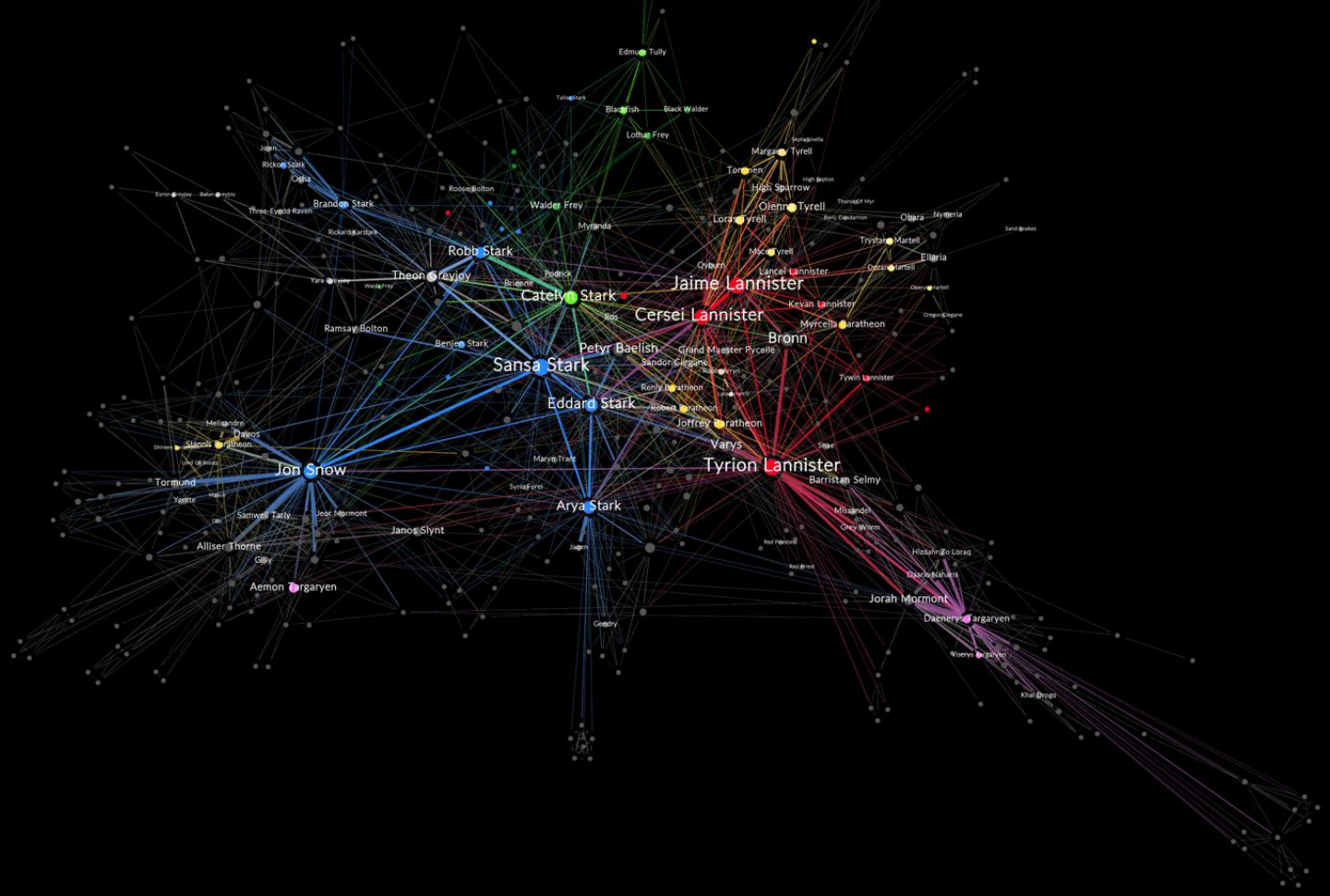
PUSL3123

AI & Machine Learning



Chamali Ranasinghe

21.1

Artificial intelligence

Artificial Intelligence is the simulation of human intelligence processes by machines, especially computer systems.

	Thinking humanly	Thinking rationally	Acting humanly	Acting rationally
Definition	This is about making machines think and act just like humans, with emotions, intuition, and common sense. It's like trying to create AI that understands and behaves like a person.	Here, we want machines to think logically and make decisions based on rules and facts, similar to how a computer follows a set of instructions without emotions or intuition.	This approach involves making machines act in a way that's indistinguishable from a human, like having a conversation that's so natural you can't tell you're talking to a machine.	In this case, AI makes decisions that are smart and well-optimized, even if it doesn't mimic human behavior. It's about acting in a way that achieves the best results based on available information, like a robot making decisions to maximize efficiency.
Example	General problem solver	All computers use energy. Using energy always generates heat. Therefore, all computers generate heat.	The turning test	an agent that is playing a game will act rationally if it tries to win the game.
Difference	Thinking Humanly aims to make AI think and act like humans, with emotions and intuition, while Thinking Rationally emphasizes logical reasoning and following formal rules. Human-like vs. Logic-based.		Acting Humanly is about making AI's behavior indistinguishable from humans, while Acting Rationally is focused on optimizing decisions for the best outcomes, even if it doesn't mimic human behavior. Human-like behavior vs. Goal-oriented, efficient behavior.	

Core components of AI

- ♠ Generalized Learning
  - → Demonstrate how well is a trained model to classify or forecast unseen data.
- ♠ Reasoning
  - → The logical process of drawing conclusions, making predictions, or constructing approaches towards a particular thought with the help of existing knowledge.
- ♠ Problem Solving
  - → Reach the desired goal or find a solution to a given situation.

Advantages of AI	Disadvantages of AI
High Accuracy with less errors	High Cost
High-Speed	Can't think out of the box
High reliability	No feelings and emotions
Useful for risky areas	Increase dependency on machines
Digital Assistant	No Original Creativity
Useful as a public utility	

Examples for AI

1. Self-driving cars
2. Recommendation systems
3. Computer vision

Strong AI and weak AI

- Strong AI refers to artificial intelligence systems that possess the ability to understand, learn, and apply knowledge across a wide range of tasks and domains just like a human being.

- These systems have general intelligence and consciousness. They can reason, learn from experience, understand natural language, and perform any intellectual task that a human can do. Strong AI remains a theoretical concept and has not yet been achieved.

- Weak AI, on the other hand, refers to artificial intelligence systems that are designed and trained for a specific task or a narrow set of tasks. These systems are not conscious or self-aware.

