# CS401 - Applied Data Analysis
# Theoretical Course Summary

**Karl Abdelnour**

# Preface

This summary only concerns the theoretical aspects of ADA as given in autumn 2024. The summary is organised in a slightly different manner to the course but references to respective weeks are given at the beggining of every section.
Everything relating to practical work can be found on the associated Jupyter Notebook. A quick indication of how the practical would be performed and a reference to the Jupyter notebook is given at the beginning of sections if deemed appropriate.

# Contents

# 1 Introduction to Data Science

*This section covers, as described in the title, the basics of data science. Sections 1.1 to 1.3 respectively cover weeks 2 to 4 of the course.*

## 1.1 Basics of Data Science

Data science is, as the name describes, the science of describing, visualising, and analysing data. Data science involves a number of steps. These steps are further described in subsections except for data analysis and visualisation as these take up the rest of the course.

1. The first quite logical step is to acquire data from different data sources
2. The second step involves the initial description of this data as a data model
3. Once the model is obtained, the data must be wrangled to make it easier to analyse
4. The final wrangled data can then be analysed (regression, causal, ML..) and visualised to obtain the sought after results (or not most of the time).

### 1.1.1 Data Sources

In an analogy to cooking, data sources can be seen as the raw ingredients that are used in the culinary art of data science. These sources may be diverse and include server logs, images and videos, books or application data bases. All of these sources are structured to differing degrees, with application databases being highly structured and images, books or videos being completely unstructured.

Most of the data sources described above give their data in structured CSV files or other similar formats.
However, the use of data coming from webpages such as encyclopedic websites involves the necessity to perform web scraping in order to extract the data. An HTTP request is first performed and the webpage returned as an HTML file that can then be parsed for relevant information. Today, an increasing number of large websites require the use of API's to scrape. The data would then be returned in different formats such as JSON, XML or plain text.

### 1.1.2 Data Models

Once the data is obtained from a source, it must be abstracted through the use of a model. **A data model is an abstraction that organises data and standardises relations within the dataset.**

These exist different types of data models whose use depends on the exact nature of the data at hand. These are listed and described below.

- The **flat model** organises data on a one to one basis, treating every individual data point as an individual entry that all share the same attributes. This is the case of server logs or any data stored as a CSV.
- The **relational model** is similar to the flat model with the notable exception that it considers any possible relationship between different entries. Such data is usually given in tables describing both the entity and the relations to others. Entities are linked using an id system. These tables are frequently accessed and processed using SQL.

- Data modelled with the **document model** treats the data hierarchically: every data entry has a number of sub attributes. Different entries for attributes can be given using an id model similar to the relational model.
- Data that can be modelled as a graph is described using the **graphical model**. This model is frequently used in scenarios where the relationship between different entities is the main target of the analysis. It can be combined with other models, most notably the document model, if the analysis of the entities themselves is required.

### 1.1.3 Data Wrangling

The final part of data preprocessing is data wrangling. It is by far the most time consuming yet important step in the data analysis pipeline. **Data wrangling involves the extraction, cleaning and standardisation of the raw data in order to enable any further analysis.**

Most, if not all, data is not "clean": some of the data may be missing, corrupted or represented in the wrong format. Data wrangling seeks to address these shortcomings
The need to wrangle data as well as the nature of the dirty data can be identified by simply looking at the raw data, analysing basics statistics relating to it or visualising the data. In the case of large data samples, wrangling may be performed on specific samples before being generalised to the whole dataset.
it is however worth noting that there exists a significant trade-off between cleaning data and the accuracy of the final analysis as extensive data cleaning can easily lead to extensive data loss.

## 1.2 Data Visualisation

Data visualisation covers all of the different manners through which one may visualise data. This can either permit the communication of results or an initial analysis of data as visualisation can enable insights that may not be seen by standard statistical analyses.
Visualisation is most often performed using graphs or plots. The different plots are first examined before going through a number of good practices for visualising data.

Data visualisation is performed from pandas dataframes using the matplotlib and seaborn library. This is described in more detail in the associated Jupyter notebook. Interactive plots are not used in this course and will thus not be described in the rest of the subsection.

### 1.2.1 Description of Plot Types

Plots can be used to either describe a distribution, a relationship, composition or to perform a comparison. Different types of plots can be used for each of these tasks. Comparisons are usually performed by "stacking" plots (usually histograms, bar charts or scatter/line plots) on a single figure.
A large number of different plots and graphs may be used for data visualisation. These are usually classified by the number of variables that are shown on the plot. The main types of plots are described in table (1) below.

Table 1: Description of the main types of plots

| Plot | N° of Variables | Description |
|---|---|---|
| Histogram | 1 | Histograms are most frequently employed in order to describe the distribution of data over a single variable. They can be very useful in determining the distribution of the data at hand. Histograms exist in two types: line histograms and column histograms. The latter is best for small data sets and is more common. Histograms are frequently augmented by adding statistical descriptions of the data (see 1.3). |
| Box plot | 1 | Box plots are an alternative to histograms. They have the advantage of intrinsically indicating quartiles and any outlying data. |
| Scatter plot | 2-3 | Scatter plots are usually employed to demonstrate the relationship between two or more variables in a dataset. They are quick and easy to set up and can give an initial indication as to any relationship between variables. Scatter plots are frequently augmented by adding statistical descriptions of the data (see 1.3). |
| Line plot | 2-3 | Line plots are an extension of scatter plots. These are employed when the relationship between any two variables is functional after binning or aggregating. |
| Heatmap | 2-3 | Heatmaps are a multi-dimensional extension of histograms. Colour gradients are used to indicate the frequency or counts of data entries with defined characteristics. |
| Plot matrices | 3+ | Plot matrices are matrices of different plots (usually scatter or line plots) on different variables. Each row or column corresponds to a specific variables and are useful yet less easily readable for comparing data. |
| Stacked plots | 3+ | Stacked plots refer to any type of plot that contains different plots that are stacked on top of one another. These are very frequently used for comparison. |

### 1.2.2 Good practices in data visualisation

A number of good practices can, when employed, significantly enhance the visualisation of data. A non exhaustive list of these practices is given below.

- Magnitude perception is very important. Without numbers, it can be most readily achievable through position, then length, area, volume and finally colour hue/saturation in the given order. When using colours, one must ensure that the colours can be easily distinguished from one another.

- Axes are also very important. They must be consistent in their graduations, labelled and be logarithmic when appropriate. The latter point is especially true for large distributions or data that follows a power law.

- Colours are very frequently used in graphs. When comparing different entities across different graphs, the colours used must be consistent. Additionally, sequential, diverging or categorical

data would be described using different colour palettes. It is also always a good idea to use colour-blind safe colours (or to at least add the option to use them).

