Welcome to the CoGrammar

Lecture: Theory Summary I

The session will start shortly...

Questions? Drop them in the chat. We'll have dedicated moderators answering questions.

Data Science Session Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: **Questions**

Data Science Session Housekeeping cont.

- For all **non-academic questions**, please submit a query: **www.hyperiondev.com/support**
- Report a **safeguarding** incident: **www.hyperiondev.com/safeguardreporting**
- We would love your **feedback** on lectures: **Feedback on Lectures**

Skills Bootcamp
8-Week Progression Overview

Fulfil 4 Criteria to Graduation

✅ Criterion 1: Initial Requirements

Timeframe: First 2 Weeks
Guided Learning Hours (GLH): Minimum of 15 hours
Task Completion: First four tasks

Due Date: 24 March 2024

✅ Criterion 2: Mid-Course Progress

60 Guided Learning Hours

Data Science - **13 tasks**
Software Engineering - **13 tasks**
Web Development - **13 tasks**

Due Date: 28 April 2024

CoGrammar

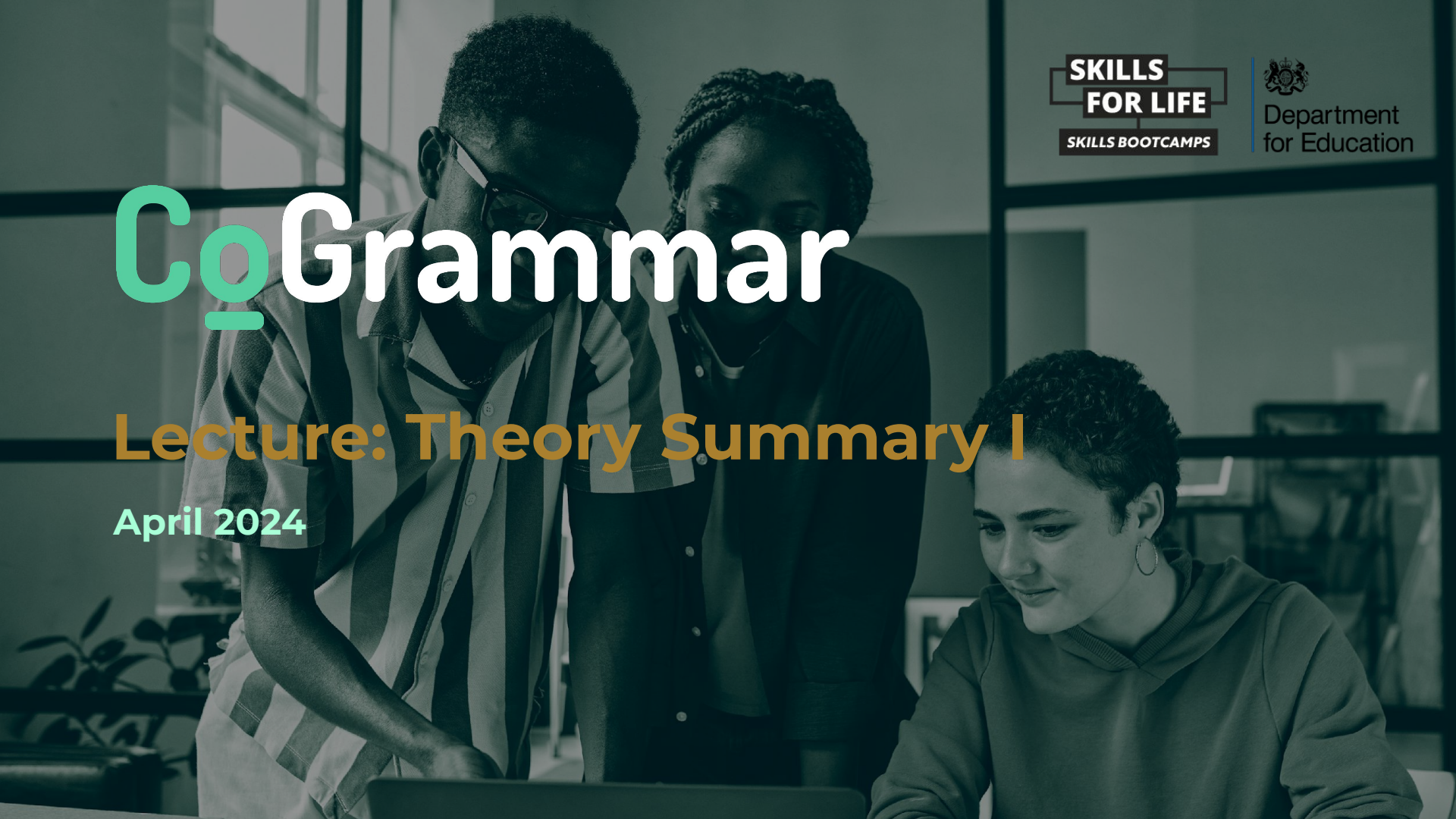
Skills Bootcamp Progression Overview

✅ Criterion 3: Course Progress

Completion: All mandatory tasks, including Build Your Brand and resubmissions by study period end
Interview Invitation: Within 4 weeks post-course
Guided Learning Hours: Minimum of 112 hours by support end date (10.5 hours average, each week)

✅ Criterion 4: Demonstrating Employability

Final Job or Apprenticeship Outcome: Document within 12 weeks post-graduation
Relevance: Progression to employment or related opportunity



SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar

Lecture: Theory Summary I

April 2024

Learning objectives

- Understand the essential Python libraries for data science
- Develop a comprehensive understanding of data visualisation techniques and tools
- Consolidate knowledge of the various steps in data exploration and preparation
- Understand fundamental concepts of the learning algorithms covered

Python packages for data science

NumPy & Pandas

CoGrammar

Python packages for data science: NumPy

- **NumPy** - stands for **Num**erical **Py**thon
- Numpy arrays differ from python lists
  - Elements of same datatype
  - Can handle arithmetic operations
  - Preferred for larger chunks of data
  - Requires proper modules to perform operations on them
- Array oriented programming, NumPy has smaller memory consumption, better runtime, and ease of data manipulation

```
pip install numpy
```

```
import numpy as np

#Define a list
distances = [1, 13.1, 26.2, 100]
print(type(distances))
print(distances)
#Output: <class 'list'>
#Output: [1, 13.1, 26.2, 100]

#Convert to numpy array
numpy_dist = np.array(distances)
print(type(numpy_dist))
print(numpy_dist)
#Output: <class 'numpy.ndarray'>
#Output: [  1.   13.1  26.2 100. ]
```

CoGrammar

Python packages for data science: Pandas

- **Pandas**, built on NumPy, is a Python module that contains high-level data structures and tools designed for fast and easy data analysis.

```
pip install pandas
```

```
import pandas as pd
```

- Fundamental pandas data structures
  - Series (1-dimensional labelled array, can hold any data type)
  - DataFrame (2-dimensional)
  - Panel (pandas -> **pan**el **da**ta)

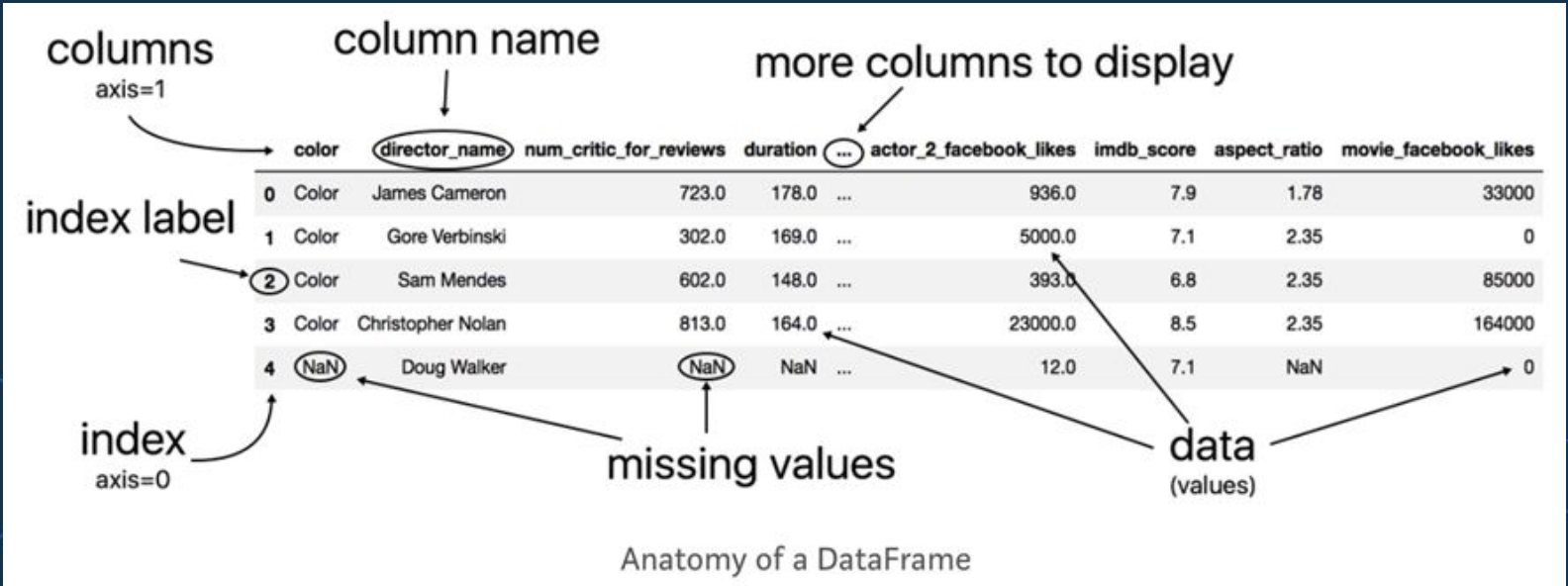
	Name	Age	Marks
0	Asha	12	96
1	Ben	12	92
2	Candice	13	94
3	Derek	12	96
4	Miriam	12	95
5	Seth	13	93
6	Zara	12	95

Pandas DataFrames

CoGrammar

Pandas DataFrames

- A DataFrame is the way the Pandas library in Python represents tabular data. It's like a powerful *spreadsheet* within your code.
- The pandas' library documentation defines a DataFrame as a "two-dimensional, size-mutable, with labelled rows and columns."



	color	director_name	num_critic_for_reviews	duration	...	actor_2_facebook_likes	imdb_score	aspect_ratio	movie_facebook_likes
0	Color	James Cameron	723.0	178.0	...	936.0	7.9	1.78	33000
1	Color	Gore Verbinski	302.0	169.0	...	5000.0	7.1	2.35	0
2	Color	Sam Mendes	602.0	148.0	...	393.0	6.8	2.35	85000
3	Color	Christopher Nolan	813.0	164.0	...	23000.0	8.5	2.35	164000
4	NaN	Doug Walker	NaN	NaN	...	12.0	7.1	NaN	0

Anatomy of a DataFrame

Pandas cheat sheet

IMPORTING DATA

- `pd.read_csv(filename)` - From a CSV file
- `pd.read_table(filename)` - From a delimited text file (like TSV)
- `pd.read_excel(filename)` - From an Excel file
- `pd.read_sql(query, connection_object)` - Reads from a SQL table/database
- `pd.read_json(json_string)` - Reads from a JSON formatted string, URL or file.
- `pd.read_html(url)` - Parses an html URL, string or file and extracts tables to a list of dataframes
- `pd.read_clipboard()` - Takes the contents of your clipboard and passes it to `read_table()`
- `pd.DataFrame(dict)` - From a dict, keys for columns names, values for data as lists

EXPORTING DATA

- `df.to_csv(filename)` - Writes to a CSV file
- `df.to_excel(filename)` - Writes to an Excel file
- `df.to_sql(table_name, connection_object)` - Writes to a SQL table
- `df.to_json(filename)` - Writes to a file in JSON format
- `df.to_html(filename)` - Saves as an HTML table
- `df.to_clipboard()` - Writes to the clipboard