- They excel in performing well-defined, specialized functions but do not possess general intelligence.

- Examples of weak AI include virtual personal assistants like Siri or Alexa, recommendation algorithms, and chess-playing computers.

Challenges of AI

Algorithms:

- Algorithms are at the heart of AI. Developing effective algorithms for various tasks is challenging because different problems require different approaches. Finding the right algorithm, optimizing it, and adapting it to real-world scenarios can be complex. Moreover, AI algorithms should be efficient, accurate, and capable of handling large amounts of data.

Data Availability:

- AI systems need a significant amount of data to learn from. In some cases, obtaining enough high-quality data can be a challenge. Data availability can be limited, especially for specialized or niche domains. Additionally, access to data may be restricted due to privacy concerns or proprietary reasons, making it challenging to train AI models effectively.

Data Quality:

- Even when data is available, ensuring its quality is crucial. Data may be noisy, incomplete, or biased. Low-quality data can lead to inaccurate AI models and flawed decision-making. Cleaning and preprocessing data to ensure its accuracy and reliability is a significant challenge in AI.
- Key components of quality data in AI
    - → Accuracy
    - → Consistency
    - → Completeness
    - → Timeliness
    - → Relevance

Cybersecurity & Privacy:

- With the increasing use of AI, there are growing concerns about cybersecurity and privacy. AI systems can be vulnerable to attacks, and their use can pose privacy risks, particularly when handling sensitive personal or business data. Safeguarding AI systems from cyber threats and ensuring the privacy of individuals' data is a considerable challenge in the AI field. Complying with privacy regulations like GDPR is also a concern.

Ethical challenges of AI

Biases – how to eliminate bias from the dataset?

Control and the mortality of AI – does AI in control of the situation?

Privacy – what are the rules around data collection?

What legislation might need to be put in place to protect users' private information?

Ownership – who is responsible for some of the things that AIs are creating?

Humanity – how does AI make us feel as humans?

```
clear all;
```

This command clears all variables and their values from the workspace, ensuring a clean slate for your MATLAB session.

```
A = [2 100 4 11];
```

A is created as a row vector with the elements [2, 100, 4, 11].

```
B = [2, 0, 5 ,1];
```

B is created as another row vector with the elements [2, 0, 5, 1].

```
C= [4 3 1; 3 3 2; 9 1 0];
```

C is created as a 3x3 matrix with specified values in the form of a matrix.

```
Zero = zeros(3,1);
```

Zero is created as a 3x1 matrix filled with zeros.

```
Ones = ones(3,2);
```

Ones is created as a 3x2 matrix filled with ones.

```
OP=A+3;
```

OP is created by adding 3 to each element of array A.

```
OP1=OP';
```

OP1 is created by transposing the matrix OP. This flips the rows and columns.

```
Newarray=[A, B];
```

Newarray is created by concatenating arrays A and B horizontally. It results in a row vector [2, 100, 4, 11, 2, 0, 5, 1].

```
Col_Row_Equal = magic(4);
```

Col_Row_Equal is created using the magic(4) function, which generates a 4x4 matrix where the sums of each row and column are equal.

```
Max_Col=max(Col_Row_Equal); % returns the maximum elements of an array.

Min_FirstCol=min(Col_Row_Equal (:,1)); % returns the maximum element of first column.

Min_FirstRow=min(Col_Row_Equal (1,:)); % returns the maximum element of row.

Mat_Size=size(Col_Row_Equal) % returns a row vector whose elements are the lengths of the corresponding
```

dimensions of matrix

```
x = linspace(0, 2*pi);
```

This line generates a row vector x containing 100 evenly spaced values between 0 and 2π (i.e., a full cycle of the trigonometric functions sine and cosine). linspace is a function that creates a linearly spaced vector of values within the specified range.

```
y = sin(x)
```

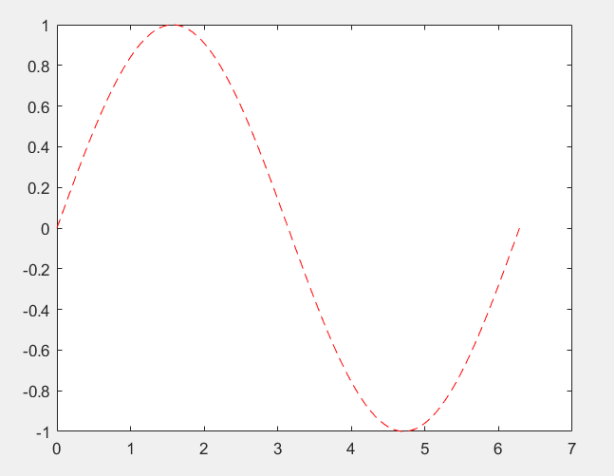
This line calculates the sine of each value in the vector x and stores the results in the vector y. Essentially, it computes the sine function for each value of x.

```
xlabel("x"), ylabel("sin(x)"), title("Plot of the Sine Function")
```

These lines set labels for the x-axis, y-axis, and the title of the plot, respectively. The labels help provide context and understanding of what the plot represents.

```
plot(x, y, "r--")
```

This line creates a plot of the sine function by specifying the x-values from the x vector and the corresponding y-values from the y vector. It also sets the line style to a red dashed line ("r--"). This line style specifies that the plot should use a red dashed line to connect the data points.




```
hold on
```

This command tells MATLAB to keep the current plot active.

```
y2 = cos(x);
```

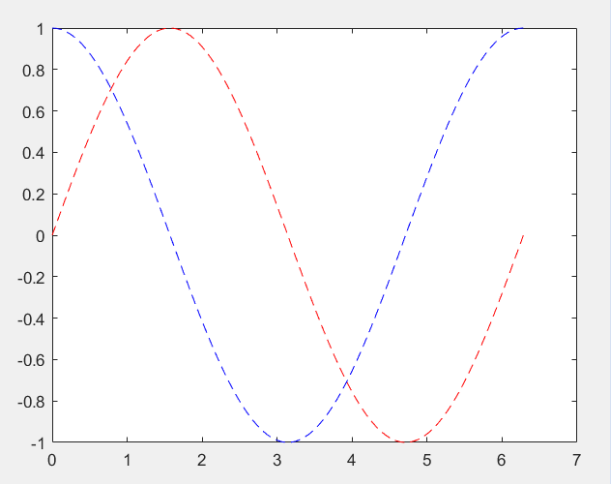
This line calculates the cosine of each value in the vector x and stores the results in the vector y2. It computes the cosine function for each value of x.

```
plot(x, y2, "b--")
```

This line creates a plot of the cosine function in blue dashed line style ("b--"). The hold on command ensures that this plot is overlaid on top of the previous sine function plot.

plot(x, y2, "b--")

This line creates a plot of the cosine function in blue dashed line style ("b--"). The hold on command ensures that this plot is overlaid on top of the previous sine function plot.



hold off

This command turns off the "hold" mode.