- Error bars and other basic statistical descriptions of error and deviation are always a plus (see 1.3).
- The data shown in charts must be kept to a minimum: only what is necessary must be shown on a plot as any superfluous data can make the plots much more difficult to read.
- visual contrasts in colour, shape or size are a great way to convey the message of a plot and should be employed when possible.

It is always advisable to use common sense when evaluating one's own data visualisations.

## 1.3 Data Description

Correctly describing data is quite important. Descriptions can involve basic statistics, uncertainty quantifications or determining the relationship between different variables. These are examined individually in the paragraphs below.

### 1.3.1 Descriptive Statistics

Descriptive statistics refer to all types of statistics regarding the average, mean, median, quartiles etc... of data. Their use must be adapted to the distribution of the data described in further detail at the end of the paragraph.
These basic statistics can be divided into two main categories.

- **Robust** statistics that are not sensitive to outlying data such as the median or quartile
- **Non-robust** statistics that are such as the mean, minimum, maximum or standard deviation.

There exist different types of means such as the root mean squared, arithmetic, geometric or harmonic means.

Data distribution refers to the way values in a dataset are spread out or arranged. It describes the frequency of data points within different ranges of values and is often visualized using charts such as histograms, box plots, or probability distribution curves. The latter element can be used to describe some of the major distributions listed in table (2) below.

Table 2: Description of the main types of probability distributions

| Distribution | Law | Description |
|---|---|---|
| Normal | $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$ | The all mighty distribution. |
| Poisson | $P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$ | Poisson's distribution of counts that occur at a certain "rate"; e.g., number of visits to a given website in a fixed time interval. |
| Binomial | $P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$ | The binomial law is very often used for events with two outcomes such as flipping a coin. |
| Exponential | $f(x) = \lambda e^{-\lambda x}, \quad x \geq 0$ | The exponential law is used to describe the interval between two such events. |
| Power | $P(x) = Cx^{-\alpha}, \quad x \geq x_{\min}$ | The power law is frequently used for frequency or count analyses. It has a similar use case to the Zipf, Pareto or Yule distribution laws. |

**In general, the more heavy tailed a distribution is (power law, skewed normal distribution...), the more important it is to use robust statistics to describe them.**

### 1.3.2 Uncertainty Quantification

The quantification of uncertainty is very important as it indicates how reliable an analysis or data may or may not be. It is most frequently represented on plots in the form of error bars.
Uncertainty is quantified in two main ways described below.

**Hypothesis testing** represents the first major approach to quantify uncertainty. Hypothesis testing is a statistical method used to evaluate a claim or hypothesis about a population parameter based on sample data. The process involves stating the null hypothesis ($H_0$), which assumes no effect or difference, and the alternative hypothesis ($H_a$), which represents the presence of an effect or difference. A significance level ($\alpha$), commonly 0.05 or 0.01, is set to determine the threshold for rejecting $H_0$. A statistical test is then selected based on the data type, and a test statistic is calculated. The test statistic is a numerical value calculated from the sample data during a hypothesis test. It measures how far the sample data is from the null hypothesis in terms of a specific distribution.
Hypothesis tests return a p-value, which indicates the probability of obtaining a result at least as extreme as the observed one assuming $H_0$ is true, is compared to $\alpha$. If $p \leq \alpha$, $H_0$ is rejected. Otherwise, it is not possible to reject $H_0$. It is important to interpret p-values correctly; a low p-value suggests statistical significance but does not imply practical significance. Key considerations include the effect size (magnitude of the effect), the power of the test (probability of correctly rejecting $H_0$ when $H_a$ is true) and meeting the assumptions of the chosen test to ensure valid results.

**Confidence intervals** (CI) are the second major approach to quantify uncertainty in data. A confidence interval is a range of values derived from sample data that is likely to contain the true population parameter. It provides an estimate of the uncertainty in a measurement and is associated with a confidence level, typically 95% or 99%, which indicates the proportion of intervals that would contain the parameter if the sampling were repeated many times. A CI consists of a point estimate,

such as the sample mean ($\bar{x}$) or proportion ($\hat{p}$), and a margin of error (MOE), calculated as $z^* \cdot \frac{\sigma}{\sqrt{n}}$ or $t^* \cdot \frac{s}{\sqrt{n}}$, where $z^*$ is the critical value from the standard normal distribution, $t^*$ is the critical value from the t-distribution, $\sigma$ or $s$ is the population or sample standard deviation, and $n$ is the sample size. For means, the confidence interval is given by $\bar{x} \pm z^* \cdot \frac{\sigma}{\sqrt{n}}$ or $\bar{x} \pm t^* \cdot \frac{s}{\sqrt{n}}$, and for proportions, it is $\hat{p} \pm z^* \cdot \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$. A 95% CI implies that if the process were repeated, 95% of the constructed intervals would contain the true parameter. Wider intervals result from higher confidence levels or smaller sample sizes. For valid intervals, data should follow a normal distribution or the sample size should be large.

### 1.3.3 Correlation of multiple variables

Correlating different variables involves determining how the evolution of one variable affects that of another. Correlation is quantified using a correlation coefficient.

The most common type of correlation measurement is Pearson's correlation coefficient that describes the linear correlation between two variables. A coefficient of -1 indicates a perfect negatively linear relationship, 1 a perfectly positive linear relationship while 0 indicates that the two values are very poorly correlated with one another.
While Pearson's coefficient is very useful, it only describes linear correlations and can thus not be used in all situations. Spearman's correlation coefficient is an alternative to Pearson's as it is a monotonic (i.e the direction of the association remains the same over time) rank correlation coefficient (that can still be used for linear dependences).

While correlation coefficients are very useful, they are very dangerous if not used properly. One must first beware of Simpson's paradox where a trend that appears in subsets disappears when brought together in a large subset.
**And, of course, correlation $\neq$ causation!**

# 2 Basic Data Analysis

*This section covers, as described in the title, the basics of data science. Sections 2.1 and 2.2 respectively cover weeks 5 and 6 of the course.*

## 2.1 Regression Analysis

The programming aspects of regression analysis will be described in the Jupyter notebook. In general, statsmodel and scikit-learn will be the most frequently used librairies.

### 2.1.1 Basics of Linear Regression

Let us define n points ($X_i$, $y_i$) where $X_i$ is a k dimensional vector of features and $y_i$ a scalar outcome. Linear regression seeks to find a line defined by the optimal coefficient vector $\beta = (\beta_1, ..., \beta_k)$ such that the line defined by the equation (1) below

$$y_i = \sum_{i=1}^{k} \beta_i \cdot X_i + \varepsilon_i \tag{1}$$

has the smallest possible error term given in the equation above by $\varepsilon_i$.
The optimal coefficient vectors are found by using the least-squares criterion defined in equation (2) below.

$$OLS = \sum_{i=1}^{k} y_i - X_i\beta \tag{2}$$

The optimal $\beta$ vector is the one that minimises the OLS (optimal least squares).

