Drive in City Hall

Levels of Complexity

1) Simple Reflex Agent

Example
If car-in-front-is-braking **then** initiate-braking

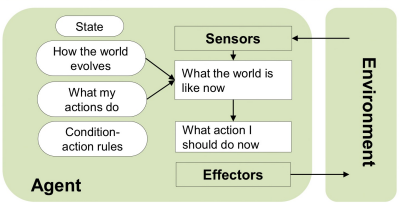
1. Find the rule whose condition matches the current situation (as defined by the percept)
2. Perform the action associated with that rule



2) Reflex agent with State

Example
If yesterday-at-NTU **and** no-traffic-jam-now **then** go-Orchard

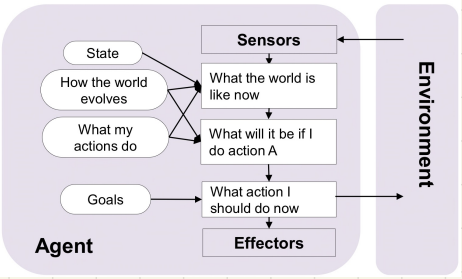
1. Find the rule whose condition matches the current situation (as defined by the percept and the stored internal state)
2. Perform the action associated with that rule



3) Goal Based Agents

Needs some sort of goal information

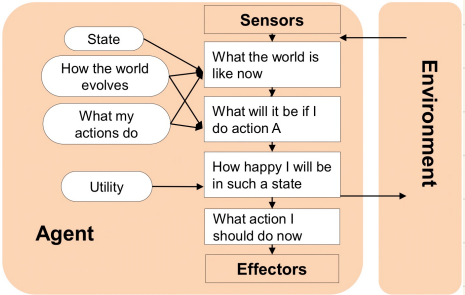
Example: Driverless Taxi
- At a junction (known state), should I go left, right, or straight on?
- Reach Orchard (Destination)?



4) Utility Based Agents

- There may be many action sequences that can achieve the same goal, which action sequence should it take?
- How happy will agent be if it attains a certain state? → Utility

Example: Driverless Taxi
- Go to Orchard (Destination) via PIE? AYE?
- Which one charges lower fare?



Types of environment

Accessible (vs inaccessible)	Agent's sensory apparatus gives it access to the complete state of the environment
Deterministic (vs nondeterministic)	The next state of the environment is completely determined by the current state and the actions selected by the agent
Episodic (vs Sequential)	Each episode is not affected by the previous taken actions
Static (vs dynamic)	Environment does not change while an agent is deliberating
Discrete (vs continuous)	A limited number of distinct percepts and actions

Chess vs Poker

Robot soccer vs GO

Chess vs Q&A Robot

Chess vs Driving

Chess vs Robot soccer

Driving AI

Accessible?	No. Some traffic information on road is missing
Deterministic?	No. Some cars in front may turn right suddenly
Episodic?	No. The current action is based on previous driving actions
Static?	No. When the taxi moves, Other cars are moving as well
Discrete?	No. Speed, Distance, Fuel consumption are in real domains

Chess

Accessible?	Yes. All positions in chessboard can be observed
Deterministic?	Yes. The outcome of each movement can be determined
Episodic?	No. The action depends on previous movements
Static?	Yes. When there is no clock, when are you considering the next step, the opponent can't move; Semi. When there is a clock, and time is up, you will give up the movement
Discrete?	Yes. All positions and movements are in discrete domains

Minesweeper

Accessible?	No. Mines are hidden
Deterministic?	No. Mines are randomly assigned in different positions
Episodic?	No. The action is based on previous outcomes
Static?	Yes. When are you considering the next step, no changes in environment
Discrete?	Yes. All positions and movements are in discrete domains

Slot Machine

Accessible?	No
Deterministic?	No
Episodic?	Yes
Static?	Yes

every roll is different

Design of Problem-Solving Agent

Idea

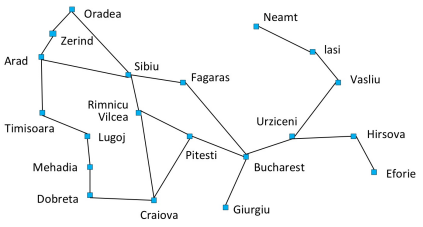
- Systematically considers the expected outcomes of different possible sequences of actions that lead to states of known value
- Choose the best one
  - shortest journey from A to B?
  - most cost effective journey from A to B?