In summary, this code generates a plot with the sine function in red and the cosine function in blue, both represented with dashed lines, over a range of values from 0 to 2π. It adds labels and a title for clarity and ensures that both sine and cosine plots are displayed on the same graph by using the "hold" feature.

t = tiledlayout(2,2);

This line creates a 2x2 tiled layout, which means there will be four subplots arranged in a 2x2 grid. It returns a tiledlayout object assigned to the variable t.

title(t, "Trigonometric Functions")

This line sets the title of the entire tiled layout to "Trigonometric Functions" using the title function and the tiledlayout object t.

x = linspace(0, 30)

This line generates a row vector x containing evenly spaced values between 0 and 30. The linspace function is used to create these values.

nexttile

The nexttile function is called four times in a row, and each time it advances to the next subplot in the tiled layout. In this case, it goes through the subplots in a left-to-right, top-to-bottom order.

plot(x, sin(x))

This line creates a plot in the current subplot. It plots the sine function, sin(x), against the values in the vector x.

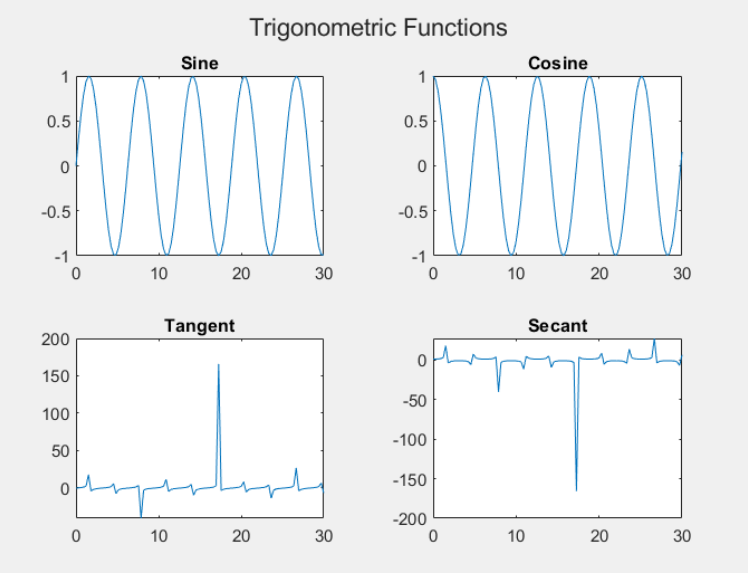
title("Sine") - This line sets the title for the current subplot as "Sine."

nexttile (repeated for the remaining three subplots):

These nexttile calls move to the next subplot in the 2x2 grid.

plot(x, cos(x)), plot(x, tan(x)), plot(x, sec(x))

These lines, in their respective subplots, create plots of the cosine, tangent, and secant functions against the values in the vector x. Each subplot represents a different trigonometric function.



title("Sine") - This line sets the title for the current subplot as "Sine."

nexttile (repeated for the remaining three subplots):

These nexttile calls move to the next subplot in the 2x2 grid.

plot(x, cos(x)), plot(x, tan(x)), plot(x, sec(x))

These lines, in their respective subplots, create plots of the cosine, tangent, and secant functions against the values in the vector x. Each subplot represents a different trigonometric function.

Generate a 2D distribution.

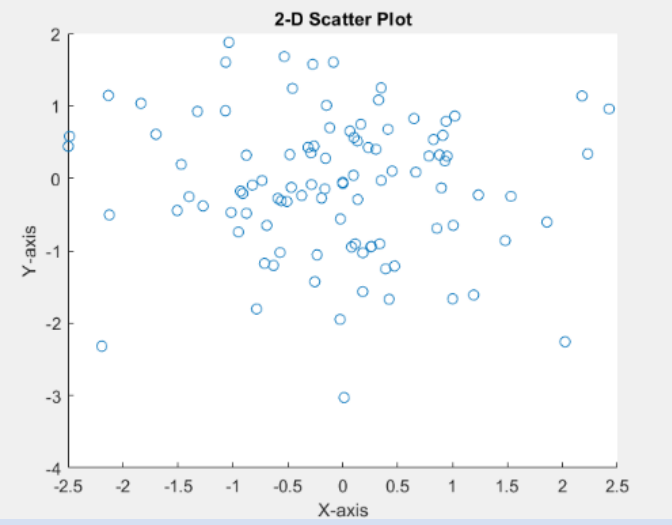
```
n_samples = 100; % You can change this to the desired number of samples
data = randn(2, n_samples);
data_size = size(data);
fprintf('Size of the data array: %d x %d\n', data_size(1), data_size(2));
scatter(data(1, :), data(2, :));
title('2-D Scatter Plot');
xlabel('X-axis');
ylabel('Y-axis');

mean_vector = mean(data, 2);
fprintf('Mean Vector: [%f, %f]\n', mean_vector(1), mean_vector(2));

cov_matrix = cov(data');
fprintf('Covariance Matrix:\n');
disp(cov_matrix);

```

Command Window

```
>> rev1
>> rev1
Size of the data array: 2 x 100
>> rev1
Size of the data array: 2 x 100
Mean Vector: [-0.080676, -0.094861]
>> rev1
Size of the data array: 2 x 100
Mean Vector: [0.059038, -0.000140]
Covariance Matrix:
    0.8395    0.1153
    0.1153    0.8445
```

scatter(data(1, :), data(2, :))

This line creates a 2-D scatter plot using the data points from the data matrix. The first row (data(1, :)) represents the X-axis values, and the second row (data(2, :)) represents the Y-axis values.



Machine Learning

Machine learning is like teaching a computer to learn and make decisions on its own. Instead of giving it specific instructions, you show it examples and let it figure things out. It's how computers can get better at tasks by learning from data, much like how you get better at a game by playing it over and over.