Linear regression is a widely used tool in statistics and data analysis. It can be used for prediction (see section 3), causal analysis (see 2.2) or mean outcome comparison in descriptive data analysis which is what is described below.

It is important to note that a number of assumptions are made when using linear regression on a dataset. These are enumerated below.

1. The analysis must be valid: the outcome measure should accurately reflect the phenomenon of interest, the model should include all relevant predictors and the model should generalize to cases to which it will be applied
2. The model should only involve linear predictors (they themselves may be any function of raw inputs)
3. All errors must be independent (i.e. there must be no interaction within the data points that are assumed to be independent)
4. The variance of errors must be equal.
5. The errors must follow a Gaussian distribution

Points 4 and 5 are not as important in practice.

### 2.1.2 Mean Outcome Comparison with Linear Regression

When comparing the average outcome using linear regression, cases with binary or continuous predictors must be treated separately.
In both cases, the coefficient $\beta_1$ will correspond to the value for a predictor of 0. Graphically, this may be seen as the intercept.

Binary predictors involve cases where the predictor takes discrete values (0 and 1 per example). The data would graphically be represented as "lines" along a given x-coordinate. Taking the average of both of the lines is not possible due to the necessity of fulfilling the least-squares criterion.

In case one must deal with continuous predictors, the same general principles apply. The major difference is that an estimate of the average outcome - and not an exact value - is obtained unlike what was the case for binary predictors.

In any case, it is important to note that both binary and continuous predictors can be combined. Additionally, if one wanted to include the interaction between two predictors $X_1$ and $X_2$, then one would simply obtain an additional term of the form $\beta_i \cdot X_1 \cdot X_2$.

### 2.1.3 Uncertainty quantification

As stated in the previous paragraph, linear regression only give estimations and not exact values of average outcomes. As such, a certain error or uncertainty is to be expected when performing any linear regressions.

Uncertainty in linear regression is described through two main parameters.
The first is the **residual** which describes "how far away" a given data point is from the line. It is given by equation (3) below.

$$r_i = y_i - X_i\hat{\beta} \tag{3}$$

The second, more frequently used one is the $R^2$. The $R^2$ parameter, or the coefficient of determination, describes the proportion of variance in the dependent variable (y) that is explained by the independent variable(s) in a regression model. It ranges from 0 to 1, where 0 indicates no explanatory power and 1 indicates that the model perfectly explains the variance in the data. It can be mathematically defined by equation (4) below.

$$R^2 = 1 - \frac{\hat{\sigma^2}}{\hat{s_y^2}} \tag{4}$$

where $\hat{\sigma^2}$ is the variance of the residuals and $\hat{s_y^2}$ the variance of the outcomes.

### 2.1.4 Predictor and Outcome Transformations

In different cases, it may be necessary to transform the values of the predictors and the outcomes after having performed a linear regression. Such transformations can take different forms described in the rest of this paragraph.

In some cases, the linear regression returns parameters with specific units. The predictor's units may thus be transformed in order to make it applicable to other data sets without actually modifying

the model's accuracy or validity.

Another transformation is the mean centring of the outcome. This involves computing the mean value of a predictor over all data points, and subtract it from each value of that predictor. The mean outcome will then have a mean of 0, thus giving a convenient interpretation of coefficients $\beta_j$ of the main predictors. What was once the intercept in the old line curve is now the model's mean giving the estimated mean outcome when every predictor takes its mean value. The children's IQ problem presented in the slides gives a good tangible example of mean centring.

Following mean score standardisation, a further transformation in the form of z score standardisation may be performed. It simply involves dividing all predictors by their standard deviations. This enables the comparison of predictors in a single model with different units as all predictors end up having the same unit. The relative importance of different predictors to an outcome can thus be determined.

Logarithmic outcomes are useful when the outcome variable $y$ follows a heavy-tailed distribution and can only take non-negative values. Practically, they transform an additive linear model into a multiplicative one by applying a logarithm to $y$: $\log y_i = b_0 + b_1 X_{i1} + b_2 X_{i2} + \cdots + \epsilon_i$. Exponentiating both sides yields $y_i = e^{b_0 + b_1 X_{i1} + b_2 X_{i2} + \cdots + \epsilon_i} = B_0 \cdot B_1^{X_{i1}} \cdot B_2^{X_{i2}} \cdot E_i$, where $B_j = e^{b_j}$. In this context, an additive increase of 1 in predictor $X_1$ is associated with a multiplicative increase in the outcome of $B_1 = e^{b_1}$. For small values of $b_1$ (e.g., $b_1 \approx 0$), $b_1$ can be approximated as the relative increase in outcomes, since $e^{b_1} \approx 1 + b_1$. For example, if $b_1 = 0.05$, then $B_1 \approx 1.05$, meaning a one-unit increase in $X_1$ corresponds to a 5% increase in the outcome.

### 2.1.5 Other Elements of Linear Regression

There exist other generalised linear regression models such as the logistic regression used for binary outcomes (not predictors!) or the Poisson regression specialised in tackling non-negative integer outcomes such as counts of frequencies.

Linear regression can, as previously mentioned, be used to perform a causal analysis (see 2.2 and the final two slides or so of week 5 on the treatment effect).

## 2.2 Causal Analysis

Believe it or not, but correlation $\neq$ causation! The following paragraphs describe how one can perform a causal analysis of data (more precisely observational data of studies and such).

### 2.2.1 Randomised Studies and Causal Relationships

Causal analysis seeks to identify cause-and-effect relationships within data, especially when controlled, randomised experiments are not feasible due to ethical, financial, or practical constraints. Randomised experiments are the best possible experiments that can be conducted.

In most observational studies, confounders which affect both treatment probability and outcome will skew the results of the study and create bias (e.g, the motivation to quit smoking in a study on a cure for smoking). The relationship between these confounders and the outcome of the study can be visualised on causal diagrams, examples of which may be found in the week 6 slides.

### 2.2.2 Matching

Matching is one of the main methods used to reduce the effects of confounders and the bias in an observational study. In the smoking cure example given in the previous paragraph, matching would ensure that as many people with the motivation to quit smoking would be placed in both the treated and control groups, thereby nullifying the confounder's effect.
The goal is to pair treated and control subjects with similar observed covariates, ensuring that differences in outcomes can be attributed to treatment rather than those covariates.

However, two main challenges, described below, come up when attempting to perform matching.

1. **Unobserved covariates** can persist through the matching process and still bias the results of the study. One way to deal with this is the naive approach which is to simply ignore their possible existence.

2. No two people will ever be identical. As such, performing an exact matching is impractical and most of the time impossible as it would lead to a **combinatorial explosion**.

Both of these issues can be respectively mitigated through the use of sensitivity analysis and propensity score matching.