Steps

1. Goal formulation
2. Problem formulation
3. Search process
   - No knowledge → uninformed search
   - Knowledge → informed search
4. Action execution (follow the recommended route)

Goal Based Agent : Example

On holiday in Romania



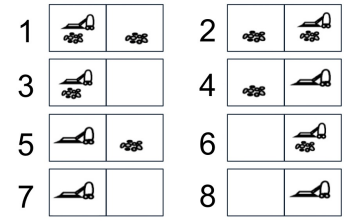
- Currently in Arad (Initial state). Flight leaves tomorrow from Bucharest.
- Goal: be in Bucharest (other factors: cost, time, most scenic route, etc.)
- State: be in a city (defined by the map)
- Action: transition between states (highways defined by the map)

Example: Vacuum Cleaner Agent

- Robotic vacuum cleaners move autonomously
- Some can come back to a docking station to charge their batteries
- A few are able to empty their dust containers into the dock as well

Example: A Simple Vacuum World

Two locations, each location may or may not contain dirt, and the agent may be in one location or the other.



- 8 possible world states
- Possible actions: left, right, and suck
- Goal: clean up all dirt → Two goal states, i.e. {7, 8}

Well-Defined Formulation

Definition of a problem	The information used by an agent to decide what to do

Specification	• Initial state
	• Action set, i.e. available actions (successor functions)
	• State space, i.e. states reachable from the initial state
	  • Solution path: sequence of actions from one state to another
	• Goal test predicate
	  • Single state, enumerated list of states, abstract properties
	• Cost function
	  • Path cost g(n), sum of all (action) step costs along the path

Solution	A path (a sequence of operators leading) from the Initial-State to a state that satisfies the Goal-Test

Measuring Problem-Solving Performance

Search Cost

- What does it cost to find the solution?
  - e.g. How long (time)? How many resources used (memory)?

Total cost of problem-solving

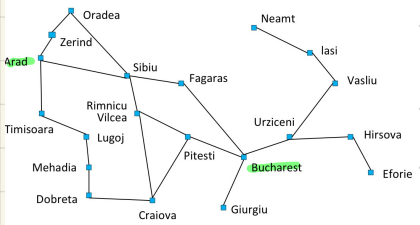
- Search cost ("offline") + Execution cost ("online")
- Trade-offs often required
  - Search a very long time for the optimal solution, or
  - Search a shorter time for a "good enough" solution

2 types of cost

Search → Search for route from NTU → Changi

Execution → Actually driving from NTU → Changi

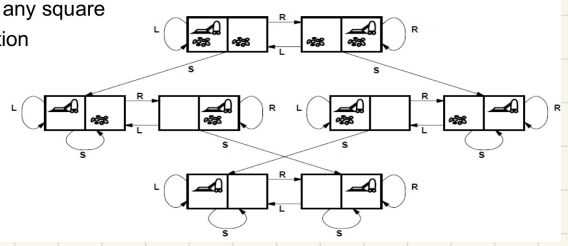
Single-State Problem Example



- Initial state: e.g., "at Arad"
- Set of possible actions and the corresponding next states
  - e.g., Arad → Zerind
- Goal test:
  - explicit (e.g., x = "at Bucharest")
- Path cost function
  - e.g., sum of distances, number of operators executed solution: a sequence of operators leading from the initial state to a goal state

Example: Vacuum World (Single-state Version)

- Initial state: one of the eight states shown previously
- Actions: left, right, suck
- Goal test: no dirt in any square
- Path cost: 1 per action



Example: 8-puzzle

- States: integer locations of tiles
  - number of states = 9!
- Actions: move blank left, right, up, down
- Goal test: = goal state (given)
- Path cost: 1 per move

Start state

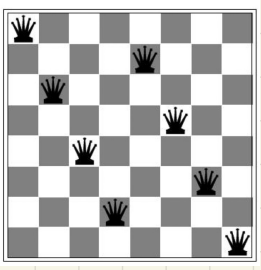
5	4	
6	1	8
7	3	2

Goal state

1	2	3
8		4
7	6	5

Example: 8-queens

- States: Any arrangement of 0 to 8 queens on the board
- Actions: Add a queen to any empty square
- Goal test: 8 queens are on the board, none attacked
- Path cost: Not necessary