SELECTION

- `df[col]` - Returns column with label **col** as Series
- `df[[col1, col2]]` - Returns Columns as a new DataFrame
- `s.iloc[0]` - Selection by position
- `s.loc[0]` - Selection by index
- `df.iloc[0,:]` - First row
- `df.iloc[0,0]` - First element of first column

VIEWING/INSPECTING DATA

- `df.head(n)` - First **n** rows of the DataFrame
- `df.tail(n)` - Last **n** rows of the DataFrame
- `df.shape()` - Number of rows and columns
- `df.info()` - Index, Datatype and Memory information
- `df.describe()` - Summary statistics for numerical columns

DATA CLEANING

- `df.columns = ['a','b','c']` - Renames columns
- `pd.isnull()` - Checks for null Values, Returns Boolean Array
- `df.rename(columns={'old_name': 'new_name'})` - Selective renaming

CoGrammar

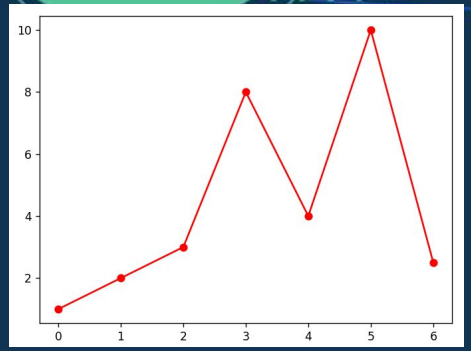
Data Visualisation

CoGrammar

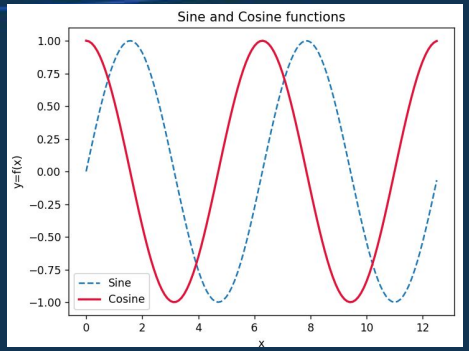
Data Visualisation

- Understanding the type of data you are working with can inform what kinds of visualisations you can create:
  - Barplots are good for plotting Categorical data vs. Continuous/Discrete data
  - Pie charts are good for plotting Categorical data vs. Discrete data. Also great for getting a sense of proportions
  - Scatterplots: Great for plotting Discrete vs. Discrete data or Continuous vs. Continuous data. Useful for finding relationships between variables.
  - Line graphs: Good for plotting Discrete / Continuous vs. Discrete / Continuous data or comparing with a Time Series.
- Matplotlib and Seaborn are two useful libraries for creating visualisations.

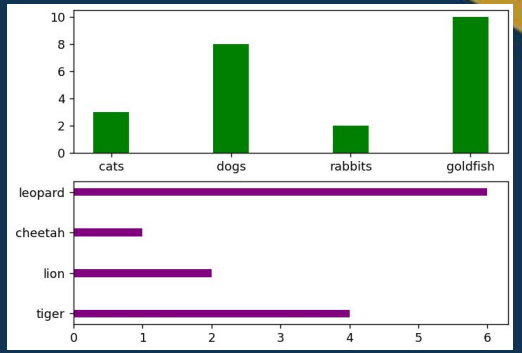
Matplotlib examples



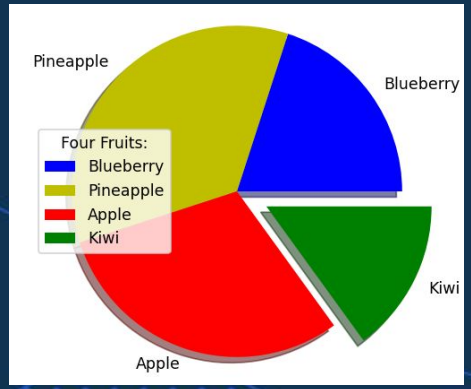
Line graph



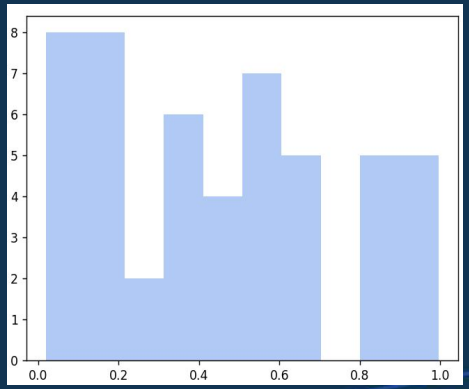
Line graph



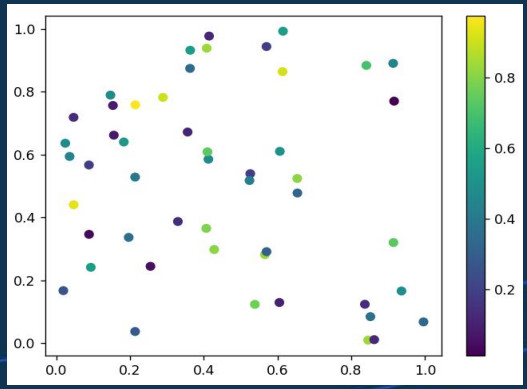
Bar plot



Pie chart



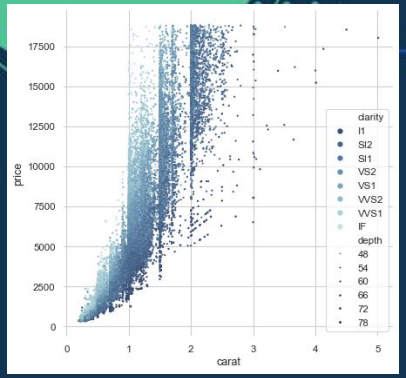
Histogram



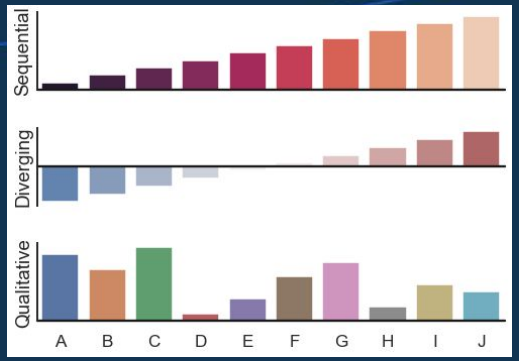
Scatterplot

CoGrammar

Seaborn examples



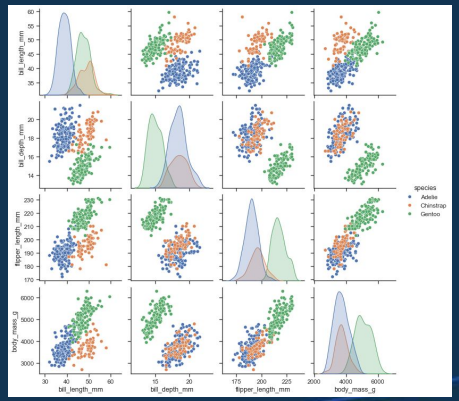
Scatterplot



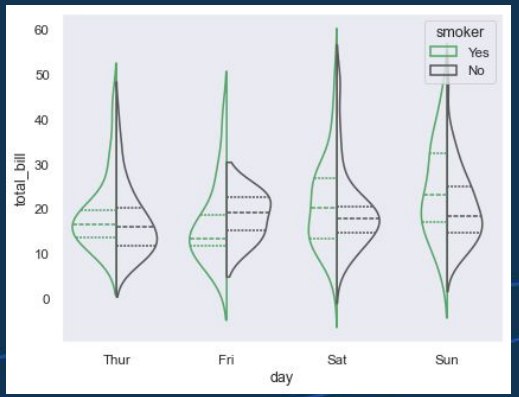
Bar plot

Line graph

Heatmap



Pairplot



Violin plot

CoGrammar

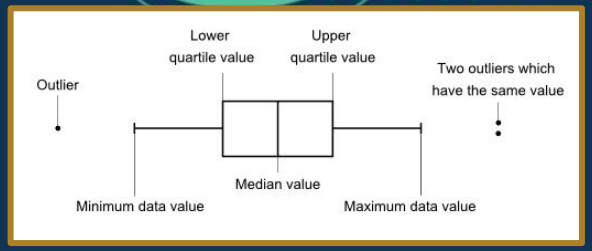
Exploratory Data Analysis

CoGrammar

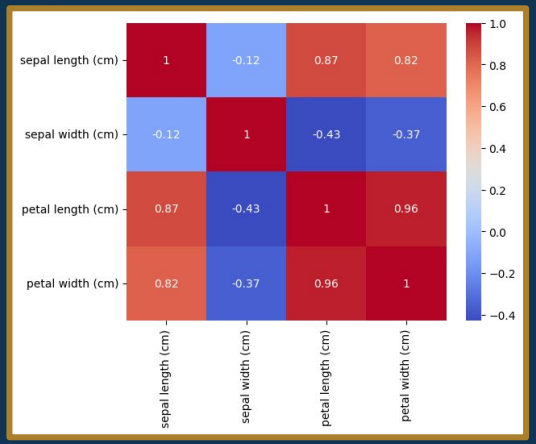
Exploratory Data Analysis

- **Definition:** Exploratory Data Analysis (EDA) is the process of investigating and understanding a dataset through visual and statistical techniques.
- Simple EDA framework:
  - Understand the data - dataset structure, check for missing values, basic statistics
  - Clean and preprocess - handle missing values, encode categorical variables, feature scaling
  - Explore relationships - analyse relationships between features and target variable, scatterplots, heatmaps, identify clusters
  - Assess feature importance - significance of features using statistical tests, Random Forests to evaluate feature importance
  - Iterate and refine - consider additional visualisations and techniques, refine data cleaning and preprocessing