**Sensitivity analysis** helps assess the robustness of conclusions against any potential biases. Sensitivity analysis introduces a parameter $\Gamma$ to quantify how much the odds of treatment assignment could differ between matched pairs due to unobserved confounders. Specifically, it assumes that for two matched subjects $i$ and $j$, the odds ratio of treatment assignment is bounded by:

$$\frac{1}{\Gamma} \leq \frac{\Pr(\text{Treated}_i \mid \text{Covariates})}{\Pr(\text{Treated}_j \mid \text{Covariates})} \leq \Gamma. \tag{5}$$

A higher $\Gamma$ (e.g., $\Gamma > 6$) implies that unobserved factors would need to have an implausibly large effect to overturn the study's conclusions, providing confidence in the causal inference. Conversely, if small values of $\Gamma$ invalidate the results, the conclusions are more sensitive to unobserved confounders.

On the other hand, **propensity scores** simplify the matching process by estimating the probability of treatment assignment given observed covariates, often using methods like logistic regression. The key feature of propensity scores is their *balancing property*, which ensures that within groups of similar propensity scores, the distribution of observed covariates is comparable between treated and control subjects. This property allows researchers to form matched pairs or groups that approximate the covariate balance achieved in randomized experiments, even when exact matching is not feasible. For matching, algorithms such as the Hungarian algorithm are employed to find approximate matches by minimizing the difference in propensity scores, thereby optimizing match quality. Formally, the propensity score is defined as:

$$\text{PS}(x) = \Pr(\text{Treated} = 1 \mid \text{Covariates} = x), \tag{6}$$

where $x$ represents the observed covariates.

# 3 Machine Learning in Data Science

*This section covers the machine learning part of the course. Section 2.1 corresponds to week 8 of the course while sections 2.2 and 2.3 respectively cover weeks 7 and 9.*

Machine learning is a subset of artificial intelligence (AI) that focuses on creating algorithms that enable computers to learn patterns and make predictions or decisions based on data without being explicitly programmed. It involves training models on historical data and using those models to analyse new, unseen data.
Features are the input variables or attributes that describe the data and are used by the model to make predictions. For example, in a house price prediction model, features might include the number of bedrooms, square footage, and location. Targets (or labels) are the output variables the model is trying to predict or classify. In the house price example, the target would be the price of the house.
Machine learning is either supervised or unsupervised depending on whether or not the features and targets are known in advance.

As was the case for linear regression, scikit learn and statsmodel are used for machine learning. The programming aspect of the course is described in the associated Jupyter notebook.

## 3.1 Applied Machine Learning

Bias and variance from week 7 here Standard machine learning is performed following a standardised pipeline:

1. The first step is to **collect the data** that will be used. This step is the most extensive one and may require labelling, discretising or standardising the data. This step will be dependent on the use of either supervised or unsupervised learning.

2. Once the data is selected, the next step requires **selecting the model** and running the algorithms. This step will be described in further detail in the next two sections covering supervised and unsupervised learning.

3. The final step is to assess the model's performance which is quite self-explanatory.

Paragraph 3.1.1 describes the different steps of data collection while paragraph 3.1.2 explores different methods of assessing model performance.

### 3.1.1 Data Collection & Feature Engineering

The first step in machine learning is data collection, where features (attributes describing the data) and class labels (targets) are identified. Features can be continuous, ordered (e.g, agree, neutral, disagree or categorical (e.g, country or gender).

If class labels are unavailable, then labelling must be performed. Data Labelling Labelling data is often challenging, time-consuming, and requires domain expertise. Crowdsourcing has become a common method, allowing tasks to be distributed to a mix of amateurs and experts. Quality control in crowdsourcing includes:

- Aggregating labels through methods like majority voting or game-theoretic techniques (e.g., prediction markets).

- Embedding "honeypot" tasks with known answers.

Feature engineering is critical for improving model performance. It may involve removing irrelevant features, generating new ones, or transforming them through different processes.
One such process is discretisation where continuous features are converted into categorical bins, useful for decision trees or when non-linear decision boundaries are required. Discretisation may be performed through unsupervised methods such as:

- Equal width where the range of features is divided into equal-sized intervals.
- Equal frequency where each bin is ensured to have an equal number of observations

Supervised methods using statistical tests like the chi-squared test to determine binning based on class relevance may also be performed.

Regardless of whether or not discretisation was performed, a final feature selection must be performed feature selection aims to reduce the number of features while retaining those most predictive of the target variable. This enhances interpretability, reduces the risk of overfitting, and improves computational efficiency. Feature selection may be performed "offline" by filtering features during preprocessing or online by performing an iterative feature selection. Offline selection methods include using correlation, mutual information or performing chi-squared independence tests.
Online selection can be performed by greedy iterative approaches: forward selection is performed by starting with no features and greedily adding them while backwards elimination does the opposite. The feature selection ceases when an optimum is reached. While computationally expensive, it remains the best method of performing feature selection due to its inclusion of feature interactions unlike offline selection.

Finally, the data may need to be standardised and normalised. This final process involves adjusting feature values to a common scale, preventing features with larger ranges from dominating the model. Standardisation may be performed through the following methods:

- Min-max scaling by rescaling the data to be in a range of [0,1]
- Standardisation the data by centring it around 0 with a standard deviation of 1
- Performing logarithmic scaling in case heavy-tailed distributions are handled.

### 3.1.2 Bias, Variance & Model Assessment

Model assessment evaluates the performance of a fixed model on unseen data, typically using a held-out test set. Metrics for evaluation are derived from the confusion matrix, which includes:

- True positives (TP), corresponding to correctly predicted positive cases
- True negatives (TN), corresponding to correctly predicted negative cases
- False positives (FP), corresponding to falsely predicted positive cases
- False negatives (FN), corresponding to falsely predicted negative cases

The entries of the confusion matrix serve as the building blocks of the most commonly used metrics for model evaluation. These include:

- Accuracy evaluates the fraction of correct predictions,

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

- Precision computes the fraction of correct positive predictions,

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- Recall returns the fraction of actual positives that have been correctly identified, Recall = $\frac{\text{TP}}{\text{TP+FN}}$
- The F1 score that is simply the harmonic mean of precision and recall,

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Other metrics include the ROC Curve that plots the recall against the false positive rate, providing a graphical view of classifier performance. The area under the curve, known as the ROC-AUC, returns a single number ranging from 0.5 (random) to 1.0 (perfect) to rate the quality of a classifier.