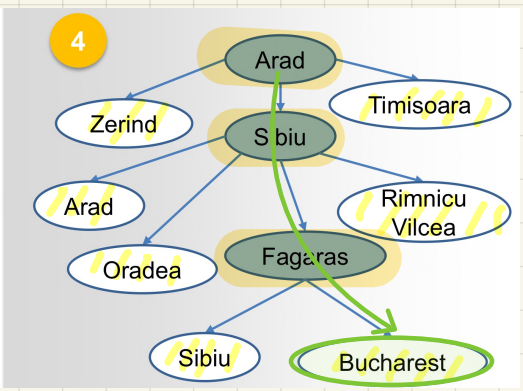


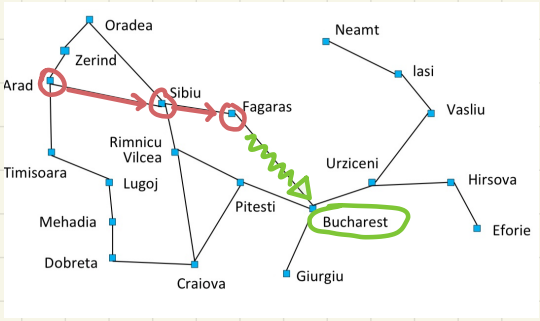
Search !

Search Algorithm

Exploration of state space by generating successors of already-explored states

- Frontier: candidate nodes for expansion
- Explored set





If is goal state, terminate...

Search Strategies

- A strategy is defined by picking the order of node expansion.
- Strategies are evaluated along the following dimensions:

Completeness	Does it always find a solution if one exists?
Time Complexity	How long does it take to find a solution: the number of nodes generated
Space Complexity	Maximum number of nodes in memory
Optimality	Does it always find the best (least-cost) solution?

Domain Knowledge

Branching factor
- Maximum number of successors of any node
- Or average branching factor

max branch factor : deepest node

Ave branch factor : $\frac{\text{node}}{\text{edge}}$

Uninformed search strategies
- Use only the information available in the problem definition
  1. Breadth-first search
  2. Uniform-cost search
  3. Depth-first search
  4. Depth-limited search
  5. Iterative deepening search

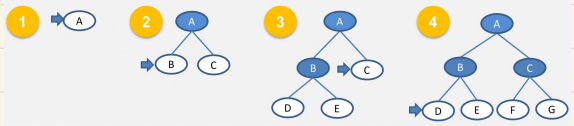
Similar

Informed search strategies
- Use problem-specific knowledge to guide the search
- Usually more efficient

Greedy
A*

(1) Breadth-First Search

Expand shallowest unexpanded node which can be implemented by a First-In-First-Out (FIFO) queue



Denote
- b: maximum branching factor of the search tree
- d: depth of the least-cost solution
- Complete: Yes
- Optimal: Yes when all step costs equally

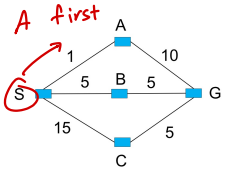
Complexity of BFS

- Hypothetical state-space, where every node can be expanded into b new nodes, solution of path-length d
- Time: $1 + b + b^2 + b^3 + \ldots + b^d = O(b^d)$
- Space: (keeps every node in memory) $O(b^d)$ are equal

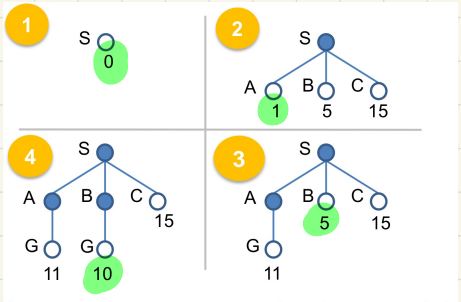
Depth	Nodes	Time		Memory	
0	1	1	millisecond	100	bytes
2	111	0.1	seconds	11	kilobytes
4	11111	11	seconds	1	kilobytes
6	10^6	18	minutes	111	megabyte
8	10^8	31	hours	11	gigabytes
10	10^{10}	128	days	1	terabyte
12	10^{12}	35	years	111	terabytes
14	10^{14}	3500	years	11111	terabytes