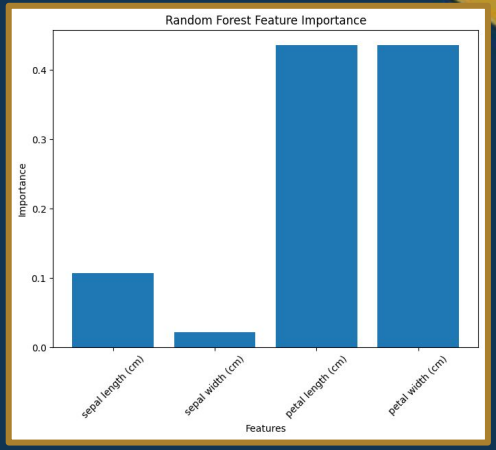
Useful visualisations for EDA



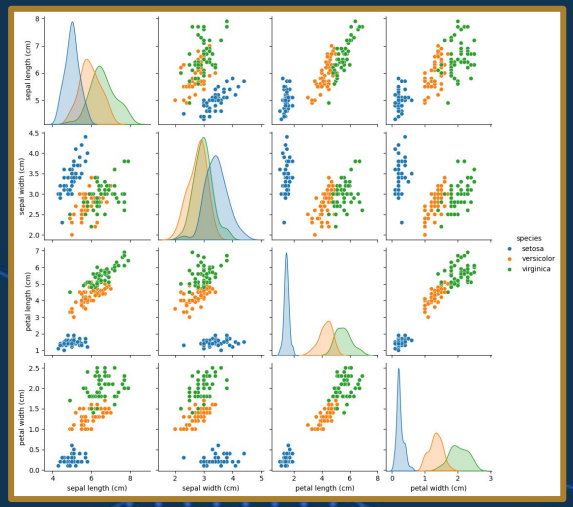
Box plot



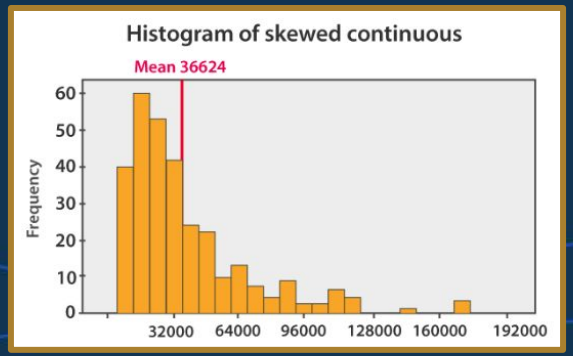
Heatmap

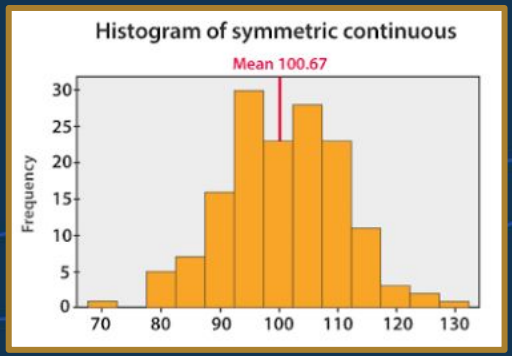


RF Feature Importance



Pairplot





CoGrammar

Feature Scaling

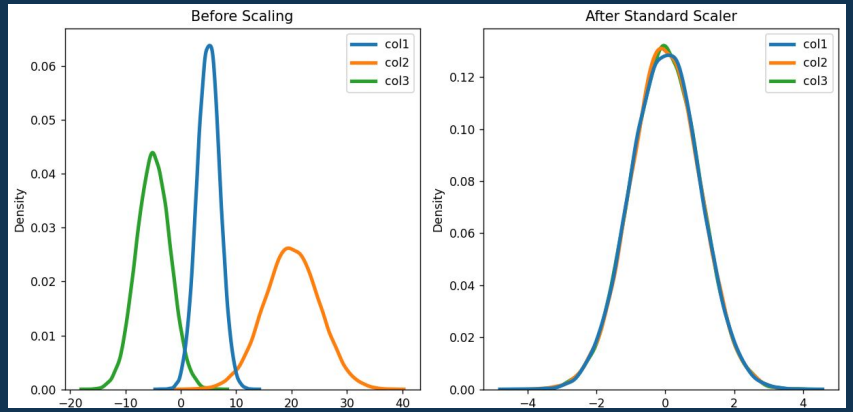
CoGrammar

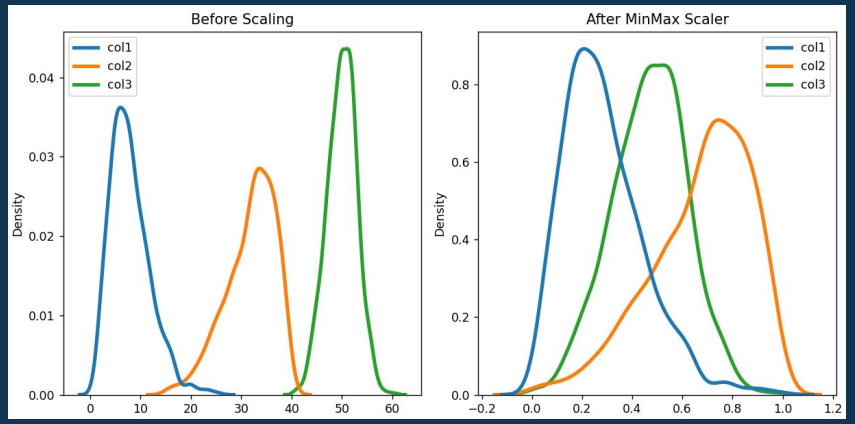
Feature Scaling

- Data columns have different values and measurement units, difficult to compare, e.g. What is kgs compared to metres? Or altitude compared to time?
- **Feature scaling - normalisation** and **standardisation**, involves transforming the data on the same scale
  - to compare easily,
  - more suitable for modeling,
  - to build accurate and effective machine learning models,
  - help improve model performance,
  - reduce the impact of outliers.

Normalisation	Standardisation
Rescales values to a range between 0 and 1	Centers data around mean and scales to standard deviation of 1
Useful when data distribution is unknown or **not Gaussian**	Useful with **Gaussian** data distribution
Sensitive to outliers	**Less sensitive to outliers**
Retains shape of original distribution	Changes shape of original distribution
May not preserve relationships between data points	Preserves relationships between data points
MinMaxScaler() (x – min)/(max – min)	**StandardScaler()** (x – mean)/standard deviation

Feature Scaling





StandardScaler
Gaussian distributions

MinMaxScaler
Non-Gaussian distribution

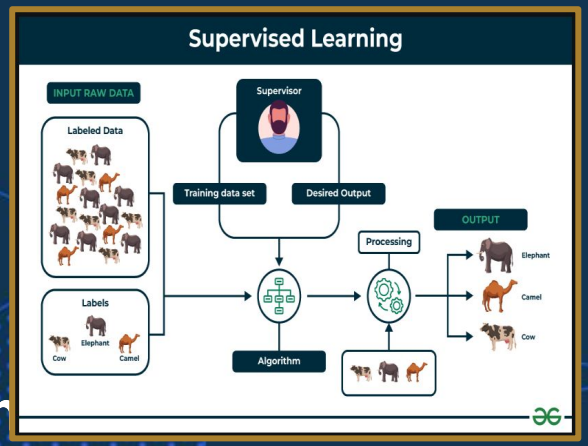
CoGrammar

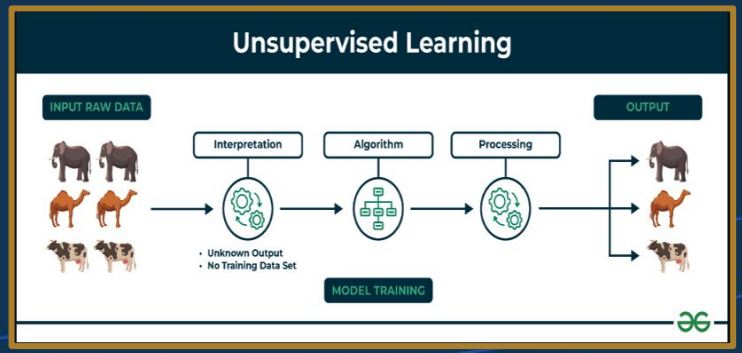
Machine Learning Overview

CoGrammar

Machine Learning Introduction

- Machine learning is a way of teaching computers to learn and improve from experience without being explicitly programmed.
- 2 types of machine learning:
  - **Supervised learning** - The computer learns from labelled data, where both input and output data are provided e.g., linear regression, logistic regression
  - **Unsupervised learning:** The computer learns from unlabeled data, discovering hidden patterns or structures on its own e.g., clustering





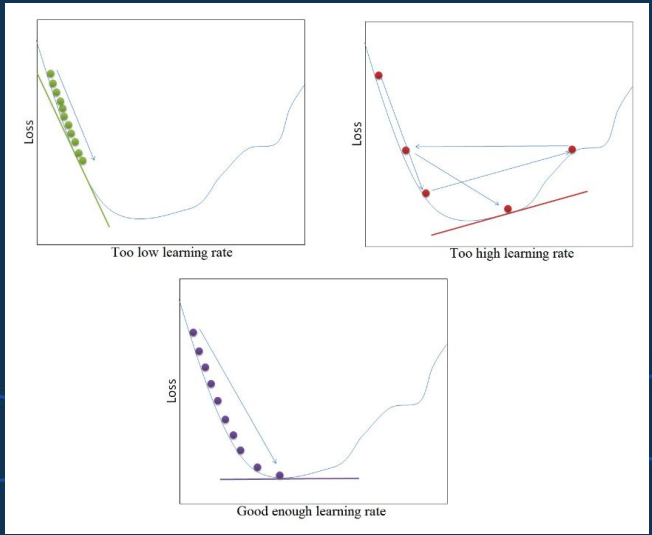
Evaluation Metrics

- It is important to evaluate the performance of your Machine Learning algorithm. The choice of evaluation metric is dependent on whether you have a **regression model** (predicting continuous numerical values, such as house prices or stock prices) or **classification model** (predicting discrete categories or classes, such as whether an email is spam or not).
- Regression model evaluation metrics:
  - **Mean Squared Error (MSE)** - measures the average squared difference between the predicted and actual values. A lower MSE indicates a better model performance.
  - **R-squared (R^2) score** - represents the proportion of variance in the target variable that can be explained by the model. An R^2 value closer to 1 indicates a better fit of the model to the data.
- Classification model evaluation metric:
  - **Accuracy** is a commonly used metric for evaluating the performance of model in classification models.

Accuracy = (Number of correct predictions) / (Total number of predictions) * 100%

Parameter Tuning

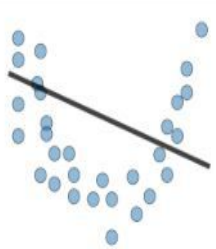
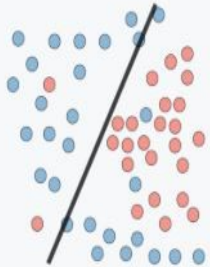
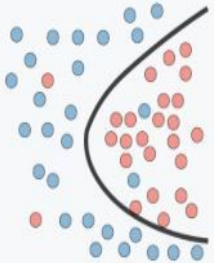
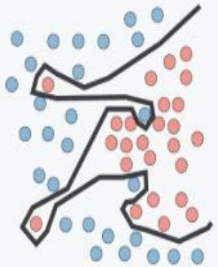
- **Parameter tuning** is the process of finding the optimal values for a model's hyperparameters to improve its performance.
- **Hyperparameters are settings** that are not learned from the data but are set before training the model.
  - Examples of hyperparameters in linear regression include the **learning rate**, **regularisation strength**, and the **number of iterations**.



CoGrammar

Bias-Variance Tradeoff

- Sometimes the models we choose can be too complex for our problem (overfitting) or too simple for our problem (underfitting). We want to find the sweet-spot or the balance between underfitting and overfitting.

	Underfitting	Just right	Overfitting
Symptoms	• High training error • Training error close to test error • High bias	• Training error slightly lower than test error	• Very low training error • Training error much lower than test error • High variance
Regression illustration			
Classification illustration			
Possible remedies	• Complexify model • Add more features • Train longer		• Perform regularization • Get more data

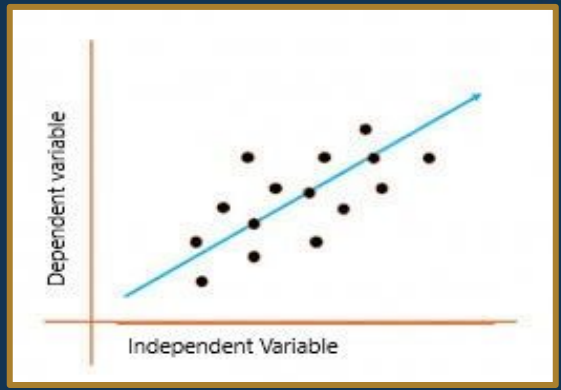
CoGrammar

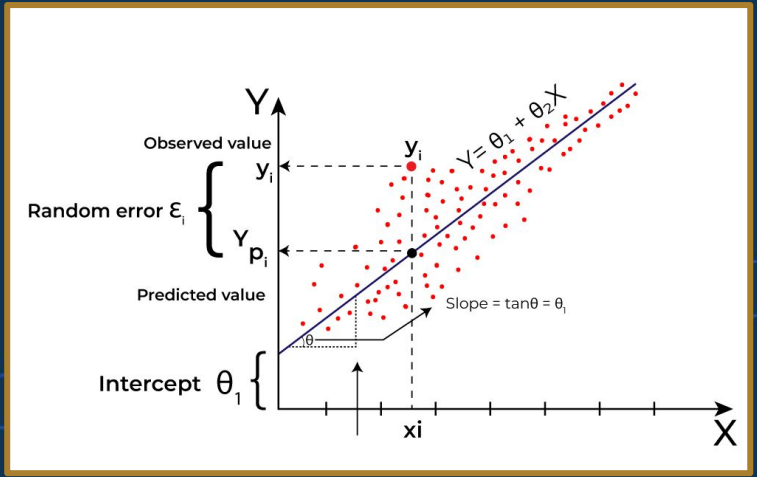
Simple & Multiple Linear Regression

CoGrammar

Simple Linear Regression

- The purpose of Simple Linear Regression is to find the best-fitting line that describes the relationship between the independent variable (x) and the dependent variable (y).
- This line can be used to make predictions about the dependent variable based on new values of the independent variable.
- The equation is written as: $y = \beta_0 + \beta_1 x + \varepsilon$
- Assumptions and limitations:
  - Linearity, independence, homoscedasticity, the errors should be normally distributed