Applications of Machine Learning

1. Image Recognition

   Identify objects, persons, places, digital images, etc

2. Speech Recognition

   Speech recognition is a process of converting voice instructions into text, and it is also known as "Speech to text", or "Computer speech recognition.“. For example, Search by voice in Google

3. Medical Diagnosis

   Machine learning is used for Disease identification. For example, It helps in finding brain tumors and other brain-related diseases easily.

Types of machine learning

Supervised learning

In supervised learning, the computer is trained using labeled data.

Labeled data means each input (like an image, text, or data point) is paired with the correct output or answer.

The goal is for the machine to learn a mapping from inputs to outputs.

It's like a teacher supervising a student's learning process by providing answers to questions in a textbook.

This method is used for tasks like image recognition, language translation, and spam email classification.

unsupervised learning

Unsupervised learning deals with unlabeled data.

Here, the computer's task is to find patterns, structures, or relationships within the data without being given explicit answers.

It's like sorting a mixed bag of marbles into groups of similar colors without being told which colors exist.

Unsupervised learning is used in clustering, dimensionality reduction, and anomaly detection.

reinforcement learning

Reinforcement learning is about training an agent to make a sequence of decisions in an environment to maximize a reward.

The agent receives feedback in the form of rewards or punishments based on its actions.

It learns through trial and error, gradually discovering the best strategies for achieving its goals.

Think of it as teaching a dog new trick through a series of rewards (treats) and punishments (no treats).

It's widely used in areas like game playing, robotics, and autonomous systems.

How Does Machine Learning Work?

1. Choose and Prepare a Training Data Set

Training data can be classified into two categories: **labelled data** and **unlabelled data**.

Labelled data

- ♠ It is a group of data samples tagged with one or more meaningful labels. Labels used to identify specific characteristics, properties, classifications, or contained objects.
- ♠ Labelled training data is used in supervised learning.

Unlabelled data

- ♠ It is raw data or data that's not tagged with any labels for identifying classifications, characteristics, or properties.
- ♠ Unlabelled data used in unsupervised machine learning.

2. Select an Algorithm to Apply to the Training Data Set

The selection process of machine learning algorithm depends on a few aspects.

- ♠ Whether the use case is prediction or clustering
- ♠ How much data is in the training set.
- ♠ The nature of the problem the model seeks to solve.

3. Train the Algorithm to Build the Model

Setting model variables and parameters to more accurately predict the appropriate results.

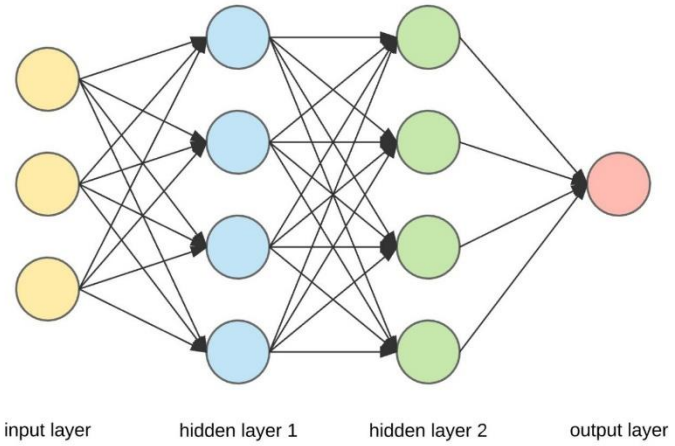
Using a variety of optimization methods depending upon the chosen model.

4. Use and Improve the Model

- ♠ Feed new data to the model as a means of improving its effectiveness and accuracy over time.
- ♠ Feature Engineering which creates new features from the existing ones.
- ♠ Feature Selection that helps to identify the most useful features in the dataset.
- ♠ Try Multiple Algorithms to identify which ones work best for data and then use that information to improve the accuracy of models.
- ♠ Adjusting Hyperparameters such as number of layers in a deep neural network.
- ♠ If the prediction and results don't match, re-train the algorithm multiple times until getting the desired outcome.

Neural Networks

Neural networks reflect the behavior of the human brain, allowing computer programs to recognize patterns and solve common problems in the fields of AI, machine learning, and deep learning.



Input Layer:

- This is where the neural network receives its initial data or input.
- Each neuron in the input layer represents a feature or attribute of the data you want the network to process.
- For example, if you're working with images, each neuron might represent a single pixel's brightness. The input layer doesn't perform any computations; it just passes the data to the next layer.

Hidden Layers:

- Hidden layers are intermediate layers in the neural network where the actual computation and information processing occur.
- These layers are called "hidden" because you don't directly interact with them; they're not part of the input or output.
- Each neuron in a hidden layer takes input from the previous layer, processes it using a weighted sum and activation function, and passes the result to the next layer.

Output Layer:

- The output layer is the final layer of the neural network, and it produces the network's predictions or outputs.
- The number of neurons in this layer depends on the specific task you're working on.
- For instance, in a binary classification task (e.g., "yes" or "no"), there would be two neurons, each representing a class.
- The output layer processes the information learned in the hidden layers to produce the final result.

Key Components of the Neural Network Architecture

- Connections—It connects one neuron in one layer to another neuron in other layer or the same layer.
- Weight - Its main function is to give importance to those features that contribute more towards the
- learning. A weight represents the strength of the connection between units.
- Bias - The role of bias is to shift the value produced by the activation function.
- Activation Function - decides whether a neuron should be activated or not- such as Sigmoid, Hyperbolic Tangent (Tanh), Rectified Linear Unit (ReLU).

Feedforward vs. Backpropagation

Feedforward:

- Feedforward is the forward pass through a neural network, where input data is processed layer by layer from the input layer to the output layer.
- Each neuron in a layer takes input from the previous layer, applies a weighted sum and an activation function, and passes the result to the next layer.
- This process continues until the output layer produces predictions or results. It's called "feedforward" because information flows in one direction, from input to output, without feedback or corrections.

Backpropagation:

- Backpropagation is the process of training a neural network.
- After the initial feedforward pass, it's likely that the predictions are not accurate.
- Backpropagation is used to adjust the weights and biases of the neurons in the network to minimize the difference between the predicted output and the desired output (the training target).
- It's like a feedback mechanism where the error is calculated at the output layer and then propagated backward through the network to update the weights and biases of each neuron layer by layer.
- This process iterates many times until the network's predictions become more accurate.

Error functions

Mean Squared Error (MSE):

- MSE is one of the most widely used error functions, primarily for regression problems.
- It calculates the average squared difference between predicted and actual values.
- MSE = (1/n) ∑(i=1 to n) (predicted_i - actual_i)^2, where n is the number of data points.