(2) Uniform-Cost Search

To consider edge costs, expand unexpanded node with the least path cost g
- Modification of breath-first search
- Instead of First-In-First-Out (FIFO) queue, using a priority queue with path cost $g(n)$ to order the elements
- BFS = UCS with $g(n)$ = Depth(n)



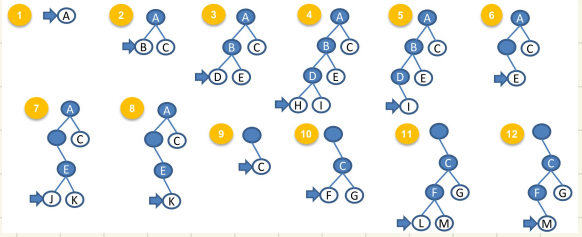
Complete	Yes
Time	# of nodes with path cost g <= cost of optimal solution (eqv. # of nodes pop out from the priority queue)
Space	# of nodes with path cost g <= cost of optimal solution
Optimal	Yes



Here we do not expand notes that have been expanded.

Depth-First Search

Expand deepest unexpanded node which can be implemented by a Last-In-First-Out (LIFO) stack, Backtrack only when no more expansion



```
Function ITERATIVE-DEEPENING-SEARCH(problem) returns a solution sequence
   inputs: problem, a problem

   for depth   0 to ∞ do
    if DEPTH-LIMITED-SEARCH(problem, depth) succeeds then return its result
   end
   return failure
```

Complete	Yes
Time	$O(b^d)$
Space	$O(bd)$
Optimal	Yes

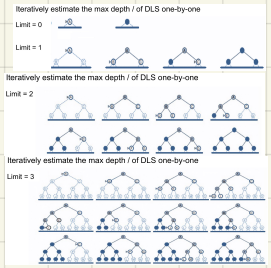
- m: maximum depth of the state space

Complete	• infinite-depth spaces: No • finite-depth spaces with loops: No • with repeated-state checking: Yes • finite-depth spaces without loops: Yes
Time	$O(b^m)$ If solutions are dense, may be much faster than breadth-first
Space	$O(bm)$
Optimal	No

To avoid infinite searching, Depth-first search with a cutoff on the max depth / of a path

Complete	Yes, if $l \geq d$
Time	$O(b^l)$
Space	$O(bl)$
Optimal	No

Iterative deepening



Criterion	Breadth-first	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Time	b^d	b^d	b^m	b^l	b^d	$b^{d/2}$
Space	b^d	b^d	bm	bl	bd	$b^{d/2}$
Optimal	Yes	Yes	No	No	Yes	Yes
Complete	Yes	Yes	No	Yes, if $l \geq d$	Yes	Yes

General Search

Uninformed search strategies

- Systematic generation of new states (→Goal Test)
- Inefficient (exponential space and time complexity)

Informed search strategies

- Use problem-specific knowledge
  - To decide the order of node expansion
- Best First Search: expand the most desirable unexpanded node
  - Use an evaluation function to estimate the "desirability" of each node

Evaluation function

- Path-cost function $g(n)$
  - Cost from initial state to current state (search-node) n
  - No information on the cost toward the goal
- Need to estimate cost to the closest goal
- "Heuristic" function $h(n)$
  - Estimated cost of the cheapest path from n to a goal state $h(n)$
    - Exact cost cannot be determined
  - depends only on the state at that node
  - $h(n)$ is not larger than the real cost (admissible)

Greedy Search

Expands the node that appears to be closest to goal

- Evaluation function $h(n)$:estimate of cost from n to $goal$
- Function Greedy-Search(problem) returns solution
  - Return Best-First-Search(problem, h) // $h(goal) = 0$

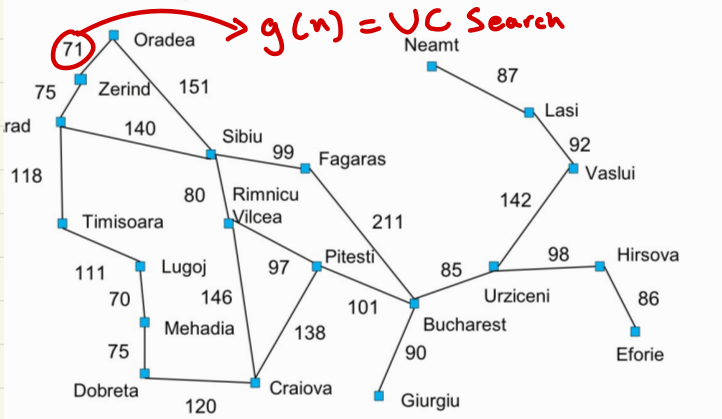
Question: How to estimation the cost from n to goal?