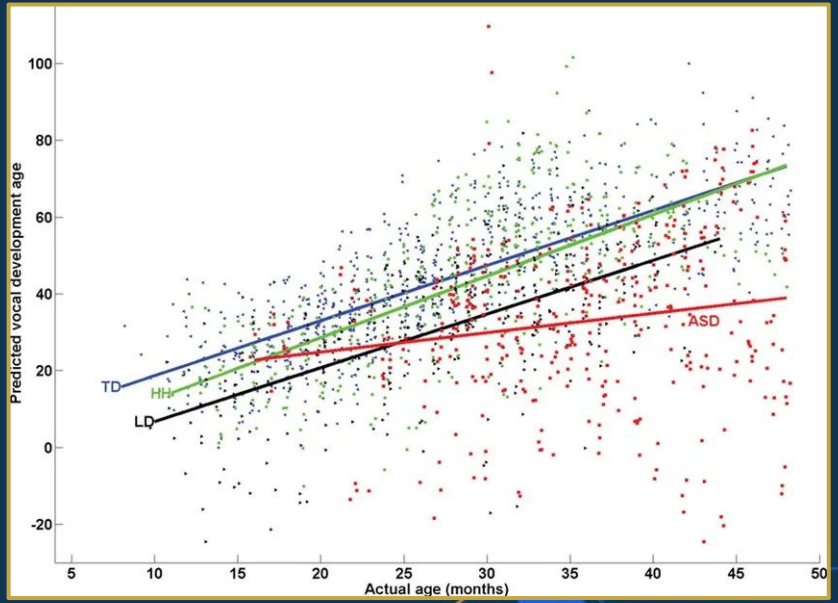
CoGrammar

Multiple Linear Regression

- Multiple Linear Regression (MLR) algorithm is used which models the linear relationship between a single dependent continuous variable and more than one independent variable.
- The equation is written as:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots + \beta_n x_n + \varepsilon$$

- Assumptions and limitations:
  - Little or no multicollinearity, homogeneity of variance, residuals must be normally distributed



CoGrammar

Logistic Regression

CoGrammar

Logistic Regression

- Logistic Regression is a supervised learning algorithm which predicts the probability of categorical variables for a given observation and assigns the observation to the category with the highest probability.
- It is a **classification algorithm** so the dependent variables must be distinct, non-continuous, categorical.
- Types of logistic regression models:
  - **Binary (Binomial) logistic regression**: Response or dependent variable has only two possible outcomes (e.g. 0 or 1, True or False, Malignant or Benign tumour, Spam or Not Spam email).
  - **Multinomial logistic regression**: Dependent variable has three or more possible outcomes; but values have no specified order (e.g., movie studios predicting film genres depending on person's age, gender, family status).
  - **Ordinal logistic regression**: Response variable has three or more possible outcome, and values have a defined order (e.g. grading scales from A to F or rating scales from 1 to 5).

Logistic function

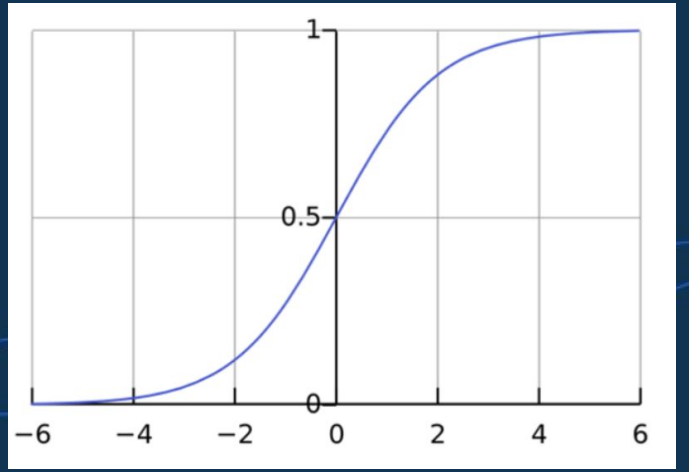
- Logistic regression: statistical model that uses the **logistic (logit) function**, as the equation between x and y (also called **sigmoid function** or **S-shaped curve**).

- Returns only values between 0 and 1 for the dependent variable, irrespective of the values of the independent variable.

- Also model equations between **multiple independent variables** and **one dependent variable.**

Sigmoid function

$$p = \frac{1}{(1 + e^{-y})}$$



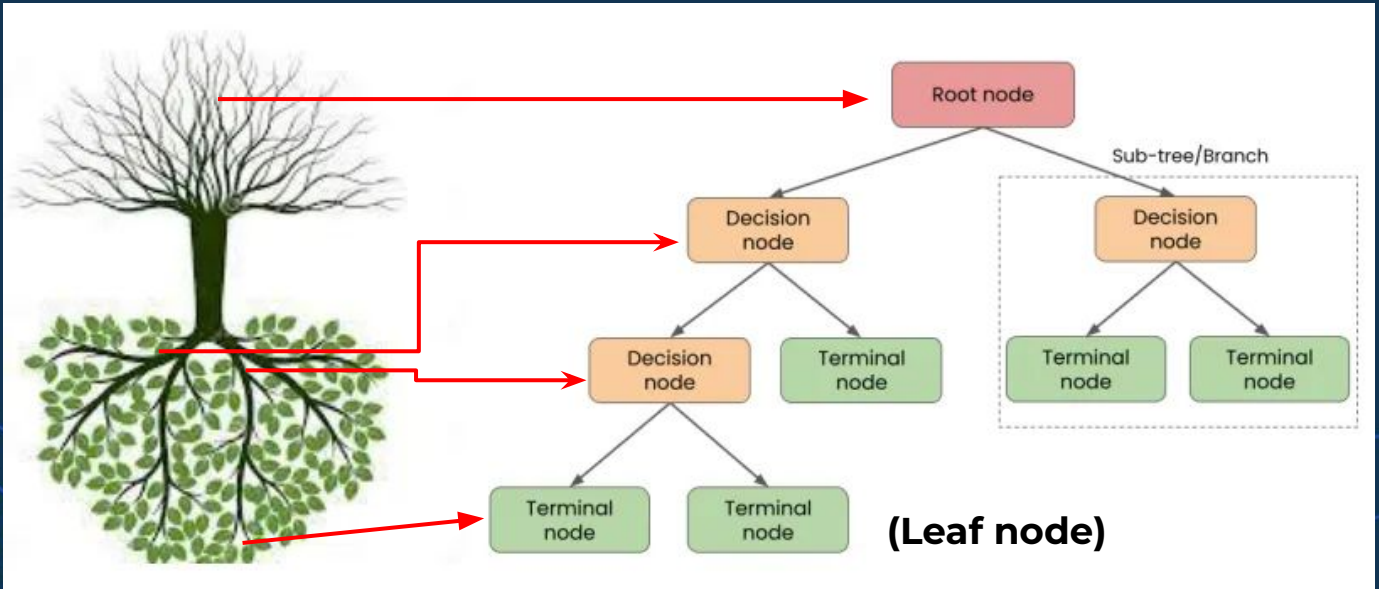
CoGrammar

Decision Trees

CoGrammar

Decision Trees

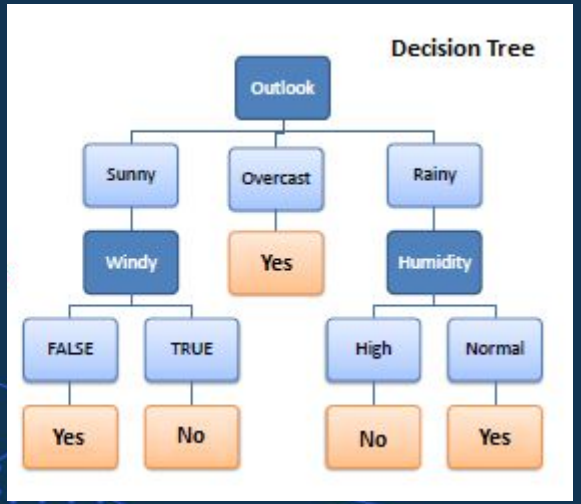
- **Supervised** learning algorithm for **regression** and **classification.**
- **Decision Trees:** (*upside down*) **tree-like** machine learning models.



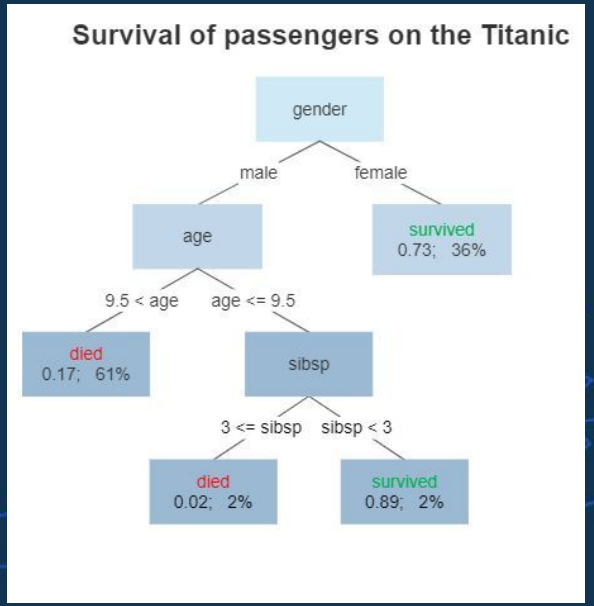
Decision Trees

- Represent data by **partitioning** it into different **smaller subsets** based on questions asked of predictive variable in the data.

Classification Tree



Regression Tree



CoGrammar

Optimising Trees: Pruning

- A **Decision tree** that is trained to its **full depth** will highly likely lead to **overfitting** the training data.
- **Pruning** avoids **overfitting**, **removes** parts of the Decision Tree that have little or **no significance** in the **decision-making process** and **prevent** it from growing to its **full depth**.
- **Pre-pruning** is done while growing the tree while **post-pruning** prunes nodes after it is built to depth.

Pre-pruning

- Tunes hyperparameters (**max_depth**, **min_samples_leaf**, **min_samples_split**) prior to the training
- 'Early stopping' - stops the growth of the **decision tree** to to reach its **full depth**
- Cross-validation error monitored at each step, if it is constant, stops growth.