Root mean square error (RMSE)

- Root Mean Square Error (RMSE) is a commonly used error metric in various fields, especially in the context of regression analysis and predictive modeling.
- It is used to evaluate the accuracy of a predictive model by measuring the average magnitude of the errors between predicted values and actual (observed) values.

The perceptron

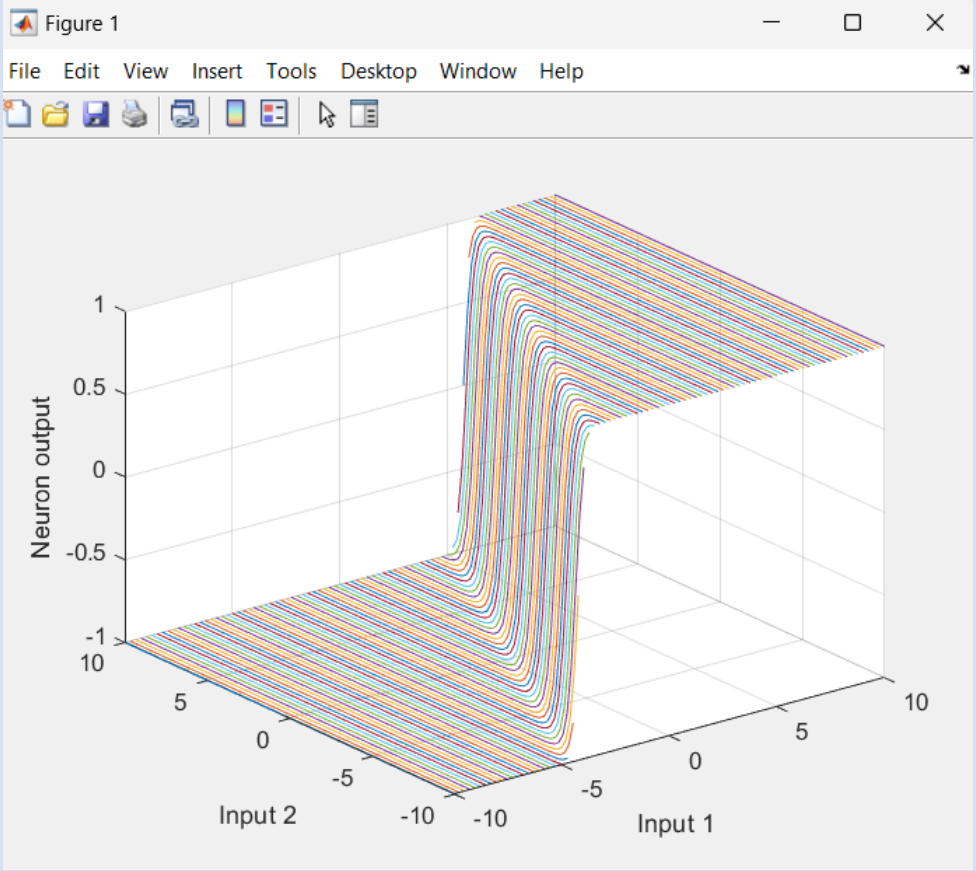
The perceptron is a fundamental building block in artificial neural networks and a simple type of neural network model. It's often used for binary classification tasks, where it learns to separate data into two categories. The key characteristics of a perceptron are as follows:

- Inputs: A perceptron receives one or more binary inputs. These inputs can represent features or attributes of data, and each input is associated with a weight.
- Weights: Each input has an associated weight, which represents the importance of that input in making the final decision. The weights are learnable parameters and can be positive or negative.
- Summation: The inputs are multiplied by their respective weights, and the weighted inputs are summed together. This weighted sum represents the strength of the evidence in favor of one class or the other.
- Activation Function: The weighted sum is passed through an activation function. The purpose of the activation function is to introduce non-linearity and determine the output of the perceptron. A commonly used activation function for perceptrons is the step function, which outputs 1 if the weighted sum is greater than a threshold and 0 otherwise.
- Output: The output of the perceptron is a binary value (0 or 1), which represents the predicted class. It's the result of applying the activation function to the weighted sum.

The perceptron's learning process typically involves adjusting the weights in response to errors. Here's a simplified description of the learning process:

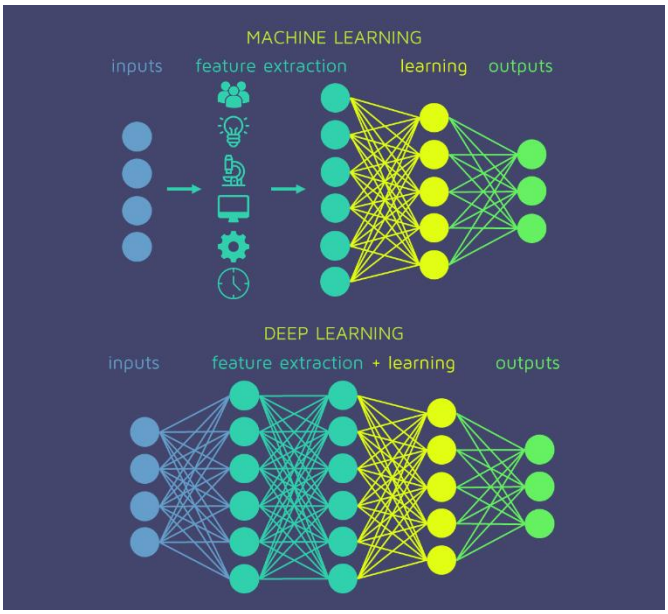
- Given a set of training data with known input-output pairs, the perceptron makes predictions.
- If a prediction is incorrect, the weights are adjusted to reduce the error. This is done through a learning algorithm that updates the weights based on the difference between the predicted output and the true output.
- The process continues iteratively until the perceptron makes fewer errors on the training data, or until a stopping criterion is met.

```
w = [4 -2]; % Neuron weights
b = -3; % Neuron bias
% Activation function
func = 'tansig'; % Here, you specify the activation function for the neuron. In this
case, 'tansig' represents the hyperbolic tangent sigmoid activation function.
v = [2 3]; % Calculate neuron output
activation_potential = v*w'+b;

%Plot neuron output over the range of inputs
[p1,p2] = meshgrid(-10:.25:10);
z = feval(func, [p1(:) p2(:)]*w'+b );
z = reshape(z,length(p1),length(p2)); %This reshapes the z vector to have the same
dimensions as p1 and p2.

plot3(p1,p2,z)
grid on
xlabel('Input 1')
ylabel('Input 2')
zlabel('Neuron output')
% Finally, the code creates a 3D plot of the neuron's output. p1 and p2 represent the
input values, and z represents the neuron's output. The plot is labeled with three
axes: 'Input 1', 'Input 2', and 'Neuron output'. The grid on command adds gridlines
to the plot.
```



Deep Learning

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example.