The bias-variance tradeoff is a fundamental concept in machine learning that explains the balance between model complexity and performance. Bias refers to the error introduced by overly simplified models that fail to capture the true patterns in the data, such as when a linear model is used for inherently non-linear data. Variance, on the other hand, refers to the error caused by overly complex models that fit the training data too closely, capturing noise instead of the underlying trend. As a general rule, an increase in model complexity leads to an increase in variance and a decrease in bias: the model becomes more and more accurate but starts taking into account too many small unexplained fluctuations that cannot be generalised.
These concepts are closely related to overfitting and underfitting: high-bias models tend to underfit the data by oversimplifying it, while high-variance models tend to overfit by being overly sensitive to small fluctuations in the training set. In both cases, the final model may not be reapplied to new datasets of the same nature, hence reducing its usefulness. Optimal model performance is achieved by finding the right balance between these opposing forces.

## 3.2 Supervised Machine Learning

Supervised learning involves training models using labelled data, where inputs (X) are paired with corresponding outputs (y). The goal is to learn a function f(X) that accurately predicts y for new, unseen data. Supervised learning tasks are broadly divided into two categories:

- Classification of discrete outputs
- Regression for the prediction of continuous outputs

### 3.2.1 k-Nearest Neighbours (k-NN)

k-Nearest Neighbors (k-NN) is a simple, instance-based learning algorithm used for both classification and regression tasks. It works by storing the entire training dataset and making predictions for a given query point based on the k closest data points in the feature space, where closeness is

determined by a distance metric such as Euclidean distance, Manhattan distance, or cosine similarity.

For classification, the algorithm predicts the class label based on a majority vote of the labels of the k nearest neighbours. For regression, it predicts the target value as the average of the values of the k nearest neighbours.

While k-NN is straightforward and non-parametric, its performance is highly sensitive to the choice of k and the distance metric. A smaller k reduces bias but increases variance, while a larger k reduces variance but increases bias. This can lead to under/overfitting problems.
The algorithm can struggle with high-dimensional data due to the curse of dimensionality, as the concept of distance becomes less meaningful in such spaces. It is best suited for small datasets with relatively low dimensionality and clear separation between classes or clusters.

### 3.2.2 Decision Trees

Decision trees are another example of a simple supervised machine learning problem.

Decision trees are interpretable and flexible models used for classification and regression tasks. They operate by recursively splitting the data based on feature values to create a tree-like structure, where each internal node represents a decision rule, each branch represents an outcome of the rule, and each leaf node represents a prediction. The splits are chosen to maximize a splitting criterion such as information gain (based on entropy) given by the equation below.

$$IG(T, A) = H(T) - \sum_{v \in \text{Values}(A)} \frac{|T_v|}{|T|} H(T_v) \tag{7}$$

where:

- $T$: The dataset (parent node).
- $A$: The attribute being split.
- Values($A$): The possible values of attribute $A$.
- $T_v$: The subset of $T$ where attribute $A$ has value $v$.
- $|T_v|$: The number of instances in $T_v$.
- $|T|$: The number of instances in $T$.
- $H(T)$: The entropy of the dataset $T$, calculated as:

$$H(T) = -\sum_{i=1}^{k} p_i \log_2 p_i$$

  where $p_i$ is the proportion of instances in $T$ belonging to class $i$.

An alternative to entropy is to use Gini impurity.

Decision trees are intuitive and easy to visualize, making them suitable for understanding complex decision processes. However, they are prone to overfitting, especially when the tree is allowed

to grow too deep. Techniques like pruning, which removes branches that add little predictive power, can mitigate overfitting. Decision trees handle both numerical and categorical data and are robust to missing values, but their performance can degrade in the presence of noisy or unbalanced data.

Decision trees are often used as building blocks for more advanced ensemble methods like random forests and gradient boosting.

### 3.2.3 Ensemble Methods

Ensemble methods combine the outputs of multiple "weak" or simpler learners to create a more robust, accurate, and generalisable model. The underlying idea is to leverage the collective decision-making of multiple models to reduce error and improve prediction quality. There are three primary types of ensemble methods: bagging, boosting, and stacking. Bagging trains learners in parallel on different bootstrap samples of the data and aggregates their predictions via majority voting (for classification) or averaging (for regression). In contrast, boosting trains learners sequentially, where each learner focuses on correcting the errors made by its predecessor. Lastly, stacking combines predictions from diverse models using a second-level learner, such as linear regression, to produce a final output.

Random Forests, an extension of bagging, create an ensemble of decision trees by growing each tree on a different bootstrap sample from the training data. Additionally, at each split within a tree, only a random subset of features is considered for selection, which introduces feature-level randomness to reduce correlations among trees and enhance diversity. Being a bagged method, the random forest is then aggregated by majority voting or averaging depending on the nature of the data. Random Forests are highly robust, easy to implement, and widely used in practice due to their ability to handle noise, outliers, and high-dimensional data effectively. They are also computationally efficient and parallelisable, making them ideal for large datasets. However, compared to boosted methods, Random Forests may underperform in situations where reducing bias is critical.
Boosted Decision Trees (BDTs) are an example of a boosted ensemble method as boosted trees are trained sequentially. Each tree is built to predict the residual errors of the previous trees, thereby iteratively reducing bias. The final prediction is the sum of predictions made by all individual trees, which often results in a highly accurate model. Boosted trees typically use shallow trees (depth 4–8) with thousands of trees to reduce bias while maintaining low variance. These models require careful hyperparameter tuning, such as learning rate, number of trees, and tree depth, but when tuned correctly, they can outperform Random Forests on many datasets. Despite their computational cost, boosted trees are highly effective for structured/tabular data and remain a top choice for machine learning competitions.
The fundamental difference between Random Forests and Boosted Trees lies in their approach to bias and variance. Random Forests rely on deep, unpruned trees to minimize bias, while variance is reduced through the aggregation of predictions from independent trees. They typically use a small number of trees (e.g., 10–30), each with many nodes, and are well-suited for dense datasets with low computational requirements. Boosted Trees, on the other hand, prioritize bias reduction by sequentially learning from the residuals of previous trees. They use thousands of shallow trees (e.g., depth 4–8), making them more sensitive to noise and computationally expensive to train. In practice, the choice between Random Forests and Boosted Trees often depends on the dataset and problem at hand: Random Forests excel in scenarios requiring quick, robust solutions, while Boosted Trees achieve state-of-the-art performance when carefully tuned.

### 3.2.4 Linear and Logistic Regression

Linear regression is one of the simplest and most widely used models for regression tasks, where the goal is to predict a continuous target variable.
Having already been discussed in this summary, it will not be explicited in further detail here. Refer to paragraphs 2.1 for further detail.

Logistic regression is a statistical model used for binary classification tasks. Instead of predicting a continuous target, it models the probability of an instance belonging to a particular class using the logistic function (sigmoid curve).
The model estimates the relationship between the features ($X$) and the log-odds of the target variable ($y$) being 1, expressed as:

$$\log\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n,$$

where $P$ is the probability of $y = 1$. Predictions are made by applying a threshold to the probability output (e.g., $P > 0.5$ predicts class 1).

### 3.2.5 Model Selection and Assessment

With all of these models existing for supervised learning alone, the important question of how to choose a model arises.

- **k-Nearest Neighbors (k-NN)** works best for small datasets with low dimensionality and clear class boundaries, especially when interpretability is not a priority and the decision boundary is non-linear.
- **Decision Trees** are ideal for interpretable models and datasets with mixed data types (categorical and numerical), handling non-linear relationships well while requiring minimal preprocessing.
- **Random Forests**: Suitable for robust, quick-to-train models that handle noisy, high-dimensional datasets effectively with minimal tuning, providing strong baseline performance.
- **Boosted Trees** enable state-of-the-art performance on structured/tabular data, especially when complex patterns exist, and computational resources are available for careful hyperparameter tuning. A lot of overlap exists between the use cases of random forests, decision trees and boosted DTs. However, boosted DTs work well on clean structured data and struggle compared to random forests when it comes to large high dimensional data. They are also much more difficult to interpret than standard decision trees making them less suited for very simple applications. In all other cases, they will outperform random forests and decision trees.
- **Linear Regression** is decent at predicting continuous outcomes when the relationship between features and the target is approximately linear, and the dataset satisfies the assumptions listed in 2.1.
- **Logistic Regression** is very effective for binary classification problems with linearly separable data.

Once one of these models has been chosen for a given task, their hyperparameters must be tuned. A hyperparameter is a (or the) "governing" parameter of the model such as the depth of a decision

tree, the number of trees in a random forest or the number of nearest neighbours.
The model must be run on the test dataset with large number of different hyperparameters. The hyperparameter(s) returning the lowest validation error will thus define the optimal machine learning model for the dataset at hand.

## 3.3 Unsupervised Machine Learning

Unsupervised learning involves discovering patterns and structures in data without labelled outputs. The goal is to compute a function f(X) that represents the data in a simplified or meaningful way. Unsupervised machine learning can, similarly to supervised machine learning, be divided into two categories

- Datasets where the outputs are discrete will use clustering models
- The presence of continuous outputs will require employing an arsenal of dimensionality reduction techniques.

The paragraphs below focus on clustering techniques. As dimensionality reduction is most commonly associated with handling text data, it will be discussed in paragraphs 4.1 in the final section of this summary.

### 3.3.1 Basics of Clustering Models

Clustering is widely used in applications such as data exploration, data compression, and feature discretisation. It supports supervised learning by generating cluster-based labels and is used in collaborative filtering for tasks like identifying movie genres based on user preferences. While clustering can provide valuable insights, it is essential to avoid "cluster bias," where humans overinterpret patterns as discrete classes even when the data is continuous. In such cases, continuous models like dimensionality reduction or soft clustering may be more appropriate.
There exist different types of clustering methods out there. Hierarchical clustering builds a nested tree-like structure (dendrogram) to represent data groupings at different levels, while flat clustering, like k-Means, partitions data directly into a predefined number of clusters. Hard clustering assigns each point to a single cluster, whereas soft clustering allows points to belong to multiple clusters with probabilities, providing flexibility for overlapping groups.

Clustering in low-dimensional, small datasets is often intuitive, but real-world applications frequently involve high-dimensional spaces, where clusters become less distinct and data sparsity increases. For example, in collaborative filtering for movies or documents, the data space can have millions of dimensions. Choosing the right distance metric and clustering method is critical in such scenarios. Common metrics include but are not limited to the euclidean distance, cosine similarity or the Jaccard index frequently used for set based data.

### 3.3.2 Examples of Clustering Machine Learning Models

The most common types of clustering machine learning models are described below.

**Hierarchical Clustering** is a versatile method that organizes data into a hierarchical tree-like structure, known as a dendrogram, where each level represents a different grouping of the data. There are two main approaches: agglomerative and divisive. Agglomerative clustering, a bottom-up approach, starts with each data point as its own cluster and iteratively merges the closest clusters

based on a distance metric. Divisive clustering, a top-down approach, begins with all data points in one cluster and recursively splits it. Clusters can be represented using centroids, which are averages of points for Euclidean data, or clustroids, which are actual data points that best represent their cluster. Hierarchical clustering provides interpretability through the dendrogram but becomes computationally expensive for large datasets. Additionally, the choice of the linkage method (e.g., single, complete, or average) significantly impacts the clustering outcome.

**k-Means Clustering** is one of the most widely used clustering algorithms due to its simplicity and efficiency. The algorithm partitions the data into k clusters by iteratively minimizing the sum of squared distances from points to their respective cluster centroids. It begins with an initialization phase where k initial centroids are chosen, often using methods like k-Means++ to improve stability. Each point is then assigned to the nearest centroid, and the centroids are updated as the mean of the points assigned to them. This process repeats until the assignments no longer change or a maximum number of iterations is reached. k-Means works well for compact, spherical clusters but struggles with non-convex clusters, outliers, and noisy data. Moreover, the number of clusters, k, must be specified beforehand, which can be challenging for datasets with unknown structures.

**Density-Based Spatial Clustering of Applications with Noise** (more commonly known as DBSCAN) is a powerful clustering method designed to identify clusters of varying shapes and handle noise effectively. Unlike methods like k-Means, DBSCAN does not require predefining the number of clusters. Instead, it relies on two parameters: the radius of the neighbourhood ($\epsilon$) and the minimum number of points (minPts) needed to form a dense region. Data points are classified as core points, border points, or noise based on their density. Clusters are formed by connecting density-reachable points, where one point is reachable from another if it lies within its $\epsilon$-radius. DBSCAN excels in identifying clusters with irregular shapes and is robust to noise, but its performance depends on the choice of $\epsilon$ and minPts. It can also struggle in high-dimensional spaces due to the curse of dimensionality, where defining density becomes less meaningful.

Each clustering method has its strengths and weaknesses, making them suitable for different types of data and applications. Hierarchical clustering is ideal for understanding the nested structure of data, k-Means is efficient for partitioning compact clusters, and DBSCAN is highly effective for noisy data and clusters of arbitrary shapes. The choice of method depends on the dataset's characteristics and the specific goals of the analysis.

# 4 Analysis of Different Data Types

*This course covers weeks 10 to 13 of the course regarding the analysis of differing types of data. Section 3.1 corresponds to weeks 10 and 11 while sections 3.2 and 3.3 cover weeks 12 and 13 respectively.*
*Do add the libraries at some point.*

## 4.1 Analysis of Textual Data

Textual data is one of the most prominent forms of unstructured data in modern datasets. It spans a variety of sources, including web pages, social media platforms, and news articles. Unlike structured numerical data, text requires substantial preprocessing to be effectively utilized for machine learning (ML) tasks.
Four primary steps form the text data pipeline for ML purposes: document retrieval, classification, sentiment analysis, and topic detection. These tasks demand the transformation of text into a numerical format compatible with ML models, emphasizing the critical role of preprocessing in the pipeline.

### 4.1.1 Manipulation of Textual Data

The manipulation of text can take many different forms that are described below.

**Character encoding** is the foundational step in text processing. It translates characters into machine-readable formats, ensuring data consistency across platforms. UTF-8 Unicode serves as the standard encoding format in modern computers. Proper encoding is vital to prevent misinterpretations of text, particularly in multilingual datasets. For instance, mismatched encoding can render characters unreadable, severely hampering subsequent processing steps.

**Language detection** is essential when working with multilingual content. It determines the language of a given text sample, enabling the application of language-specific preprocessing techniques. Common methods use letter trigrams—three-character combinations indicative of linguistic patterns.

**Tokenization** splits text into smaller units, typically words or subwords, which serve as the basic building blocks for further analysis. While straightforward in languages like English, which have clear word boundaries, tokenization becomes challenging in languages such as Chinese, where words are not delimited by spaces. Additionally, compound words in languages like German or Swedish require nuanced approaches to identify meaningful splits. Standard tokenization tools, including spaCy and NLTK, are optimized for English but may require customization for other languages. Poor tokenization can introduce noise, affecting the quality of downstream representations.

**Stopword removal** eliminates common, high-frequency words that contribute little to the overall meaning of the text. Examples include articles ("the," "a"), conjunctions ("and," "or"), and auxiliary verbs ("is," "are"). While this step reduces noise and computational complexity, excessive or inappropriate stopword removal can harm certain applications, such as sentiment analysis or psychological profiling. It is crucial to tailor stopword lists to specific tasks and datasets to avoid losing valuable contextual information.
Stop word removal represents a problem similar to that of handling **social media text data**. Social media text introduces unique challenges due to informal language, abbreviations, misspellings, and

repeated characters. For instance, "ikr smh" represents "I know, right? Shake my head," while "loooove" emphasizes "love" with repeated characters. Traditional NLP tools struggle with such cases, necessitating specialized libraries like TweetNLP to effectively process and normalize social media text.

**Word normalization** forms a final important step in text data handling. It transforms words into a consistent format to reduce sparsity in the feature matrix. This process has two key components:

1. Case Folding converts all text to lowercase, reducing variation caused by capitalization. For example, "Apple" and "apple" would be treated as the same feature. However, case folding risks conflating semantically distinct terms, such as "Apple" (the company) and "apple" (the fruit).

2. Stemming and Lemmatization map word variants to a common root or lemma. Stemming, exemplified by the Porter Stemmer, applies heuristic rules to strip affixes, while lemmatization uses lexicons to produce accurate base forms. These methods are particularly useful for morphologically rich languages like German or Finnish, though lemmatization requires more resources and is harder to implement.

### 4.1.2 Feature Representation

Once the text data has been through its initial preprocessing stage described in the previous paragraph, it can ow be represented as features for machine learning applications.
Representing text data as features can take different forms. The most widely employed ones are described below.

**The Bag of Words** (BoW) model represents text as an unordered collection of word frequencies, disregarding grammar, word order, and syntax. This simple approach transforms text into numerical feature vectors compatible with ML algorithms. However, BoW generates high-dimensional and sparse matrices, as each unique word in the vocabulary corresponds to a dimension. For instance, the matrix for a dataset like Wikipedia, containing millions of documents and words, can have trillions of entries. Sparse matrix formats mitigate storage issues by focusing on non-zero values, but preprocessing steps like encoding, normalization, and stopword removal are crucial to ensure meaningful input.
**Term Frequency-Inverse Document Frequency** (TF-IDF) builds on the BoW model by weighting words according to their importance. It emphasizes less frequent but more informative words while downweighting common terms. The formula combines:

- Term Frequency (TF) of a word in a document.
- Inverse Document Frequency (IDF) which is a logarithmic measure of how rarely a word appears across the corpus.

The resulting weighted matrix provides a more nuanced representation, improving the ability of ML models to identify key features in text data.
The **n-grams** approach extends BoW by capturing sequences of n consecutive words. For example, bigrams (n=2) and trigrams (n=3) add context by preserving local word order, enabling models to identify phrases like "United States" or "operating system." However, this method can lead to a combinatorial explosion in dimensionality, necessitating feature selection techniques to retain meaningful n-grams without overwhelming the system.

**Latent Dirichlet Allocation (LDA)** and **Latent Semantic Analysis (LSA)** using **Singular Value Decomposition (SVD)** are both dimensionality reduction techniques, but they differ in approach and output. LDA is a probabilistic method that models documents as mixtures of latent topics, where each topic is represented as a probability distribution over words. It produces interpretable results by assigning probabilities to topics for each document, making it effective for tasks requiring explicit topic identification. In contrast, LSA leverages linear algebra to decompose the term-document matrix into a lower-dimensional continuous space, capturing latent features or "topics" based on variance in the data. While LDA emphasizes semantic coherence and interpretable topic distributions, LSA focuses on mathematical efficiency and continuous latent space representations, often resulting in less interpretable dimensions. Both techniques address sparsity and high dimensionality but cater to different needs in text analysis.
These techniques represent examples of continuous features in unsupervised machine learning.

Finally, **word embeddings** represent words as dense vectors in continuous space. These embeddings capture semantic relationships based on context, addressing the sparsity of traditional models. For instance, words like "king" and "queen" might have similar vectors due to shared contextual patterns. Advanced embeddings, like BERT, incorporate surrounding text to produce contextualized vectors, resolving ambiguities such as polysemy (e.g., "bat" as an animal versus a sports tool).

## 4.2 Analysis of Graph Data

### 4.2.1 Basics of Network Data

Network data represents relationships or interactions between entities, modelled as graphs where nodes represent entities and edges represent connections between them. Networks appear in a variety of domains, including social networks, biological systems, transportation, and communication networks. Unlike traditional datasets, network data captures both node-level attributes and the structure of relationships, making it highly expressive but also complex to analyse.

Networks are composed of entities (known as nodes) that are linked together. These links may be either directed or undirected. Graphs as a whole can be weighed and bipartite graphs, where every single node is present on one of two parallel lines of nodes can also be analysed.
Every node has a degree that indicates to how many other nodes it is linked. In the case of directed graphs, both the in and out-degrees are defined separately.

### 4.2.2 Numerical Representation of Graphs

It would be trivial to explain why computers do not store graphs as we perceive them.

**Adjacency matrices** are one of the most common ways to represent graphs on computers ($A_{ij}$ = 1 if nodes i and j are linked). In the case of directional graphs, $A_{ij} = 1$ if there is a link from i to j. The matrix would then obviously not be symmetric.

While simple, adjacency matrices are not particularly efficient as they tend to be very sparse and include a lot of junk data. As such, more efficient ways of representing graphs on computers are frequently used.
The use of edge lists (i, j) is an example of a more efficient approach to adjacency matrices (i would

be the source and j the target of the edge if the graph is directional). Edge lists enable drawing up adjacency lists that simply list what nodes are connected by edges originating at a given node.

### 4.2.3 Properties of Real Networks

Real-world networks exhibit unique characteristics that distinguish them from arbitrary graphs. Despite their diversity—spanning social, biological, and informational systems—they share common properties.
These are listed below.

- Networks tend to be **sparse** as nodes are connected to only a small fraction of nodes. In the case of social networks, node degree is limited by practical limits.
- Unlike uniform or random distributions, real-world networks show a **skewed degree distribution** where a small number of nodes (hubs) have a disproportionately high degree. This contributes to the robust and scalable nature of these networks, as hubs facilitate efficient information and resource flow.
- Networks demonstrate a tendency for nodes connected to a common neighbour to also connect with each other in a process called **triadic closure**. This is quantified by the clustering coefficient $C_i$, which measures the density of triangles around a node. High clustering coefficients are typical in social and biological networks.
- Real-world networks cluster into tightly-knit **community structures**, connected via weaker ties. Granovetter's theory on "The Strength of Weak Ties" emphasizes the importance of these weak connections in bridging communities and filling "structural holes" in networks. Importantly, communities in networks are often overlapping rather than disjoint.
- Despite their sparsity, real-world networks have remarkably **short paths between nodes**. For example, in social networks, the "six degrees of separation" phenomenon highlights this. However, the critical insight lies in the navigability of these short paths as shown in the Erdős-Rényi model.
- Networks show tendencies for similar nodes to connect (**homophily**), as seen in shared behaviours among friends, or for dissimilar nodes to connect (**heterophily**) in diverse collaborations. Homophily often complicates studies, as it blurs the line between influence (adoption of behaviour due to connections) and inherent similarity.

### 4.2.4 Node Importance Measurements

The importance of a node in a network can be quantified using *centrality measures*, which assign a scalar value $C(i)$ to each node $i$ based on its role and position in the network.
Different measures of centrality provide varying perspectives on what constitutes an important node. The most widely used ones are described below.

**Degree centrality** is the simplest measure of importance. It calculates the number of direct connections (neighbours) a node has:

$$C(i) = \text{degree}(i)$$

Nodes with higher degrees are considered more important.

**Closeness centrality** assesses how quickly a node can reach other nodes in the network. It is defined as the reciprocal of the sum of distances from the node to all other nodes:

$$C(i) = \frac{1}{\sum_j d(i,j)}$$

Here, $d(i,j)$ is the shortest-path distance between nodes $i$ and $j$. Nodes with higher closeness centrality can efficiently spread information. A variant, *harmonic centrality*, sums the reciprocals of distances, making it applicable to disconnected graphs.

**Betweenness centrality** measures the extent to which a node lies on the shortest paths between other nodes. It is calculated as:

$$C(i) = \sum_{s \neq i \neq t} \frac{\sigma_{st}(i)}{\sigma_{st}}$$

where $\sigma_{st}$ is the total number of shortest paths between nodes $s$ and $t$, and $\sigma_{st}(i)$ is the number of those paths passing through node $i$. Nodes with high betweenness centrality serve as bridges, connecting different parts of the network. This measure is computationally expensive, as it requires analysing all pairs of nodes.

**Katz centrality** generalises degree centrality by accounting for both direct and indirect neighbours. It assigns less weight to connections farther from the node. The centrality is computed as:

$$C(i) = \sum_{k=1}^{\infty} \alpha^k \left(A^k\right)_{ij}$$

where $A$ is the adjacency matrix, $\alpha$ is a damping factor, and $A^k$ represents the number of paths of length $k$. Katz centrality is more robust than degree centrality, as it incorporates the global structure of the network.

**PageRank centrality**, popularized by Google's search algorithm, uses a recursive definition:

$$C(i) = \sum_{j \in \text{in-neighbours of } i} \frac{C(j)}{\text{degree}(j)}$$

This approach considers the importance of incoming connections, emphasizing nodes linked by other significant nodes. In matrix notation, it solves the equation:

$$\mathbf{x} = M\mathbf{x}$$

where $\mathbf{x}$ is the centrality vector, and $M$ is derived from the adjacency matrix $A$. PageRank centrality corresponds to the principal eigenvector of $M$ and can be interpreted as the steady-state distribution of a random walker on the network.

## 4.3 Analysis of Massive Data

IN the real world, data can easily become very large - from the 100's of GB to the PB.. Traditional methods that assume data can fit into the memory of one machine, such as using Pandas, are insufficient for handling real-world datasets generated by sources like social media, sensors, or financial markets, which can range from gigabytes to petabytes. The rapid growth of data compared to the

limitations of hardware, such as disk read speeds, CPU performance, and memory, necessitates new approaches to data processing. A key solution is to distribute data across multiple machines in a cluster, though this brings its own challenges, including task distribution, fault tolerance, and the handling of slow or failing nodes.

Distributed computation models help address these challenges. One foundational model is MapReduce, which splits tasks into mapping and reducing operations, allowing parallel data processing while ensuring fault tolerance and task monitoring. For example, counting word frequencies in a large document can be efficiently accomplished by distributing parts of the document to multiple machines using this model. Another prominent framework introduced is Spark, which provides a more flexible abstraction for distributed data processing through Resilient Distributed Datasets (RDDs). RDDs support two types of operations: transformations, which create new RDDs and are executed lazily, and actions, which trigger actual computation and return results. Spark optimizes performance by delaying execution until necessary, allowing for more efficient query planning.

Spark introduces additional features such as broadcast variables, which minimize data transfer by efficiently sharing read-only data across all nodes, and accumulators, which aggregate information from parallel tasks. RDD persistence allows intermediate results to be cached, improving the performance of iterative algorithms by avoiding recomputation. Beyond RDDs, Spark offers higher-level abstractions like DataFrames, which resemble tables with rows and columns, and Spark SQL, which enables querying structured data using SQL syntax. These abstractions simplify data analysis and are often more efficient than direct RDD manipulation due to built-in optimizations.
Spark supports its own built in machine learning library, MLlib, which provides scalable algorithms for classification, regression, and unsupervised learning. It supports optimization techniques such as stochastic gradient descent (SGD) and enables large-scale machine learning tasks to be performed efficiently. Ultimately, the lecture highlights the importance of distributed computing frameworks like Spark for handling datasets that exceed a few terabytes. While traditional tools are suitable for smaller data, scalable systems become essential for larger datasets. Spark provides a powerful platform for distributed data processing and machine learning, though it requires practice and deeper understanding to use effectively.