Post-pruning

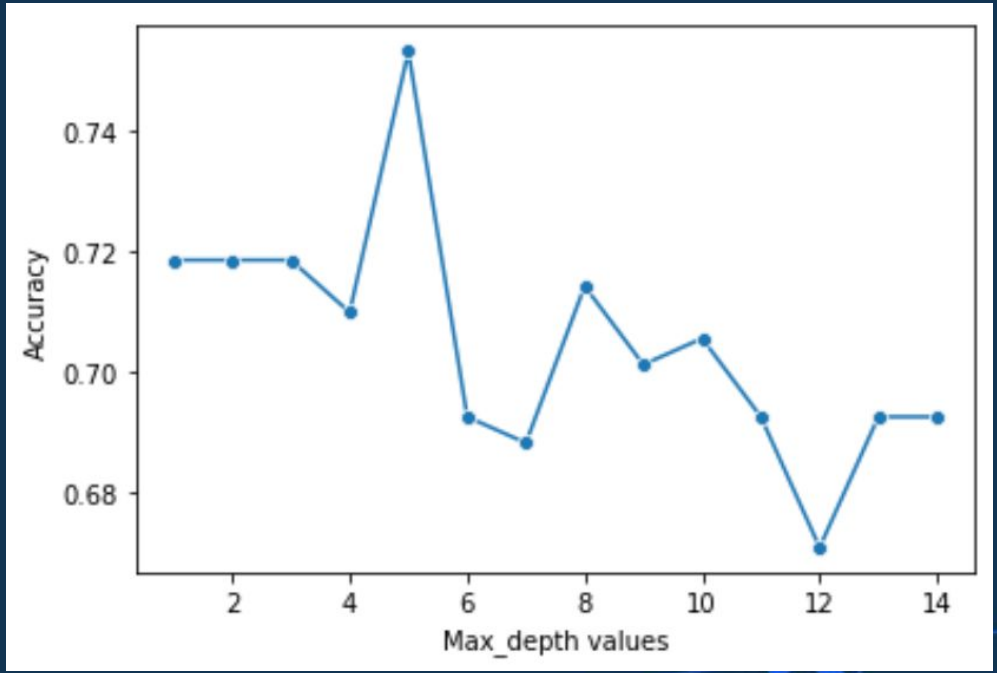
- **Decision Tree** model grows to its **full depth**, **tree branches** are **removed** to prevent the model from overfitting.
- Tune hyperparameter (**ccp_alpha - Cost Complexity Pruning**) to control the size of a tree, higher value leads to an increase in the number of nodes pruned.

Example: Pre-Pruning

We will choose **max_depth = 5** (default = None)

We will also choose **min_samples_leaf = 2,** (default=1), the minimum number of samples required to be at a leaf node.

We will also change the **splitting criterion** (the function to measure the quality of a split) to **'entropy'** instead of the default 'gini'.



How **accuracy** changes with the **max_depth** values. (The tree has a maximum depth of 14).

CoGrammar

Random Forests

CoGrammar

Ensemble Methods

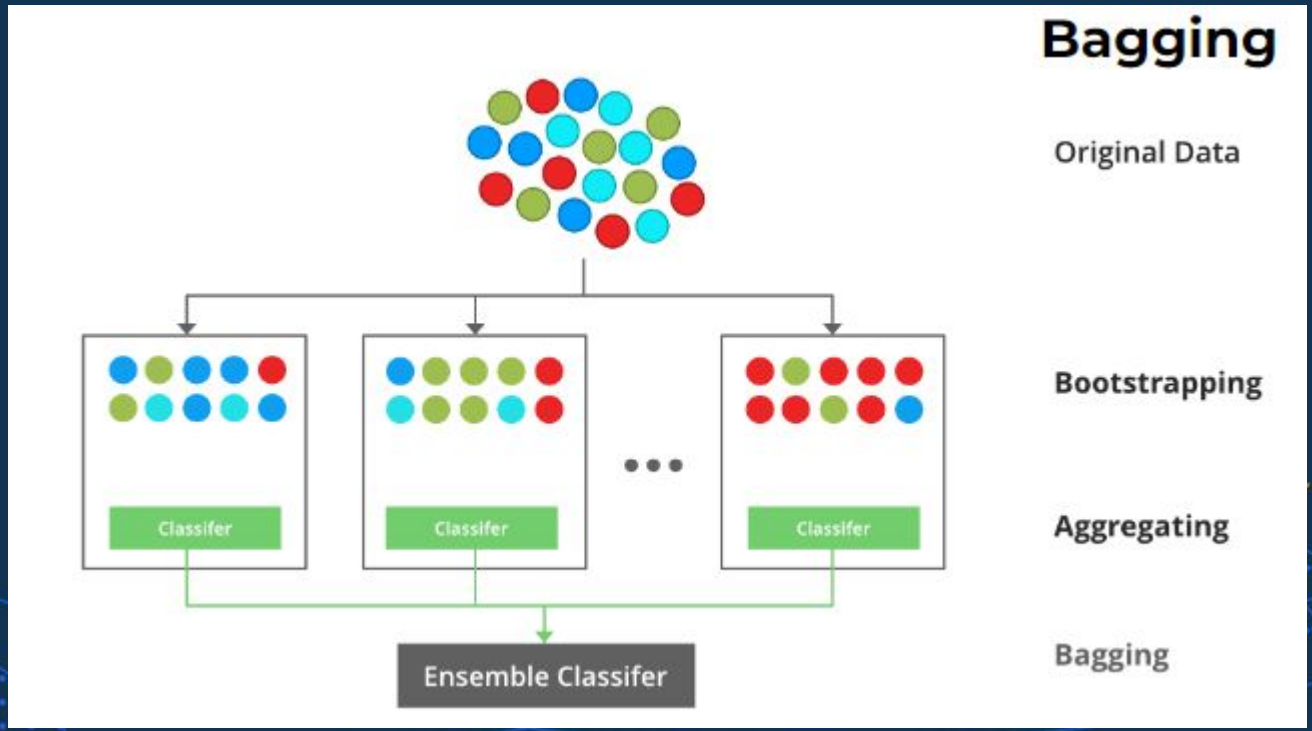
- **Decision Trees** are easy to understand, apply, interpret and visualise. However, they are **not very robust**, **small perturbations in the training data** could give rise to **substantially different predictions** at test time.

- Predictions of decision trees have very **high variance.** Ideally, we'd like our models to capture general patterns, not to be **so dependent on the data** they have trained on that a bit of noise or a different sample changes predictions entirely.

- Ensemble methods **aggregate the predictions of multiple classifiers/regressors** into a single, improved prediction.

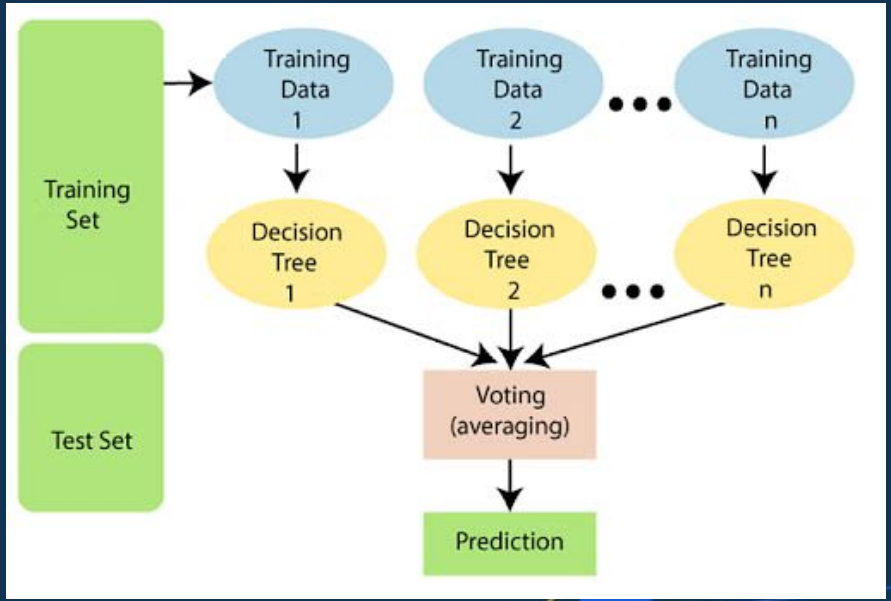
- **Ensemble techniques** work like a group of diverse experts teaming up to make decisions, and create a more robust solution than any individual could achieve alone.

Bagging: Bootstrap Aggregation



Random Forests

- **Random Forests:** created from **many decision trees** during training phase using **random subset of the dataset** to measure a **random subset of features** in each partition.
- Randomness introduces **variability** among individual trees, **reducing** the risk of **overfitting** and **improving** overall **prediction performance**.
- Aggregates predictions of all trees, either by **voting** (**classification problems**) or by **averaging** (**regression problems**).
- **Collaborative decision-making process**, supported by **multiple trees** with their insights, gives **stable** and **precise results.**



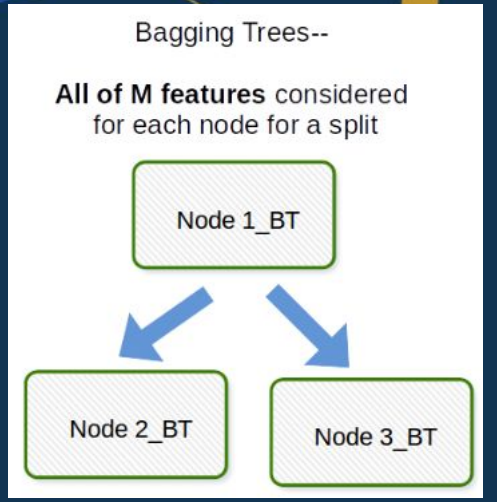
CoGrammar

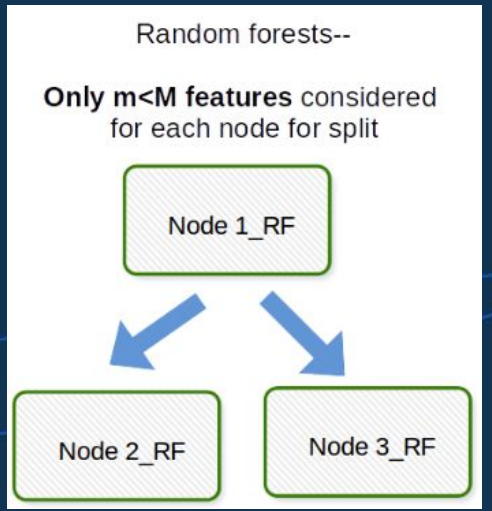
Differences with Decision Trees

Decision Trees	Random Forests
Can suffer from **overfitting** if allowed to grow without any control.	Created from subsets of data, and final output is based on average or majority ranking; **overfitting is mitigated.** Better **bias-variance trade-off.**
When a data set with features is taken as input by a decision tree, some **rules formulated to make predictions.**	**Randomly** selects observations, builds a decision tree, and takes the average result. Does not use any set of rules.
A single decision tree is **faster** in computation.	Comparatively **slower**.

Differences with Bagged Trees

Bagged Trees	Random Forests
All features are selected.	**Randomly selected features.**
Highly correlated trees, can **reduce diversity** of model. More prone to **overfitting**.	**Randomness lowers correlation** between trees, results in **diverse set** of trees, **improving** model **accuracy** by **reducing overfitting** and increasing the **diversity** of the model.
Less computationally expensive.	More computationally expensive.

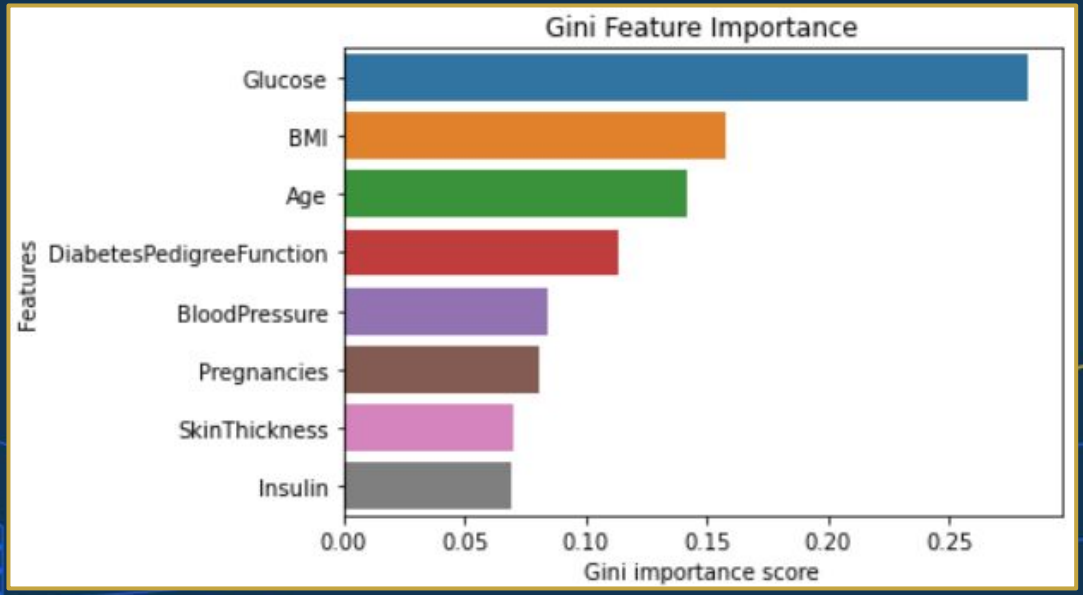




m = √M is a good place to start

Feature importance

- **Feature importance** calculates a **score** for all the input features for a given model to establish the "**importance**" of each feature in the decision-making process. The **higher the score for a feature**, the **larger effect** it has on the model to predict a certain variable.