Traditional Pattern Recognition consists of two parts.

- Feature extraction
- Classifier (e.g. ANN)

The Growth of Deep Learning

- Increasing dataset sizes
- Increasing machine learning model sizes
- Availability of computational resources, especially GPU, which can handle high-dimensional inputs
- Availability of sufficient framework/platforms, such as TensorFlowFramework (open source) to coordinate the GPUs to work on complex ML tasks.

The MNIST Dataset

- The MNIST Dataset of handwritten digits, a subset of a larger set from NIST

Image Net

- ImageNET: over 14M images, over 21K synsets indexed.

Difficulties with deep learning

- Lots of data is required – this is becoming less of a challenge but there are still difficulties (e.g., getting labelled data).
- Training deep learning networks requires substantial computational power because of the massive number of networks weights to calibrate –GPUs often used.
- Deep neural networks are generally for supervised learning – if we want to achieve general AI, we will need unsupervised learning models.
- "Deep" refers to the network architecture rather than understanding.

CNN

A Convolutional Neural Network (CNN) is a type of artificial neural network designed to process and understand visual data like images and video. It's used in tasks such as image recognition, object detection, and more. Let's explain CNNs in a simple way:

- Layers of Filters: Imagine you have a photo. CNN works by looking at the photo through different "filters" or "windows." These filters are small grids that slide over the photo to examine small pieces of it at a time.
- Finding Features: As each filter slides over the image, it looks for specific features like edges, colors, or patterns. For example, one filter might look for vertical lines, another for curves, and another for corners.
- Stacking Filters: CNNs don't just use one filter; they have multiple layers of filters that learn to recognize increasingly complex features. Early layers might detect simple things like edges, while deeper layers detect more complex shapes or even specific objects.
- Pooling: After each layer of filters, there is a step called pooling or subsampling. It simplifies the information by taking the most important parts and reducing the size of the data.
- Fully Connected Layers: At the end of the CNN, there are fully connected layers that take the high-level features found by the filters and use them to make predictions. For example, if it's a CNN for recognizing cats and dogs, these layers determine if the image is more likely a cat or a dog.

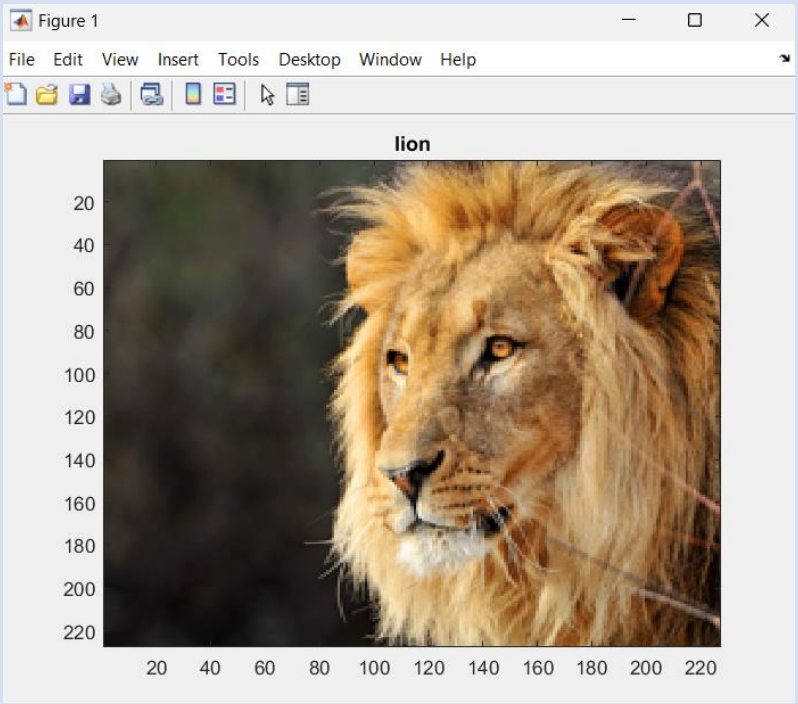
- Training: Before CNN can do all of this, it needs training. It shows lots of images with known labels (e.g., this is a cat, this is a dog), and it adjusts the filters' settings and parameters to get better at recognizing these labels.
- Prediction: Once trained, CNN can take any new image and give you a prediction, like "This is a cat with 80% confidence."

In simple terms, a CNN is like a smart detective for pictures. It looks at small parts of an image, gradually figuring out what's in the picture, and finally tells you what it thinks the picture contains. It's really good at finding patterns and shapes in images, which makes it great for tasks like identifying objects in photos or analyzing medical images.

Benefit of using CNN

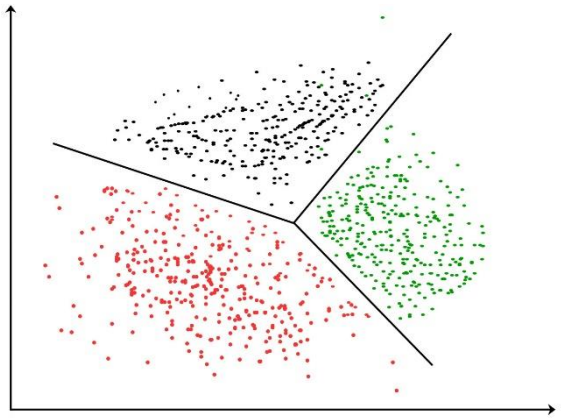
- CNNs eliminate the need for manual feature extraction—the features are learned directly by the CNN.
- Reduce the feature dimensionality while maintaining high recognition rate.
- CNNs can be retrained for new recognition tasks, enabling you to build on pre-existing networks.
- CNNs Scale up neural networks to process very large images / video sequences.

```
picture=imread("image1.JPG"); % This line reads an image named "image1.JPG" from the current directory and stores it in the variable picture.

nnet = alexnet; %Here, the code loads a pre-trained neural network called AlexNet and assigns it to the variable nnet. AlexNet is a popular convolutional neural network architecture for image classification tasks.

picture = imresize(picture,[227,227]); %The code resizes the picture to a specific size of 227x227 pixels. Neural networks often have input size requirements, and this resizing step ensures that the input image matches the required size for AlexNet.

label = classify(nnet, picture); %This line uses the pre-trained AlexNet (nnet) to classify the resized picture. The result of this classification is stored in the variable label. It identifies what object or category the image is most likely to represent.


image(picture); %Show the picture
title(char(label)); %This sets the title of the displayed image to be the classification label (the result of the classification) converted to a character format.

drawnow;
```



Clustering

Clustering is a fundamental concept in unsupervised machine learning and data analysis. It refers to the process of grouping similar data points together into clusters, where data points within the same cluster are more similar to each other than to those in other clusters. The primary goal of clustering is to discover hidden patterns or structures within a dataset without prior knowledge of the labels or categories.



Common uses of Clustering

- Recommendation engines
- Market Segmentation
- Statistical data analysis
- Social network analysis
- Image segmentation
- Anomaly detection

K-means clustering

K-Means is a popular clustering algorithm used in unsupervised machine learning to group similar data points into clusters. It is a straightforward and effective way to find patterns in data. Here's a more detailed explanation of how K-Means works:

- Initialization: Start by choosing the number of clusters you want to create, denoted as "K." This is a critical decision, as the algorithm will attempt to divide the data into K clusters.

Additionally, initialize K points in your dataset as the initial cluster centers. These points can be randomly selected from the data or using some other strategy.

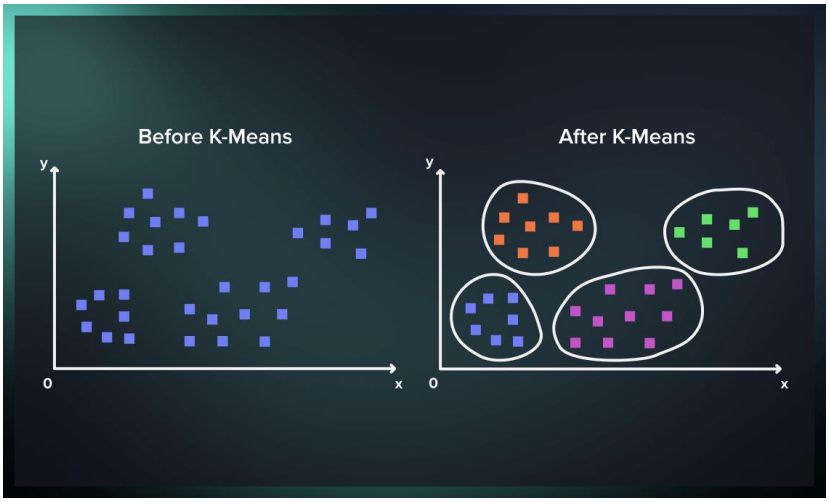
- Assignment: For each data point in your dataset, calculate the distance between that point and each of the K cluster centers. Assign the data point to the cluster whose center is closest (based on distance).

- Update Cluster Centers: After assigning all data points to clusters, recalculate the cluster centers by finding the mean (average) of all the data points in each cluster. These new cluster centers represent the "center of mass" for the data in each cluster.

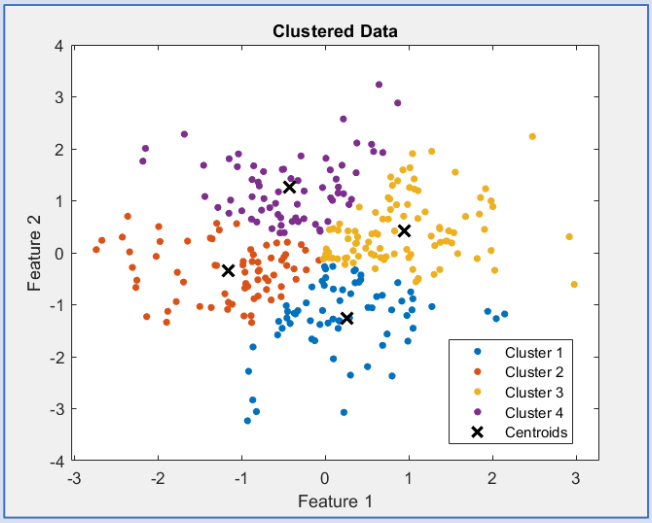
- Repeat: Iteratively repeat the assignment and center-updating steps until one of the stopping criteria is met:
    1. The cluster assignments do not change significantly.
    2. A set number of iterations have been completed.
    3. Other convergence criteria, like a threshold for changes in cluster centers, are met.

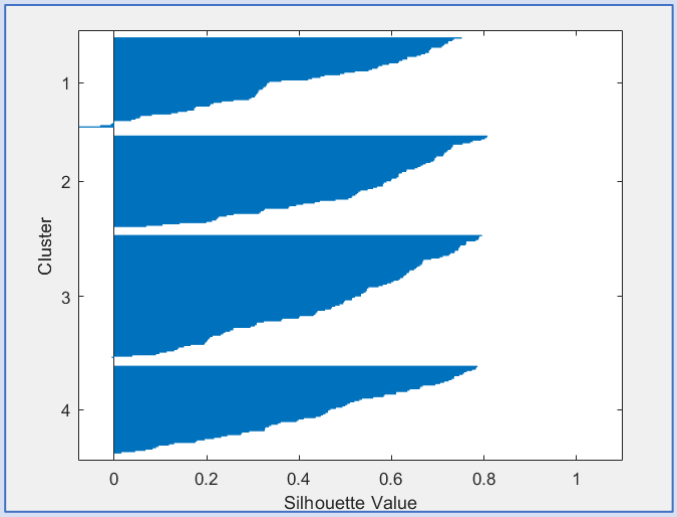
- Result: The algorithm stops, and you have K clusters, with each cluster containing data points that are more similar to each other than to those in other clusters.

K-Means is often used in various applications, such as customer segmentation, image compression, and data preprocessing. However, it has some limitations, such as its sensitivity to the initial placement of cluster centers, and it works best when clusters are roughly spherical and equally sized. Despite its simplicity, K-Means can be quite effective for many clustering tasks.