Answer: Recall that we want to use problem-specific knowledge

Greedy Search

Example: Route-finding from Arad to Bucharest

$h(n)$ = straight-line distance from n to Bucharest



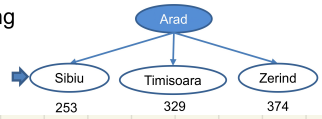
- Useful but potentially fallible (heuristic)
- Heuristic functions are problem-specific

Straight-line distance to Bucharest

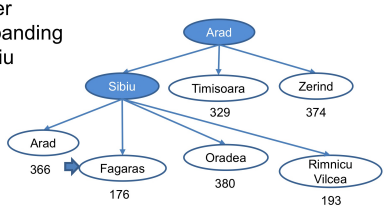
	$h(n)$
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Efoire	161
Fagaras	176
Giurgiu	77
Hirsova	151
Lasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

a) The initial state

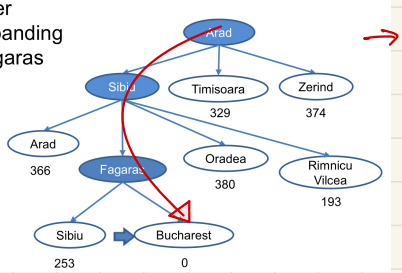
b) After expanding Arad



c) After expanding Sibiu



d) After expanding Fagaras



→ Calculation : 140 + 99 + 211 = 450

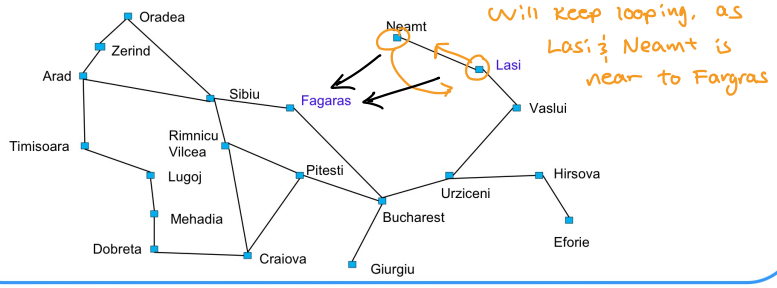
- m: maximum depth of the search space

Complete	No
Time	$O(b^m)$
Space	$O(b^m)$(keeps all nodes in memory)
Optimal	No

↳ But this is worst case. In reality, is better, but Not 100% optimal?

Question: Is this approach complete?

Example: Find a path from Lasi to Fagaras

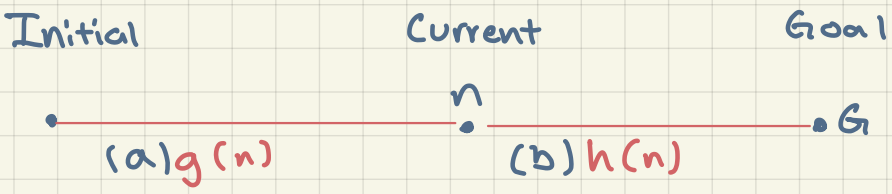


Will keep looping, as Lasi & Neamt is near to Fargras

Answer: No

A* Search

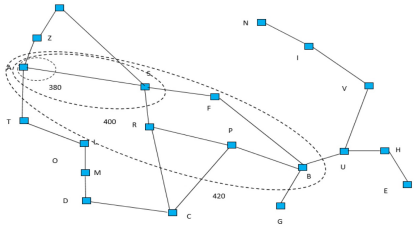
- Uniform-cost search
  - *g(n):* cost to reach n (Past Experience)
  - optimal and complete, but can be very inefficient
- Greedy search
  - *h(n):* cost from n to goal (Future Prediction)
  - neither optimal nor complete, but cuts search space considerably



Idea: Combine Greedy search with Uniform-Cost search

Evaluation function: $f(n) = g(n) + h(n)$

- *f (n):* estimated total cost of path through *n* to goal (Whole
- If g = 0 → greedy search; If h = 0 → uniform-cost sea
- Function A* Search(problem) returns solution
  - Return Best-First-Search(problem, $g + h$)

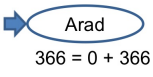


Time	Exponential in length of solution
Space	(all generated nodes are kept in memory) Exponential in length of solution

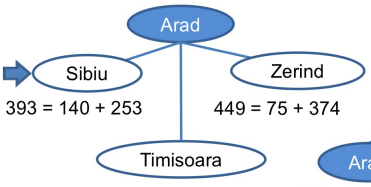
With a good heuristic, significant savings are still possible compared to uninformed search methods

Best-first-search with evaluation function $g + h$

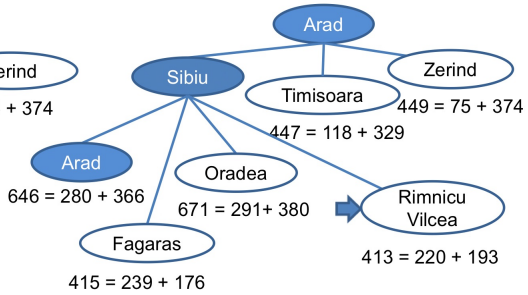
(a) The initial state



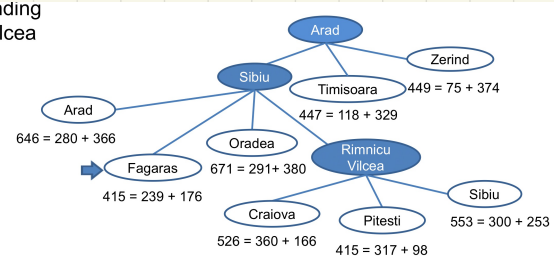
(b) After expanding Arad



(c) After expanding Sibiu

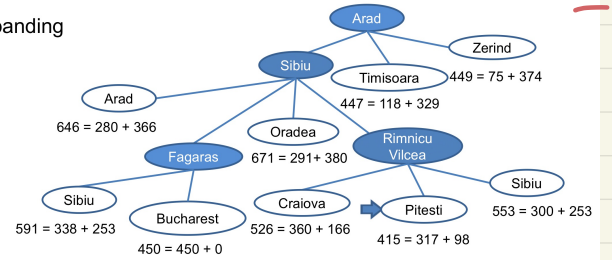


(d) After expanding Rimnicu Vilcea



Best-first-search with evaluation function $g + h$

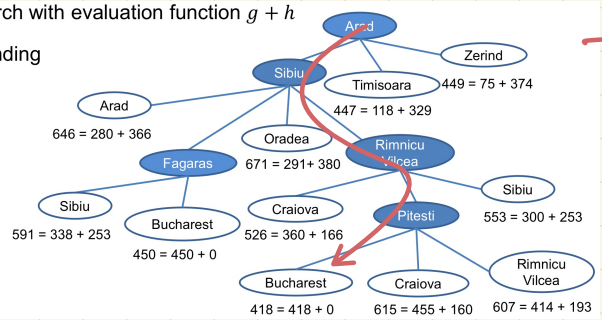
(e) After expanding Fagaras



– Here, don't goal check Bucha as you are not expanding it. Expand Pitesti instead

Best-first-search with evaluation function $g + h$

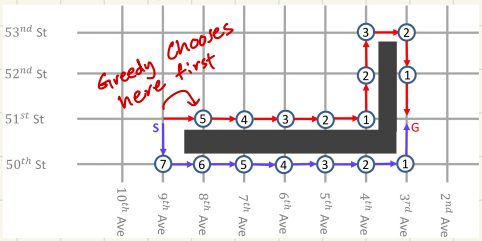
(f) After expanding Pitesti



– See? Optimal Solution

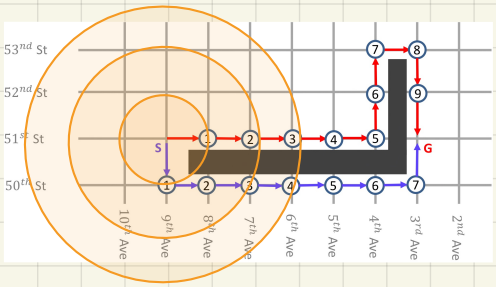
Example: Route finding in Manhattan

Greedy

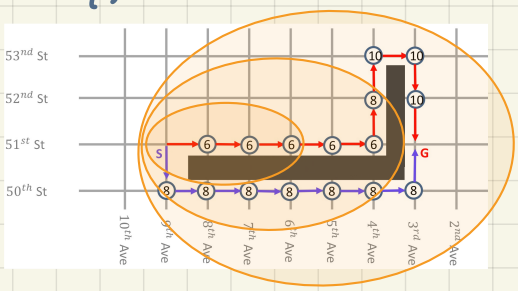


Cost = 10

UCS



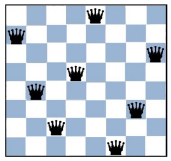
A^*



Constraint Satisfaction Problem

Goal: discover some state that satisfies a given set of constraints

Example: 8-Queens Problem



Example: Cryptarithmetic Puzzle

```
 SEND
+MORE
-----
MONEY
```

Example: Sudoku

							1	3
			7					6
			5		9			
						9		
1		6						
						2		
7	4						5	
	8					4		
				1				

Example: Minesweeper

Real Life Problems

- Assignment problems
  - e.g. who teaches what class
- Timetabling problems
  - e.g. which class is offered when and where?
- Hardware configuration
- Transportation scheduling
- Factory scheduling
- Floor-planning

State

- defined by variables V_i with values from domain D_i

Example: 8-queens

- Variables: locations of each of the eight queens
- Values: squares on the board

Goal test

- a set of constraints specifying allowable combinations of values for subsets of variables

Example: 8-queens

- Goal test: No two queens in the same row, column or diagonal

```
 SEND
+MORE
-----
MONEY
```

- Variables: D, E, M, N, O, R, S, Y
- Domains: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
- Constraints
  - *Y = D + E or Y = D + E - 10*, etc.
  - D ≠ E, D ≠ M, D ≠ N, etc.
  - M ≠ 0, S ≠ 0 (unary constraints: concern the value of a single variable)

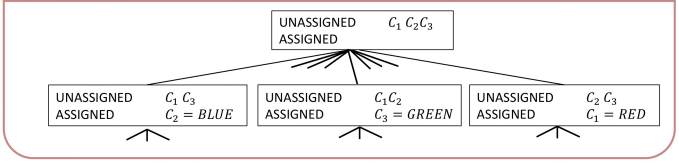
A	B	C
J	2	D
I	3	E
H	G	F

- Variables: The cells
- Domains: {0; 1} representing {safe, mined}
- Constraints: Each cell has a number $m \in \{1, \ldots, 8\}$ indicating the number of mines nearby, so m is equal to sum of value of neighbour cells

- A state of the problem is defined by an **assignment** of values to some or all of the variables.
- An assignment that does not violate any constraints is called a **consistent** or **legal** assignment.
- A **solution** to a CSP is an assignment with every variable given a value (**complete**) and the assignment satisfies all the constraints.

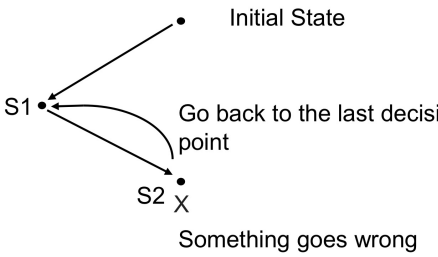
- States: defined by the values assigned so far
- Initial state: all variables unassigned
- Actions: assign a value to an unassigned variable
- Goal test: all variables assigned, no constraints violated

Example: map colouring

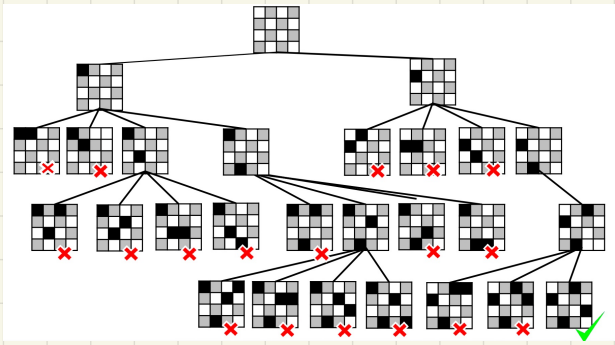


- Number of variables: n
- Max. depth of space: n
- Depth of solution state: n (all variables assigned)
- Search algorithm: depth-first search

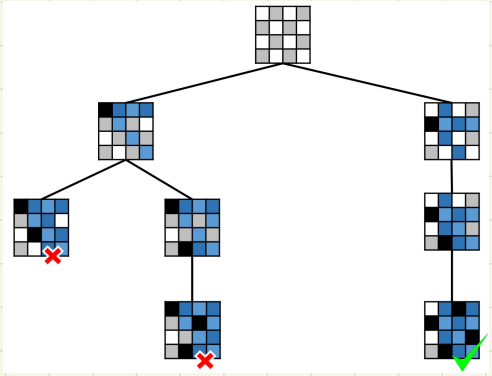
Backtracking search: Do not waste time searching when constraints have already been violated



- Before generating successors, check for constraint violations
- If yes, backtrack to try something else



Constraint propogation → cut the domain of possible values



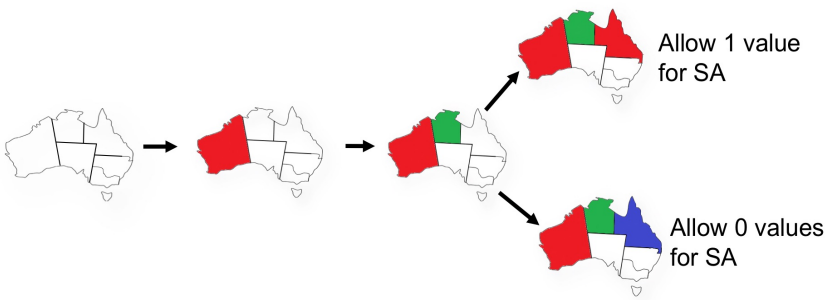
Most Constrained Variable

Or minimum remaining values (MRV) heuristic

Example: map colouring

To reduce the branching factor on future choices by selecting the variable that is involved in the **largest number of constraints** on unassigned variables.

Least Constraining Value

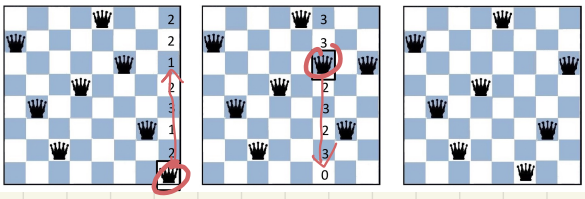


Choose the value that leaves maximum flexibility for subsequent variable assignments

Min conflict assignment (local Search)

A local heuristic search method for solving CSPs

Given an initial assignment, selects a variable in the scope of a violated constraint and assigns it to the value that minimises the number of violated constraints



Abstraction

- Ideal representation of real world problems
  - e.g. board games, chess, go, etc. as an abstraction of war games
  - Perfect information, i.e. fully observable
- Accurate formulation: state space representation

Uncertainty

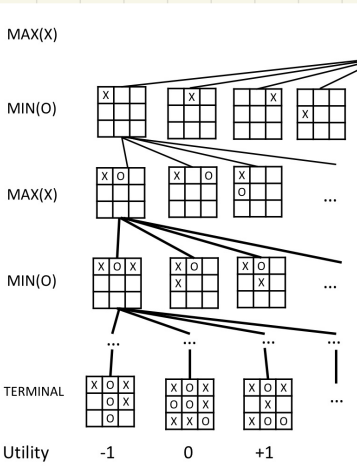
- Account for the existence of hostile agents (players)
  - Other agents acting so as to diminish the agent's well-being
  - Uncertainty (about other agents' actions):
    - not due to the effect of non-deterministic actions
    - not due to randomness

    → Contingency problem

	Deterministic	Chance
Perfect information	Chess, Checkers, Go, Othello	Backgammon, Monopoly
Imperfect information		Bridge, Poker, Scrabble, Nuclear war

Perfect information

- each player has complete information about his opponent's position and about the choices available to him



- The utility value of the terminal state is from the point of view of MAX
- MAX uses the search tree to determine the best move