CoGrammar

Hyperparameter Tuning

Hyperparameter to increase the Predictive Power

- **n_estimators:** number of trees the algorithm builds before taking the maximum voting or taking the averages of predictions. In general, a higher number of trees increases the performance and makes the predictions more stable, but it also slows down the computation.
- **max_features**: maximum number of features RF considers to split a node.
- **min_sample_leaf**: minimum number of leafs to split an internal node.

Hyperparameters to increase the RF model's speed

- **n_jobs, random_stat, oob_score** (out-of-bag sampling).

CoGrammar

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Image Processing

CoGrammar

Image Processing

- Image processing refers to the **manipulation and analysis of digital images** to **extract meaningful information** and enhance visual content.
- Enables computers to **interpret and understand visual data**.
- **Digital Image:** A digital image is represented as a two-dimensional array of pixels, where each pixel contains intensity or colour values.

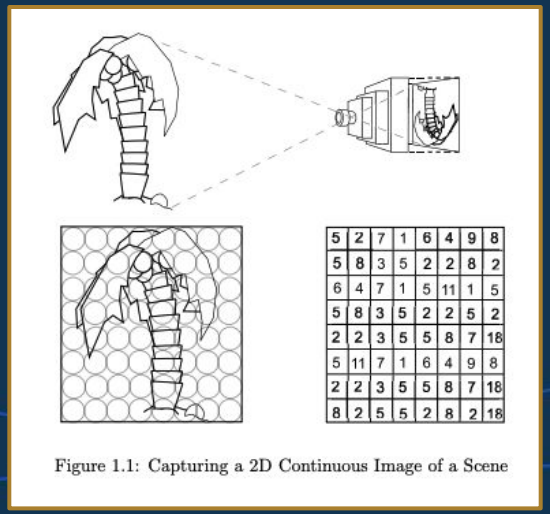
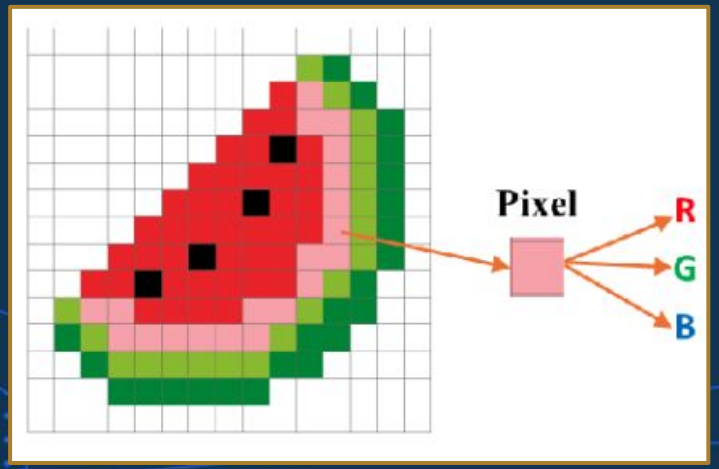


Figure 1.1: Capturing a 2D Continuous Image of a Scene

Source: Clemson University

Key Concepts: Pixel

- **Pixel:** A pixel is the smallest unit of a digital image and is represented by **one or more numerical values**. In grayscale images, each pixel has a single intensity value, while in colour images, pixels typically have three values corresponding to the **red, green, and blue** colour channels.

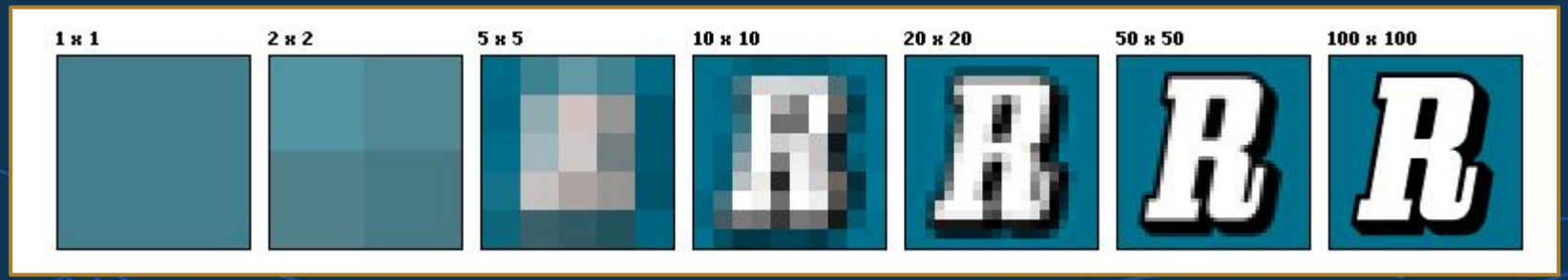


Source: ResearchGate

CoGrammar

Key Concepts: Image resolution

- **Image Resolution:** Image resolution refers to the number of pixels in an image, usually expressed in terms of **width and height** (e.g., 1024x768). Higher resolution images contain more pixels and provide **more detailed and clearer representations** of the visual content.



Source: Wikipedia

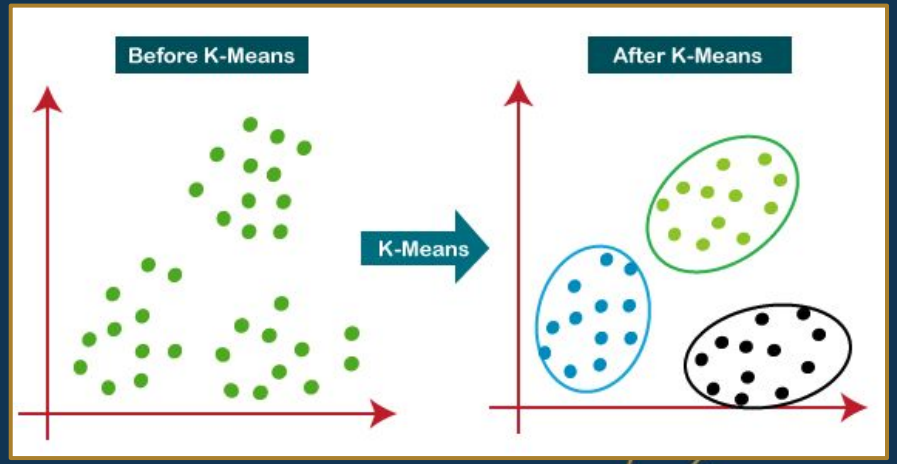
CoGrammar

K-means Clustering

CoGrammar

K-means Clustering

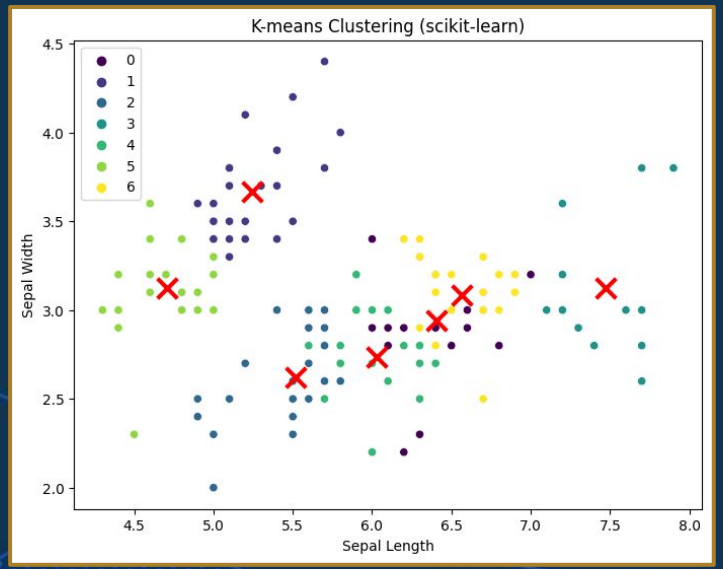
- K-means clustering is an **unsupervised learning algorithm used to partition a dataset into K distinct clusters.**
- The goal is to **minimise the within-cluster variation** and **maximise the separation between clusters**.
- It is a popular technique for **exploratory data analysis** and **pattern discovery**.



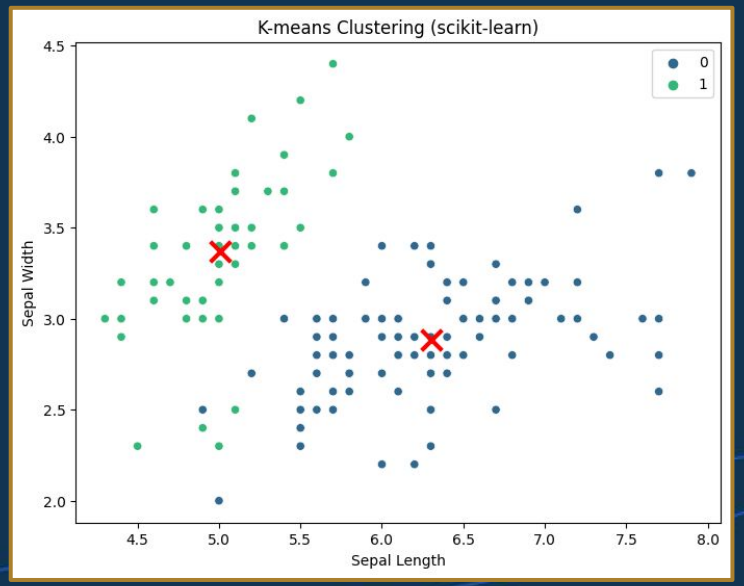
Source: Javapoint

Choosing K

Selecting an appropriate number of clusters (K) is crucial for effective clustering.



VS

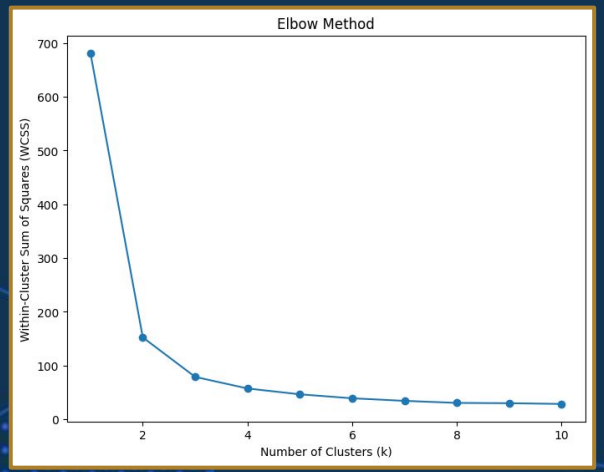


CoGrammar

Techniques for choosing K

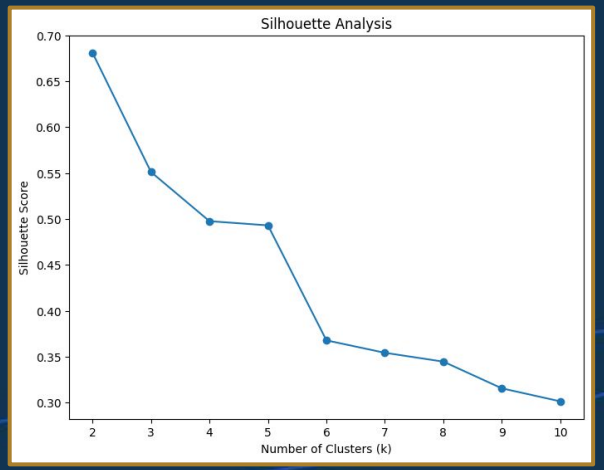
Elbow Method:

- Plot the within-cluster sum of squares (WCSS) against different values of K.
- Look for the **"elbow point"** where the rate of decrease in WCSS slows down significantly.