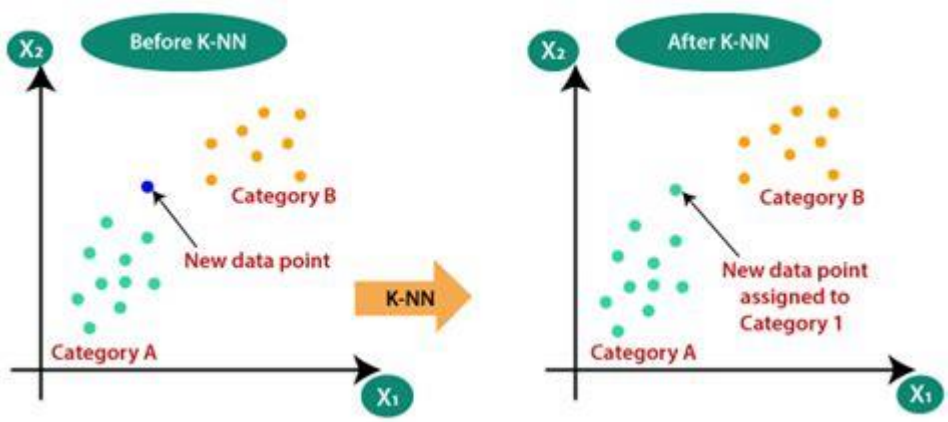
```
% Step 1: Generate a random dataset
rng(42); % Set the random seed for reproducibility
data = randn(300, 2); % Create a random dataset with 300 samples and 2 features

% Step 2: Create clusters (you can choose the number of clusters)
n_clusters = 4;
opts = statset('Display','final');
[idx, C] = kmeans(data, n_clusters, 'Options', opts, 'Replicates', 5); % K-means
clustering

% Step 3: Calculate the silhouette score
silhouette_values = silhouette(data, idx);

% Calculate the mean silhouette score
mean_silhouette = mean(silhouette_values);
disp(['Average Silhouette Score: ', num2str(mean_silhouette)])

% Plot the silhouette values
figure;
silhouette(data, idx);

% You can also visualize the clusters using a scatter plot
figure;
gscatter(data(:, 1), data(:, 2), idx);
hold on;
plot(C(:, 1), C(:, 2), 'kx', 'MarkerSize', 10, 'LineWidth', 2);
title('Clustered Data');
xlabel('Feature 1');
ylabel('Feature 2');
legend('Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Centroids');
```





K nearest neighbors (KNN)

K-Nearest Neighbors is a simple machine learning algorithm used for classification and regression tasks. It works by finding the K data points in the training dataset that are closest to a new data point (the one you want to classify or predict), and then making predictions based on the majority class or the average value of those K nearest neighbors. Here's a simple explanation of how K-NN works:



For Classification:

1. Training: You start with a dataset containing labeled examples. Each example consists of data points with associated class labels. For instance, you might have a dataset of fruits with features like color, size, and shape, and labels such as "apple," "banana," and "cherry."

2. Choosing K: You choose a value for K, which represents the number of nearest neighbors to consider when making a prediction. A common choice is K = 3 or K = 5, but you can experiment with different values.

3. Predicting a New Data Point: When you want to predict the class label of a new data point (e.g., a new fruit with unknown label), you calculate the distance between this data point and all the data points in your training dataset. Common distance metrics include Euclidean distance and Manhattan distance.

4. Selecting K Nearest Neighbors: You then identify the K data points from your training dataset that are closest to the new data point. These K data points are your "nearest neighbors."

5. Majority Voting: For classification, you look at the class labels of the K nearest neighbors and count how many belong to each class. The predicted class for the new data point is the class that occurs most frequently among its K nearest neighbors. This is known as majority voting.

For Regression:

The K-NN algorithm can also be used for regression tasks, where you predict a numerical value rather than a class label. In this case, instead of taking the majority vote, you calculate the average (mean) of the target values of the K nearest neighbors.

Key points about K-NN:

- ♠ K-NN is non-parametric, meaning it doesn't make any underlying assumptions about the data distribution.
- ♠ The choice of the distance metric and the value of K can impact the results, so you may need to experiment to find the best combination.
- ♠ K-NN is relatively simple and easy to understand, but it can be sensitive to the scale of the features in your dataset, so feature scaling is often necessary.
- ♠ It's a lazy learner, which means it doesn't create a model during training but rather stores the entire dataset for prediction, making it computationally expensive for large datasets.

Confusion Matrix

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.

K-Mean Implementation

```
clear;close all; clear all; clc;
warning off
data=readtable('voice.csv');
DataC=data;
%Checking the Null values
disp(sum(ismissing(data)));
% Labeling our data with (1 for Male and 0 for Female)
k=["male","female"];
l=[1,0];
g=DataC.label;
% generete vector of 0 and 1 values
number=zeros(length(g),1);
for i=1:length(k)
rs=ismember(g,k(i)); number(rs)=l(i);
end
DataC.label=number;
```

`data=readtable('voice.csv');` % This line reads the data from a CSV file named 'voice.csv' and stores it in a table named 'data.' The data is presumably a dataset of voice-related information.

`DataC=data;` % This line creates a copy of the 'data' table and assigns it to 'DataC.' This allows you to work with the copy without modifying the original data.

`%Checking the Null values`
`disp(sum(ismissing(data)));` % This line calculates the sum of missing values in each column of the 'data' table and displays the result. It's used to check how many missing values are in the dataset.

`% Labeling our data with (1 for Male and 0 for Female)`
`k=["male","female"];` % This line creates an array 'k' with two strings, "male" and "female." These strings likely represent the two classes, Male and Female.

`l=[1,0];` % This line creates an array 'l' with two values, 1 and 0. These values correspond to the labels for Male and Female, respectively.

`g=DataC.label;` % This line extracts the 'label' column from the 'DataC' table and assigns it to 'g.'

`% generete vector of 0 and 1 values`
`number=zeros(length(g),1);` % This line initializes a vector 'number' filled with zeros, with a length equal to the number of elements in the 'g' vector.

`for i=1:length(k)` % This is the beginning of a loop that will iterate over the two classes: "male" and "female."

`rs=ismember(g,k(i)); number(rs)=l(i);` % This line checks whether each element in 'g' is equal to the current class 'k(i)' (either "male" or "female"). The result is a logical vector, and 'rs' will be 1 (true) where 'g' is equal to 'k(i)' and 0 (false) otherwise.

`end`
`DataC.label=number;` % This line replaces the 'label' column in the 'DataC' table with the 'number' vector, which contains the new labels for Male and Female