Silhouette Analysis:

- Calculate the silhouette score for each data point and average them for different values of K.
- Choose the K value that **maximises the average silhouette score.**



CoGrammar

Cluster Profiling

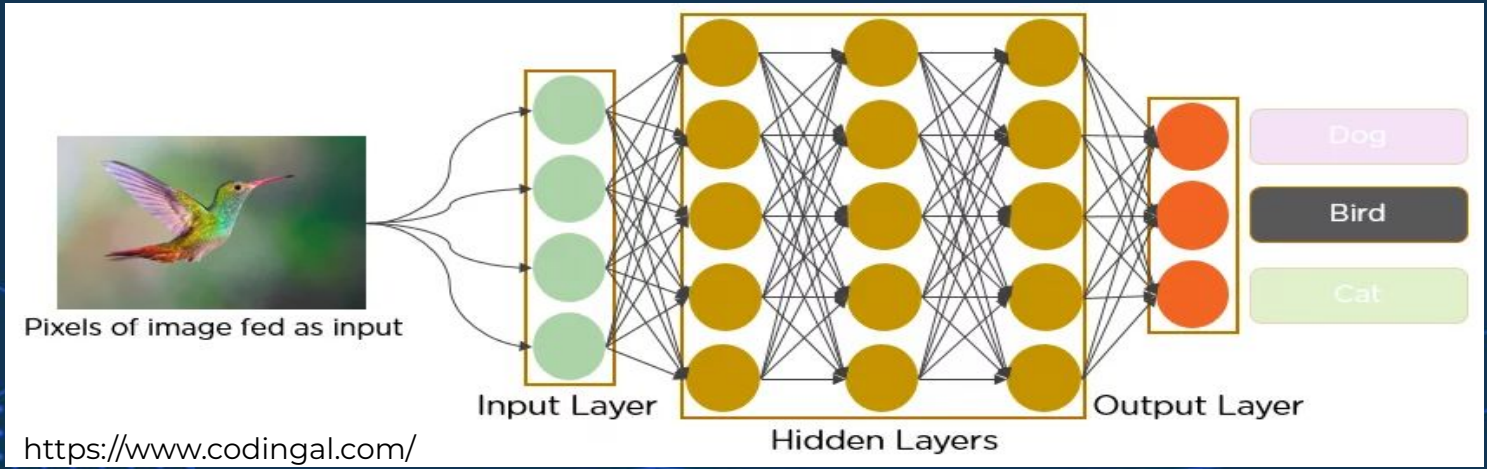
- After obtaining the clustering results, it's important to **interpret and analyse the clusters.**
- Examine the characteristics and centroids of each cluster to **understand their distinguishing features.**
- Assign **meaningful labels or descriptions** to the clusters based on domain knowledge and the patterns observed in the data.

Neural Networks

CoGrammar

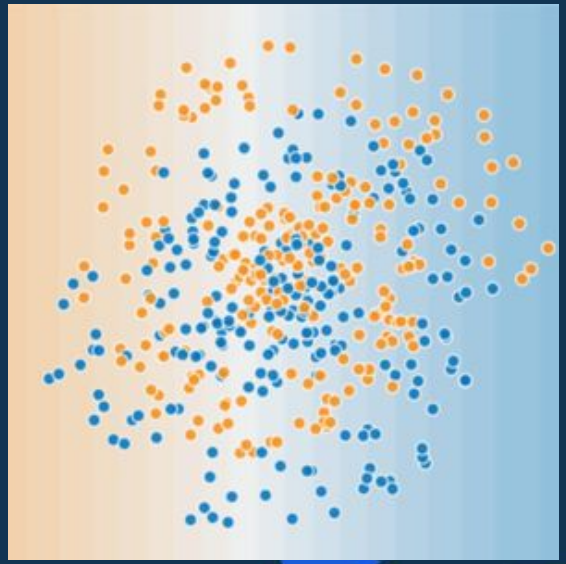
Neural Networks

- A **neural network (NN)** is a machine learning model inspired by the structure and function of the central nervous system in a human body.
- Consist of interconnected **nodes or neurons,** processing units that pass data to each other, like in the brain, biological neurons pass electrical impulses to each other.
- Enable tasks such as **pattern recognition** and **decision making** in machine learning.



Importance of Neural Networks

- Ideally suited to solve **complex and nonlinear** problems
- Make generalisations and inferences
- Reveal **hidden relationships, patterns** and **predictions**
- Model **highly volatile data** (financial time series data) and variances needed to **predict rare events** (fraud detection).

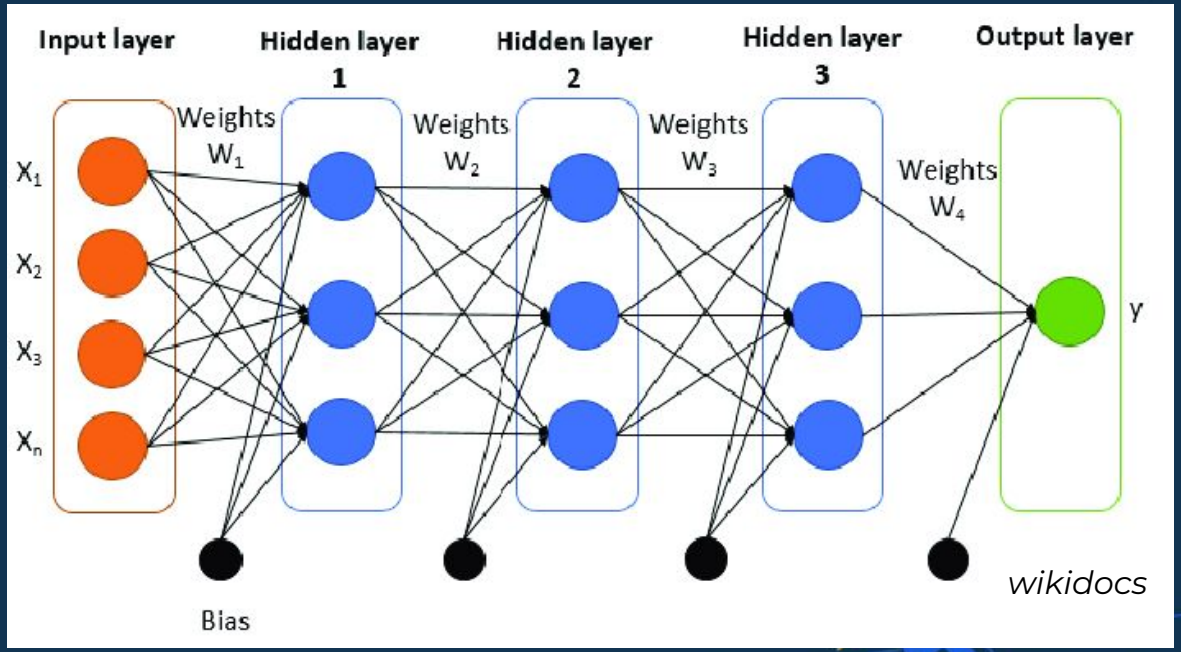


Non-linear classification

Neural Network Layers

Input layer: sends the data to subsequent layers, no. of nodes depend on features

Hidden layer/s: weighted inputs fed into these intermediate layers for computations; extracts features from the data.



Output layer: takes input from preceding hidden layers and comes to a final prediction based on the model's learnings

CoGrammar

Neural Network Layers

- **Input layer:** number of neurons in the input layer is equal to the number of features in the data, sometimes one additional for bias.

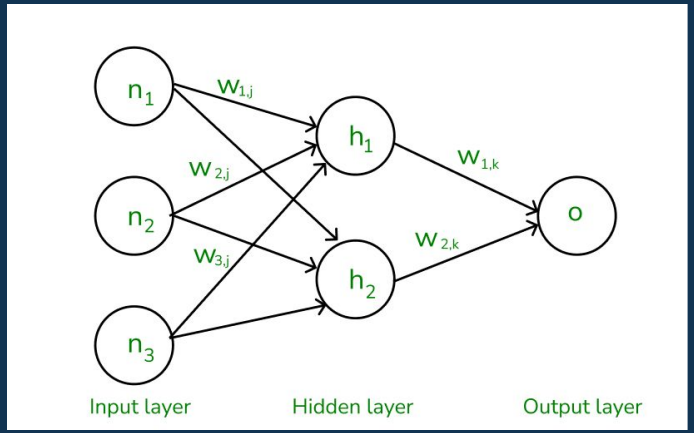
- **Hidden layer/s:** intermediate layer/s between input and output layer where all the computation is done. If number of layers is
  - 0: Only capable of representing linearly separable data
  - **1 - 2:** Data is less complex and have fewer dimensions or features
  - More layers for optimum solution in large dimensions/many features

Most common

- **Output layer:** number of neurons depends on whether the model is a regressor (only one neuron) or classifier (one neuron for each class label).

Neural Network Components

- Components include **neurons, connections, weights, biases, activation functions**.
- **Neurons** receive inputs, governed by **thresholds** and **activation functions.**

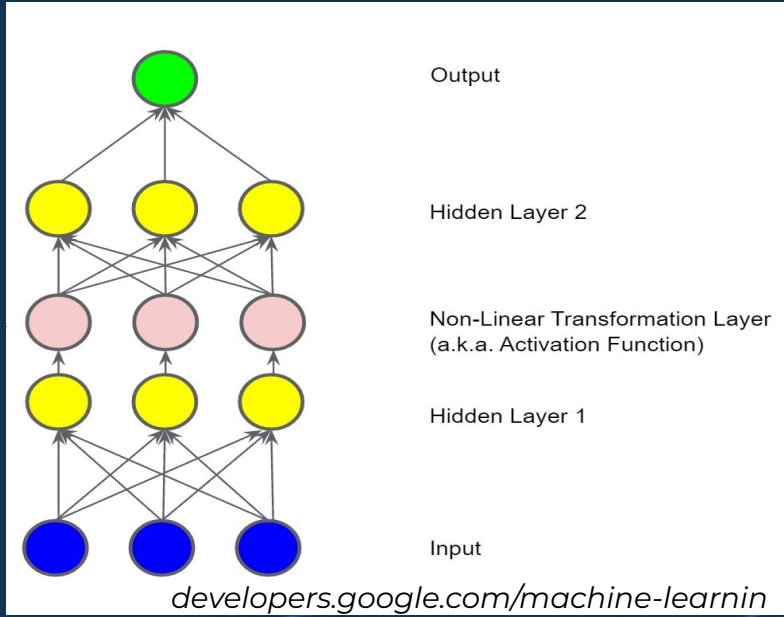


- **Connections** involve **weights** and **biases** regulating information transfer.
- Learning, adjusting weights and biases, occurs in **three stages: input computation, output generation, and iterative refinement** enhancing the network's proficiency in diverse tasks.

Activation Functions

To introduce **non-linearity** to model **non-linear problems**

- Pipe each hidden layer node through a nonlinear function.
- The value of each node in Hidden Layer 1 is transformed by a nonlinear function before being passed on to the weighted sums of the next layer.
- This nonlinear function is called the **activation function.**

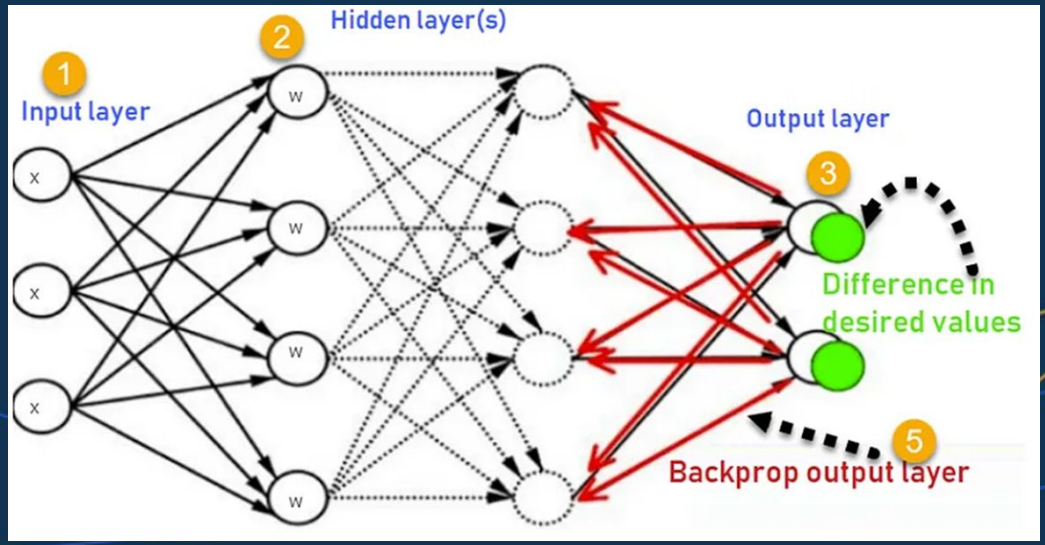


CoGrammar

Backpropagation

- Backpropagation algorithm widely used to train feedforward NNs. It **minimises the cost function** by adjusting network's **weights** and **biases**.
- The level of adjustment is determined by the **gradients of the cost function** with respect to those parameters.

- **Gradient descent** or **stochastic gradient descent** estimation method used by the optimisation algorithm to compute the network parameter updates and train neural network models.



CoGrammar

blog.stackademic

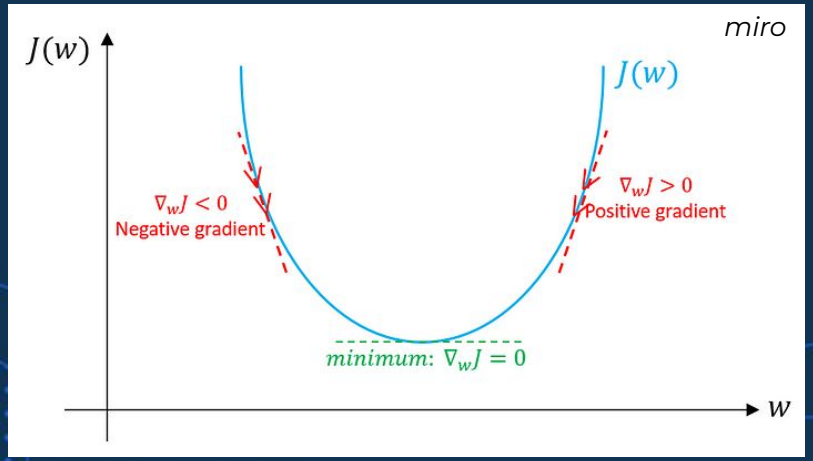
Gradient Descent

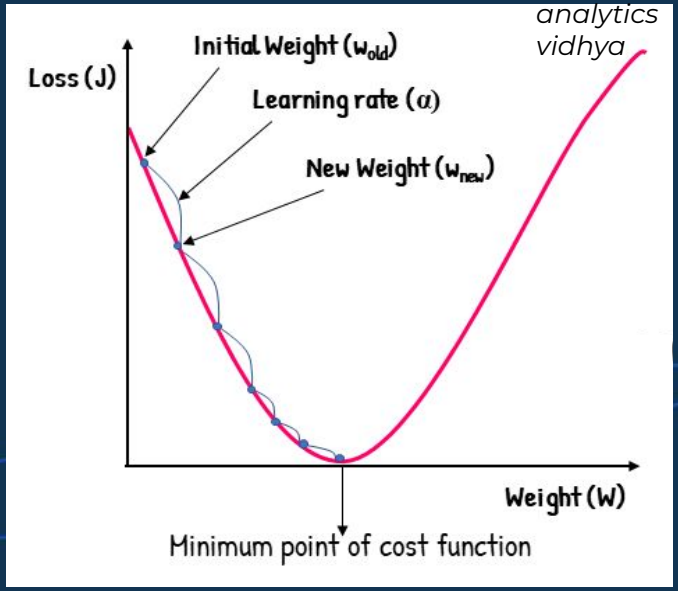
Updating the weights

$$w_{new} = w_{old} - \alpha \frac{\delta J}{\delta w}$$

α = Learning rate
$\delta J/\delta w$ = partial derivative of the loss function for each weight w (rate of change of the loss function to the change in weight.)

Gradient of J = ∇ J has all the partial derivatives (**∇ = Del / nabla operator)**



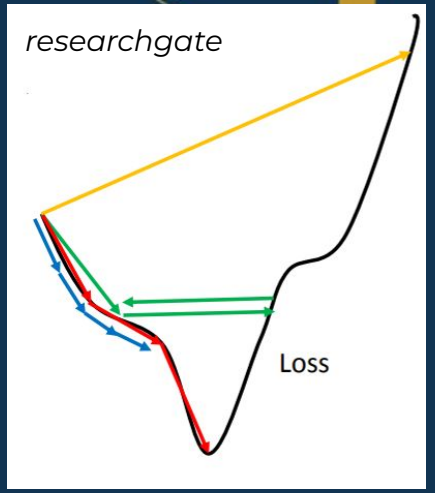


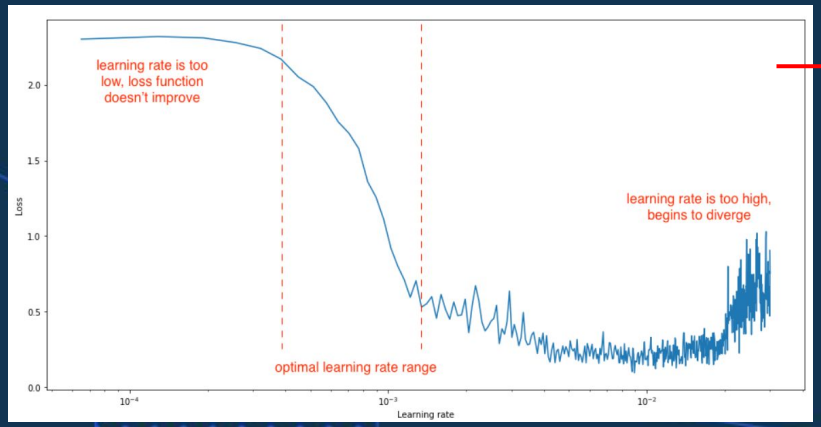
CoGrammar

Learning Rates

High learning rates: result in larger steps but risks overshooting the minimum. (yellow and green lines)

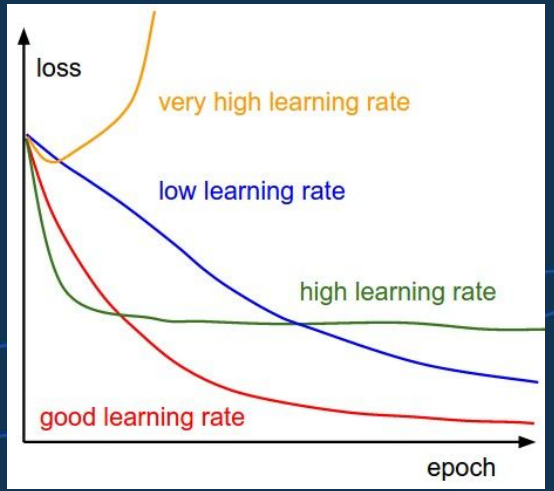
Low learning rate: small step sizes, more precision, but compromises overall efficiency, more iterations takes more time and computations to reach the minimum (blue lines)





jeremyjordan.me

Set learning rate bounds to observe all three phases, making the optimal range trivial to identify.



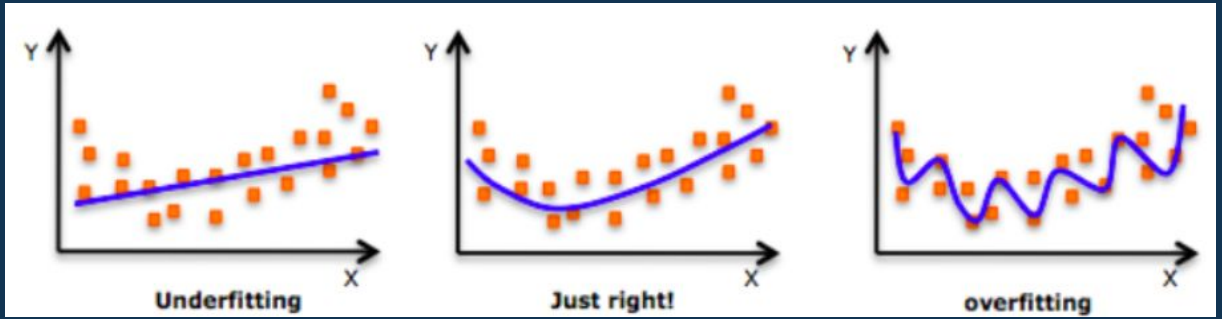
CoGrammar

Loss Function

- A **loss function or cost function** measures how good a neural network model is in performing a regression or classification task.
- Minimise the loss function value during the backpropagation step to make the neural network perform better.
- The kind of loss function used depends on the problem:
  - **Binary cross-entropy** is used when performing binary classification, and **categorical cross-entropy** is used for multi-class classification.
  - **MSE loss function** used for **regression** tasks, when we want the network to predict continuous numbers.
  - **MAPE loss function** used during **demand forecasting** to check network performance during training time, **regression** tasks.

Regularisation Techniques

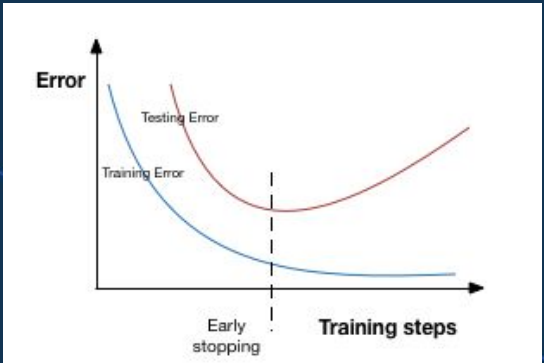
Regularisation techniques improve neural network's generalisation ability by reducing overfitting.



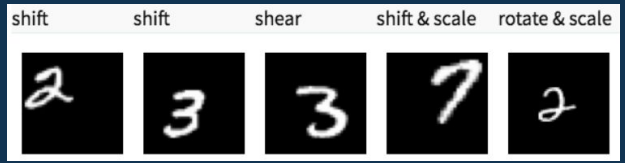
- Done by minimising complexity, exposing network to more diverse data.
- Can add a penalty term to the loss function during training which discourages the model from becoming too complex or having large parameter values.
- Models become more robust and better at making accurate predictions on unseen data.

Regularisation Techniques

Early stopping: Prevent training loss from becoming arbitrarily low, model is less likely to overfit on training dataset, and will generalise better.



Data augmentation: e.g. applies transformations to **images** (rotating the image, flipping, scaling, shifting) to create a larger dataset.



L1 and L2 regularisation prevent overfitting by adding a penalty term to the loss function.

$$\lambda \sum_{i=1}^{N} |\theta_i|$$

$$\lambda \sum_{i=1}^{N} \theta_i^2$$

CoGrammar

Questions and Answers

CoGrammar

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar