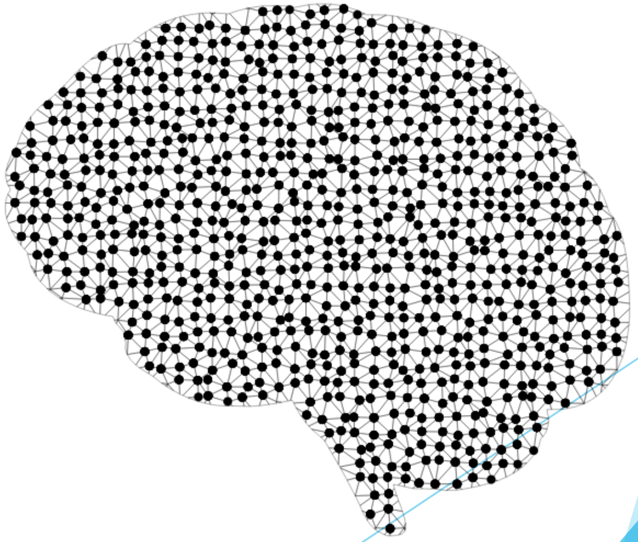
# AI ToolKits in MATLAB

ENGAGED QUALITY INSTRUCTION THROUGH PROFESSIONAL DEVELOPMENT

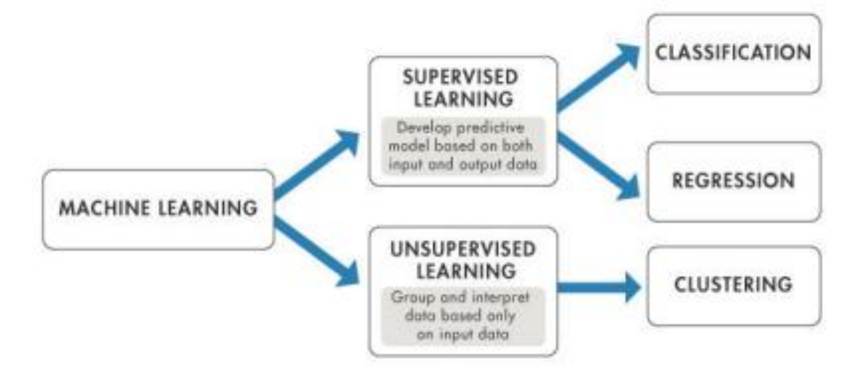
EQuIPD

UF Herbert Wertheim College of Engineering
UNIVERSITY *of* FLORIDA



# So… What is Artificial Intelligence Anyways?

First, we will watch a short 8-minute video that introduces the concept of **artificial intelligence**, please click on the following link to the YouTube website:


**Let's discuss the following question:**

- How does AI, akin to using APIs and web scraping, gather data to suggest content?

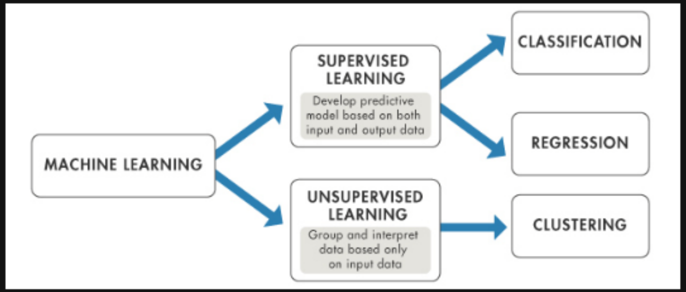
# Exploring AI with MATLAB

**Please copy over the files for Section 05 from the MATLAB Drive**

Table of Contents



## Machine Learning, AI tools, and Analysis



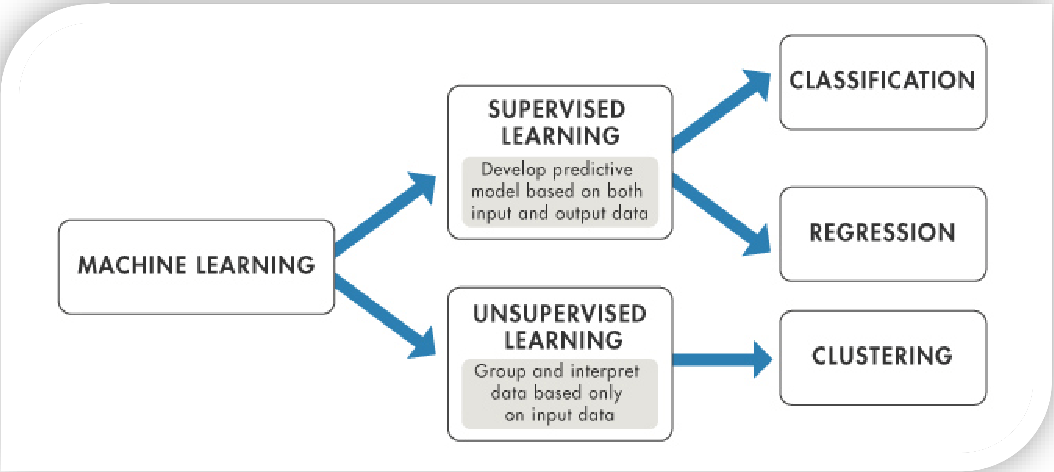

### Part A: Introduction

[The MATLAB files for this section can be found at this link.]()

# Machine Learning, AI tools, and Analysis

## Part A: Introduction

Imagine teaching a computer to learn from examples, just like how you learn from practice problems in math class. That's what machine learning is about, and MATLAB makes it easy to explore. It's like having a virtual lab where you can play with data and teach your computer to recognize patterns or make predictions without telling it exactly what to do.




With MATLAB, you can create cool projects like predicting future stock prices or identifying handwritten digits. It's not just about coding; it's about teaching computers to think a bit like us. So, if you're curious about how computers learn and want to dive into the world of AI, MATLAB is your playground!

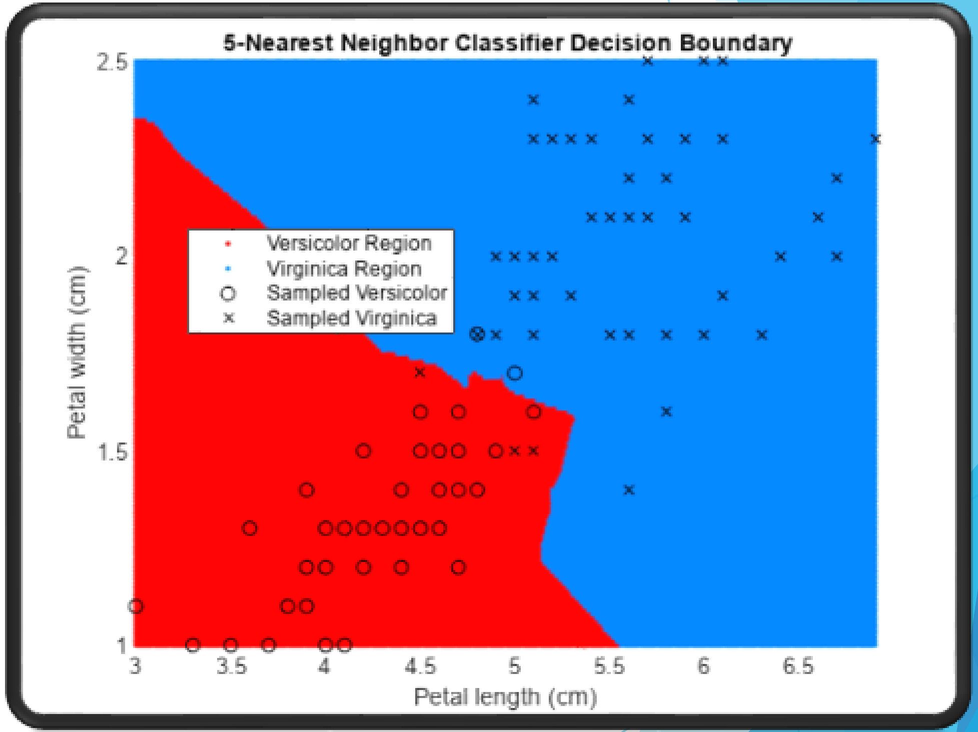
# Machine Learning, AI tools, and Analysis

## Part B: What is a KNN Model?

Imagine you're a music enthusiast building a recommendation system. The K-Nearest Neighbors (KNN) model is like having a group of friends who help you make recommendations based on the countless songs they've listened to.

In the KNN model, when you give it a new data point, it looks at the 'nearest neighbors' – the data points closest to it – and makes a decision based on what those neighbors are like.

It's a straightforward approach that adapts well to various problems like classification, regression, and recommendation systems. Plus, it doesn't require training beforehand; it learns as it goes.



# Machine Learning, AI tools, and Analysis

## Part C: KNN Model Example

All models require data, so first we are going to review how use the readtable() function and dot notation to import data stored in a spreadsheet or text file.

The code on the right shows how to import the data from the spreadsheet myfile.xlsx and store it in a table variable called data.

```
% data = readtable("myfile.xlsx");
```

The first line of code extracts the variable Xdata from the table mytable and stores the result in a new variable named x using dot notation. Similarly, the second line of code extracts the variable Ydata into y.

```
% x = mytable.Xdata;
% y = mytable.Ydata;
```

# Machine Learning, AI tools, and Analysis

## Step 1: Gather & Visualize Data

Alright, now that we all remember how to extract data from files let's start working with the xlsx file RacerStartingPositionFinalPosition.xlsx which contains a table of 56 starting and final positions for 5 drivers from 13 races in the 2022 Formula 1 season. The table has three variables: StartingPosition, FinalPosition, and Driver.

**Try It!**

Plot the extracted features from RacerStartingPositionFinalPosition.xlsx, by using the scatter() function, with StartingPosition on the horizontal axis and FinalPosition on the vertical axis.

```
clc; clear;
% Read the table data from the Excel file "RacerStartingPositionFinalPosition.xlsx"
% and store it in the variable RacerTestData.
%%% YOUR CODE GOES HERE %%%

% Create a scatter plot of the data, with the StartingPosition column on the x-axis
% and the FinalPosition column on the y-axis.
%%% YOUR CODE GOES HERE %%%
```

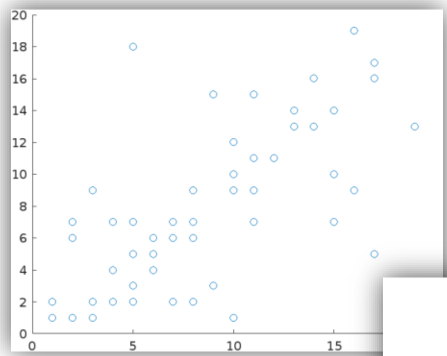
Run

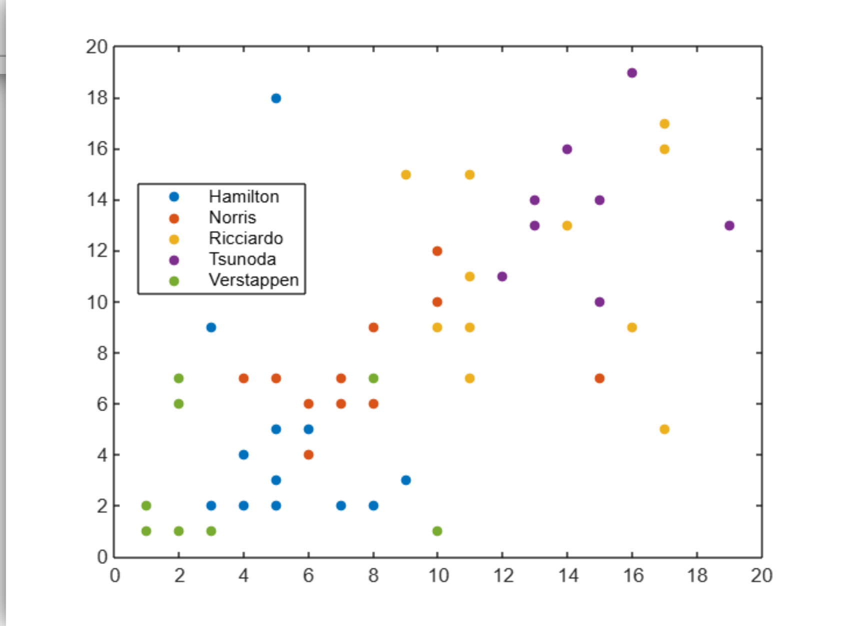
# Machine Learning, AI tools, and Analysis

## Step 1: Gather & Visualize Data

One issue with the current plot is that we can't distinguish the 5 drivers in the data set (Verstappen, Hamilton, Norris, Tsunoda, and Ricciardo ). The gscatter() function makes a grouped scatter plot: a scatter plot where the points are colored according to a grouping variable.

Let's use the gscatter() function to create the same scatter plot as before but grouped by the driver.





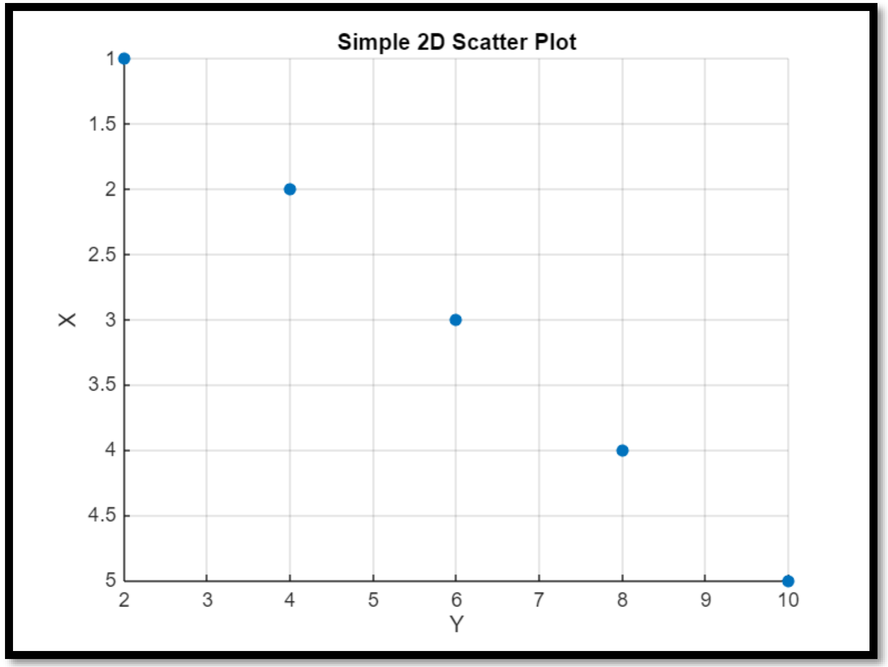

```
% Creates a grouped scatter plot of the data from the RacerTestData table.
% The StartingPosition column is used for the x-axis values,
% the FinalPosition column is used for the y-axis values,
% and the Driver column is used to group the data points by different drivers.
gscatter(RacerTestData.StartingPosition, RacerTestData.FinalPosition, RacerTestData.Driver)
Run
```

The 3rd input tells the function how to group the data points.

# Machine Learning, AI tools, and Analysis

## Step 1: Gather & Visualize Data

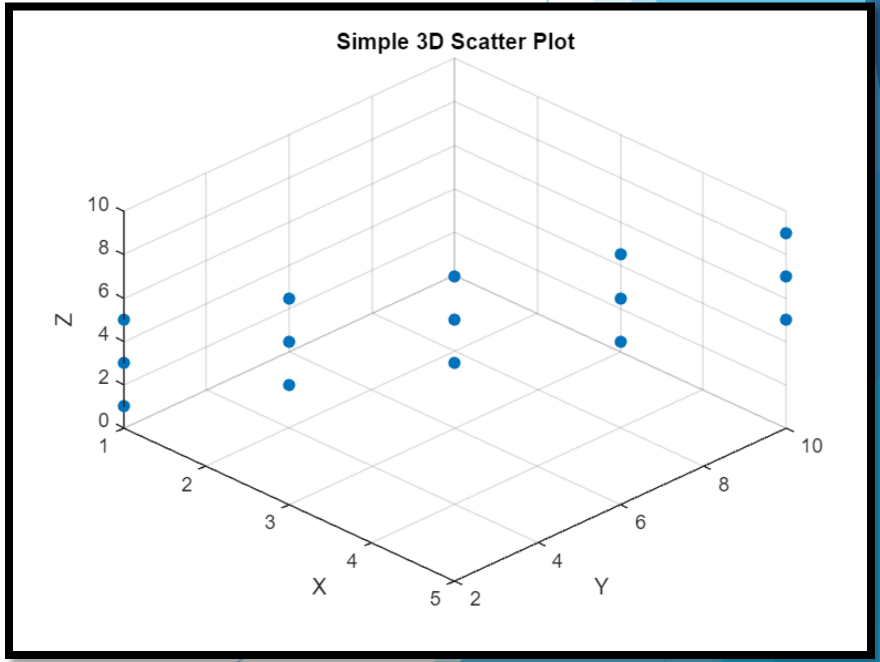
Good job! It appears the new scatter plot worked. However, distinguishing between multiple drivers remains challenging due to overlapping data points. To address the overlapping data, we need to use 3D scatter plot. Unlike 2D plots which only have two axes (x, y), a 3D plot has three axes (x, y, z).




**These images shows two simple scatter plots, which both contain 15 data points.**

**However, overlapping makes it appear that the 2D plot only has 5 data points.**

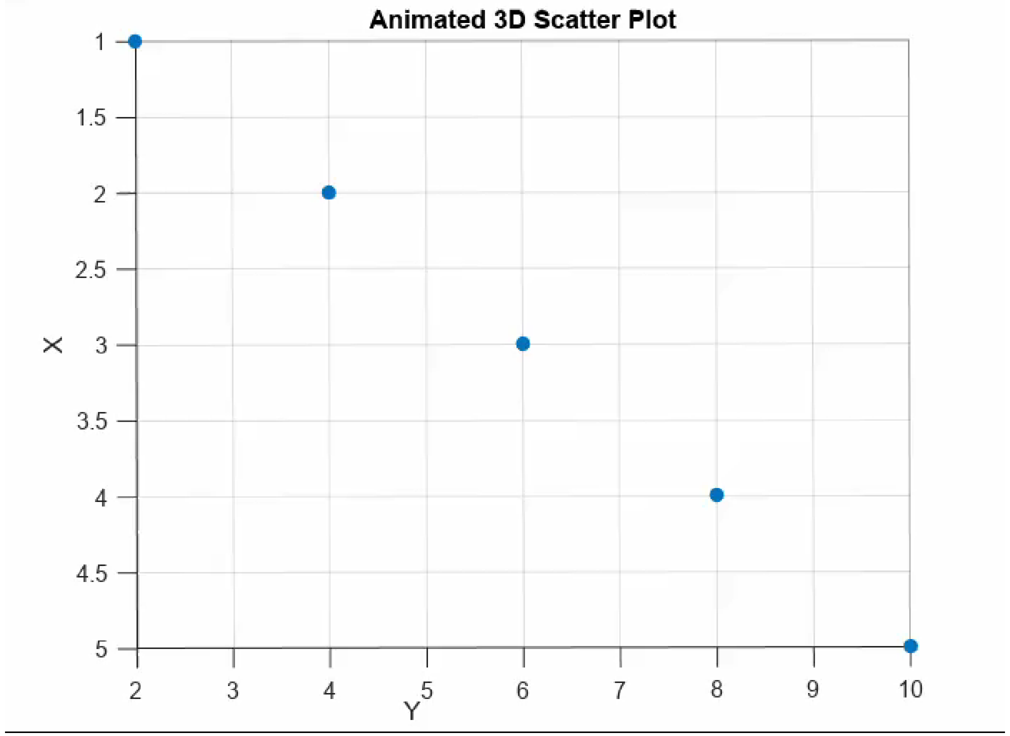
**For this reason, you need a 3D plot to get the full picture.**



# Machine Learning, AI tools, and Analysis

**Step 1: Gather & Visualize Data**

Here is an animation to help you visualize the example on the previous slide!




On the next slide is a step-by-step code and comments to guide you through the creation of a 3D plot for the driver data!

# Machine Learning, AI tools, and Analysis

## Step 1: Gather & Visualize Data

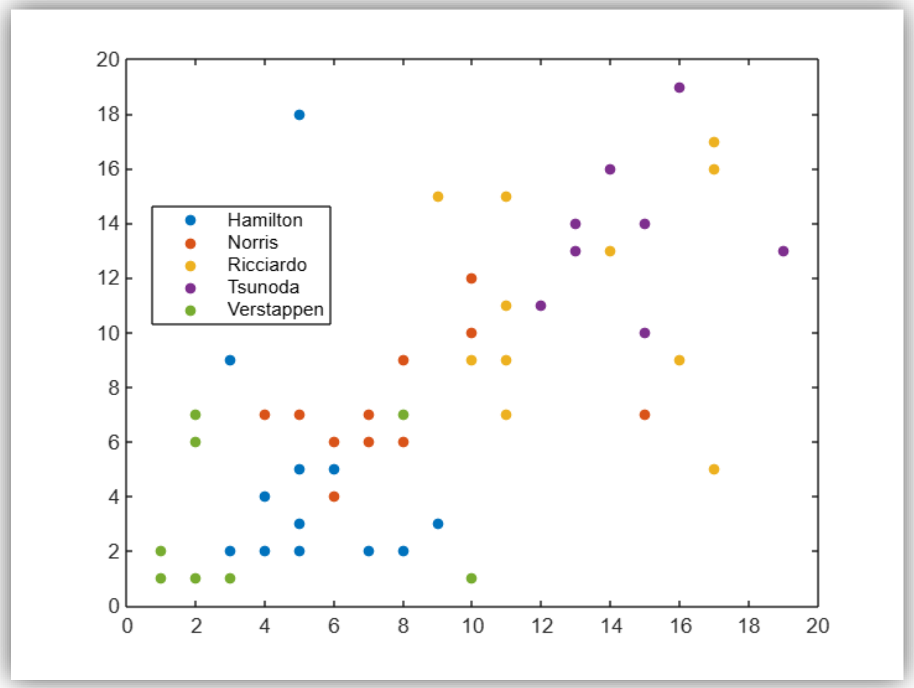
1. Extract and save the driver's names in a categorical data
2. Use scatter3() function to makes a 3D plot with the x-axis as 'Starting Position', y-axis as 'Final Position', and z-axis as 'Drivers'. Group the plot by driver name.
3. Add appropriate title and label all the axes.
4. Adjust the view of the plot
5. Make legend for the plot in the form of a color bar.

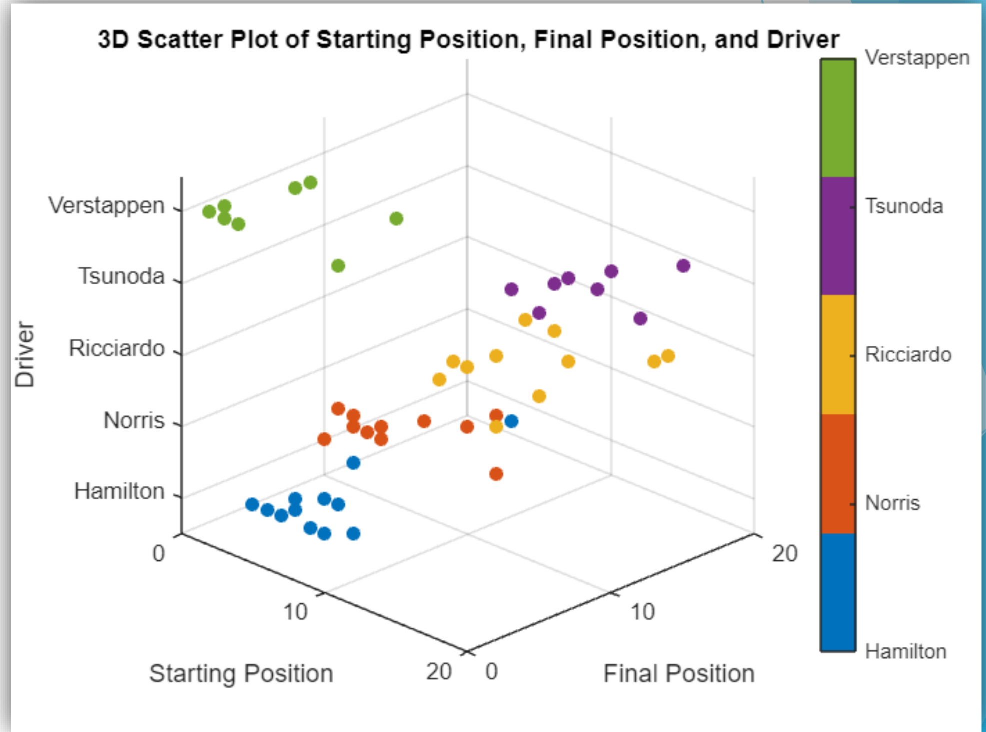
```
RacerTestData = readtable("RacerStartingPositionFinalPosition.xlsx");

% Extract the 'Driver' column from the table RacerTestData as a categorical array
DriverNames = categorical(RacerTestData.Driver);

% Create a 3D scatter plot
figure;

% variables for each axes, Size 36 for markers, color coded by Driver
scatter3(RacerTestData.StartingPosition,RacerTestData.FinalPosition, DriverNames, ...
    36, DriverNames, 'filled');

% Add a title and label the axes
title('3D Scatter Plot of Starting Position, Final Position, and Driver');
xlabel('Starting Position');
ylabel('Final Position');
zlabel('Driver');

% Show the plot
grid on;
view(45, 25) % Adjust the plot angle for better visibility

%Finally let's create a legend for the scatter plot
colormap(lines(length(categories(DriverNames))));
c = colorbar; % Create a colorbar
c.Ticks = 1:length(categories(DriverNames)); % Sets number of tick marks = number of drivers.
c.TickLabels = categories(DriverNames); % Label names of the drivers.
```

# Machine Learning, AI tools, and Analysis

**Step 1: Gather & Visualize Data**

Now we can actually see all of the data clearly! It is important to correctly visualize your data, so that you can fully understand what it represents.

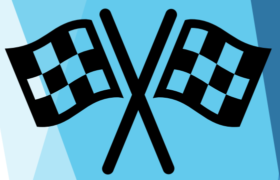


# Machine Learning, AI tools, and Analysis

## Step 2: Fit the KNN Model

Now that we have successfully visualized the data, we can begin to working with the KNN model. You can fit a KNN model by passing a table of data through the fitcknn() function.



```
% mdl = fitcknn(data,"ResponseVariable");
```

The second input is the name of the response variable in the table (the class you want the model to predict). The output is a variable containing the fitted model.

### Now You Try It!

Fit a model to the data stored in RacerTestData by using the fitcknn() function. The known classes are stored in the variable named Driver. Store the resulting model in a variable named knnmodel.

```
% Fit a k-nearest neighbors (KNN) classification model using the data in RacerTestData.
% The model predicts the 'Driver' based on the other variables in the table.
% The resulting model is stored in the variable knnmodel.
%%% YOUR CODE GOES HERE %%%
Run
```

**The class names in the output should be Hamilton, Norris, Ricciardo, Tsunoda, & Verstappen.**

# Machine Learning, AI tools, and Analysis

## Step 2: Fit the KNN Model

The model is now ready to start making predictions using  the predict() function.

```
% predClass = predict(model,newdata)
```

The inputs are the trained model and the observations. The output is a categorical array of predictions for each observation in newdata.

```
% Use the trained KNN model (knnmodel) to predict the driver for a new data point.
% The new data point has a starting position of 12 and a final position of 10.
% The predicted driver is stored in the variable predicted.
predicted = predict(knnmodel, [12, 10])

predicted = 1x1 cell array
    {'Tsunoda'}
```

The code above uses the predict() function along with the trained model knnmodel to determine, which racer is likely to have a starting position of 12 and a final position of 10. Based on the inputs the model predicts that the driver is Tsunoda.

# Machine Learning, AI tools, and Analysis

## Step 2: Fit the KNN Model

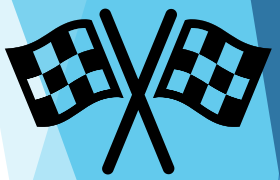
Another feature of the KNN model is that you can specify the value of k in the kNN model by setting the "NumNeighbors" option when calling fitcknn.

```
% mdl =
fitcknn(data,"ResponseVariable", ...
"NumNeighbors",10);
```

Adjusting the number of nearest neighbors the model utilizes can have an impact on the model's prediction, so let's rerun the model to see if the prediction changes.

**Now You Try It!**

Repeat the commands from the previous tasks but use the "NumNeighbors" option to change the number of neighbors in the model to 8.



```
% Fit a k-nearest neighbors (KNN) classification model
% using the data in RacerTestData.
% The number of neighbors to consider for the
% KNN algorithm is set to 8.
%%% YOUR CODE GOES HERE %%%

% Use knnmodel to predict the driver for a new data point.
% The new data point has a starting position of 12 and
% a final position of 10.
%%% YOUR CODE GOES HERE %%%
```

Run

# Machine Learning, AI tools, and Analysis

## Step 3: Evaluate the Model

How good is the kNN model? You can use the model to make predictions, but how accurate are those predictions? Typically, you want to test the model by having it make predictions on observations for which you know the correct classification.

The file VerificationTestData.xlsx contains a table, that has the same variables as RacerTestData, including the known classes for the test observations. You can use the predict function to determine the predictions of the KNN model for the observations in Verificationtestdata, and then compare the predictions to the known classes to see how well the model performs.

```
% Fit a k-nearest neighbors (KNN) classification model
% using the data in RacerTestData.
% The model predicts the 'Driver' based on the other variables in the table.
% The number of neighbors to consider for the KNN algorithm is set to 8.
knnmodel = fitcknn(RacerTestData, "Driver", "NumNeighbors", 8);

% Read the table data from the Excel file 'VerificationTestData.xlsx'
% and store it in the variable VerificationTestData.
VerificationTestData = readtable('VerificationTestData.xlsx');

% Use the trained KNN model (knnmodel) to
% predict the drivers for the new data points
% in VerificationTestData.
predictions = predict(knnmodel, VerificationTestData)
```

```
predictions = 47x1 cell
    'Verstappe…
    'Hamilton'
    'Verstappe…
    'Verstappe…
    'Verstappe…
    'Norris'
    'Verstappe…
    'Verstappe…
    'Verstappe…
    'Hamilton'
       ⋮
```

Array of predictions based on the verification test data

# Machine Learning, AI tools, and Analysis

## Step 3: Evaluate the Model

Now that we have made some predictions using our trained KNN model and we can compare them to the correct results stored in the variable Verificationtestdata.

To compare predictions to the known classes we can use the @isequal operator as shown in the code below.

```
% Compare the predicted drivers with the actual drivers in VerificationTestData.
% cellfun applies the isequal function to each element in the predictions and VerificationTestData.Driver.
% The result is a logical array (true/false) indicating whether each prediction is correct.
iscorrect = cellfun(@isequal, predictions, VerificationTestData.Driver)
```

```
iscorrect = 47x1 logical array
     1
     0
     1
     1
     1
     0
     1
     1
     1
     0
     :
```

1 means the prediction is accurate.

# Machine Learning, AI tools, and Analysis

## Step 3: Evaluate the Model

The logical array shows us which predictions that are correct by marking their position with the number 1. To quantify the accuracy of our kNN model we can divide the number of correct predictions by the total number of predictions.

**Try It!**

Calculate the accuracy of the model and store the result in a variable named accuracy. You can use the sum() function to determine the number of correct predictions and the numel() function to determine the total number of predictions.

```
% Calculate the accuracy of the predictions by dividing the number of correct predictions
% by the total number of predictions made.
% The variable 'iscorrect' contains a logical array indicating whether each prediction is correct.
% 'sum(iscorrect)' calculates the total number of correct predictions.
% 'numel(predictions)' calculates the total number of predictions made.
% The result is stored in the variable 'accuracy'.
%%% YOUR CODE GOES HERE %%%

Run
```

# Machine Learning, AI tools, and Analysis

## Step 3: Evaluate the Model

Rather than accuracy (the proportion of correct predictions), a commonly used metric to evaluate a model is misclassification rate (the proportion of incorrect predictions).

To determine the misclassification rate, we divide the number of wrong predictions by the total number of predictions made.

In the code below, the variable iswrong contains a logical array indicating whether each prediction is incorrect. The line sum(iswrong) calculates the total number of wrong predictions and numel(predictions) calculates the total number of predictions made.

```
% The result is stored in the variable 'misclassrate'.
iswrong = ~cellfun(@isequal, predictions, VerificationTestData.Driver);
misclassrate = sum(iswrong) / numel(predictions)
```

```
misclassrate = 0.3617
```

Run

# Machine Learning, AI tools, and Analysis

## Step 3: Evaluate the Model

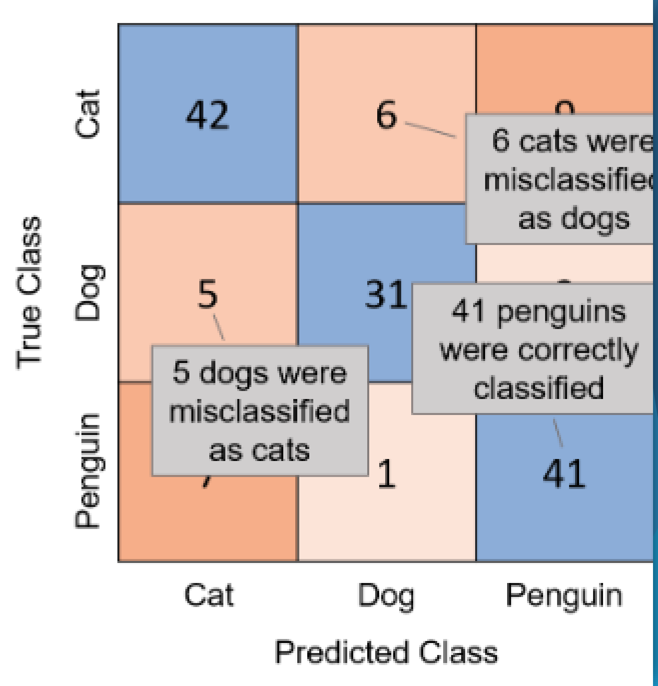
Accuracy and misclassification rate give single values for the overall performance of the model, but it can be useful to see a more detailed breakdown of which classes the model confuses.

A confusion matrix shows the number of observations for each combination of true and predicted class. A confusion matrix is commonly visualized by shading the elements according to their value.

Often the diagonal elements (the correct classifications) are shaded in one color and the other elements (the incorrect classifications) in another color. You can visualize a confusion matrix by using the confusionchart() function.

```
%confusionchart(ytrue,ypred);
```

Where ytrue is a vector of the known classes and ypred is a vector of the predicted classes.



# Machine Learning, AI tools, and Analysis

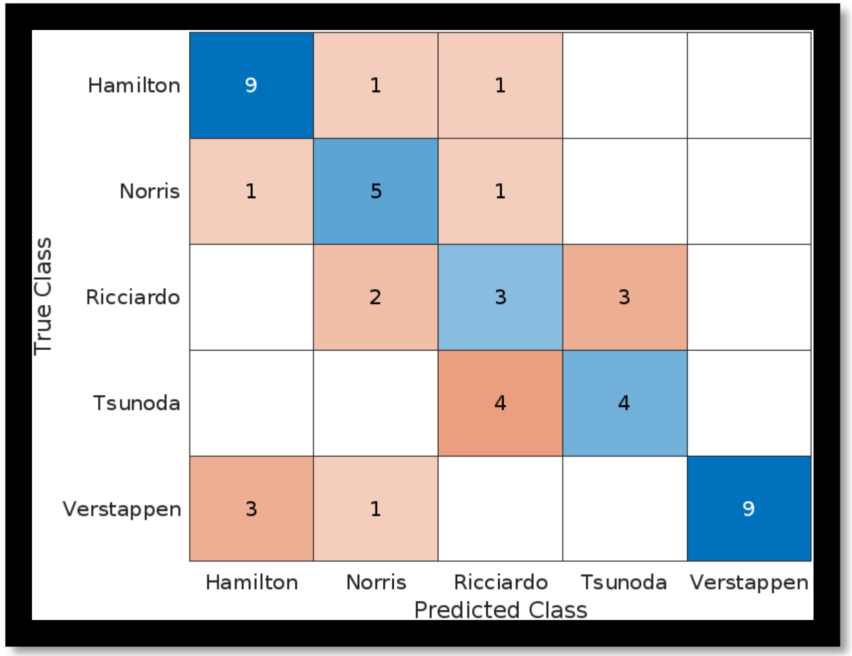
## Step 3: Evaluate the Model

**Now You Try It!**

Compare predictions to the known labels (stored in the variable Driver in the table VerficationTestData) by using the confusionchart() function.

```
% Generate a confusion chart to visualize the performance of the classifier.
% The confusion chart compares the actual drivers (ground truth) from VerificationTestData
% with the predicted drivers from the 'predictions' variable.
%%% YOUR CODE GOES HERE %%%
```

Run




**The resulting confusion matrix should look like the one on the left.**

# Train A KNN Model

**Please copy over the files for Section 05 from the MATLAB Drive, the Part 2 live script should be in the same folder.**

## Train A kNN Model

**Table of Contents**



### Problem 1

In this task, you will utilize the Spotify API to gather data on songs by Kendrick Lamar and Sabrina Carpenter. The API will provide essential features such as song duration and popularity score for each track. **Please do not delete or modify the first or third text box as they import crucial data from the Spotify API and execute a function to save it in an Excel file**. Once the data is collected, your objective is to build a machine learning model that predicts whether a given song belongs to Kendrick Lamar or Sabrina Carpenter based on its duration and popularity. To achieve this, you will select song duration and popularity score as the features for classification, considering their potential to capture distinct characteristics of each artist's songs.

You will train a machine learning model using the training data, employing algorithms such as logistic regression, decision trees, or random forests (Hint: Use the KNN model from the lesson). The model will utilize song duration and popularity score as input features to make predictions. Once trained, you will evaluate the model's performance on the testing data, employing metrics such as accuracy and precision to assess its effectiveness (remember to include a confusion matrix).

#### Step 1: Access Spotify API

[The MATLAB files for this section can be found at this link.]()

# Train A KNN Model

**Practice Problem**

Utilize the Spotify API to gather data on songs by Kendrick Lamar and Sabrina Carpenter. The API will provide essential features such as song duration and popularity score for each track.

**Please do not delete or modify the first or third text box in the code as they import crucial data from the Spotify API and execute a function to save it in an Excel file.**

Once the data is collected, build a machine learning model that predicts whether a given song belongs to Kendrick Lamar or Sabrina Carpenter based on its duration and popularity.To achieve this, you will select song duration and popularity score as the features for classification.

Lastly, you will evaluate the model's performance on the testing data, employing metrics such as accuracy and precision to assess its effectiveness (remember to include a confusion matrix).

# Train A KNN Model

## Step 1: Access Spotify API

Run the first segment after inputting your Client ID and Client Secret from your **Spotify Developer Account**. This code will retrieve the data you need using the custom function fetchAlbumDetailsAndWriteToExcel().

```
clc;clear;

% Set your client ID and client secret
clientID = 'your ID goes here'; % Replace with your actual client ID
clientSecret = 'your Secret goes here'; % Replace with your actual client secret

% Create the URL for token retrieval
url = 'https://accounts.spotify.com/api/token';

% Encode the client ID and client secret in base64
authString = matlab.net.base64encode([clientID ':' clientSecret]);

% Set the options for the HTTP request
options = weboptions('RequestMethod', 'post', 'MediaType', 'application/x-www-form-urlencoded', ...
    'HeaderFields', {'Authorization', ['Basic ' authString]});
```

```
% Define album sets
albumIDsSet1 = {'6PBZN8cbwkqm1ERj2BGXJ1', '7iOAJaGBmk67o337zaqt0R'};  % Replace with actual album IDs
albumIDsSet2 = {'79ONNoS4M9tfIA1mYLBYVX', '5kDmlA2g9Y1YCbNo2Ufxlz'};  % Replace with actual album IDs

% Fetch album details and write to separate Excel files
fetchAlbumDetailsAndWriteToExcel(albumIDsSet1, 'SongTestData.xlsx', Token);
fetchAlbumDetailsAndWriteToExcel(albumIDsSet2, 'VerificationTestData2.xlsx', Token);
```

Run

# Train A KNN Model

**Step 2: Read & Plot the Data**

Now you need to read and plot the data stored in the file "SongTestData.xlsx".

**Where you need to add your code:**

```
% Read the data from the Excel file into a table
%%% Your Code Goes Here %%%

% Extract durations and popularity data from the table
%%% Your Code Goes Here %%%
%%% Your Code Goes Here %%%

% Plot the durations vs. popularity data
%%% Your Code Goes Here %%%

% Scatter plot of durations vs. popularity
%%% Your Code Goes Here %%%

% Grouped scatter plot of durations vs. popularity, color-coded by artists
%%% Your Code Goes Here %%%
```

**Hint: Name you data variable SongTestData**

**Hint: Review Slide 6**

**Hint: Review Slide 8 & group using SongTestData.artists**

# Train A KNN Model

## Step 3: Fit the Model

Now fit the KNN model classification model and make some predictions.
After making predictions evaluate the accuracy.

**Hint: Review Slide 15 and set the "NumNeighbors" to 8.**
**For your prediction use the duration 20000 & popularity of 40.**

**Hint: Read the new verification data and make new predictions.**

**Hint: Review slide 17 and slide 18**

**Hint: Review slide 19**

```
% Fit a k-nearest neighbors (KNN) classification model using the data
% and save in the variable knnmodel.
% Predict the artist based on durations and popularity and store in
% the variable predicted.
%%% Your Code Goes Here %%%
%%% Your Code Goes Here %%%

% Read the verification data from the Excel file into a table called
% VerificationTestData2 from the file 'VerificationTestData2.xlsx'
%%% Your Code Goes Here %%%

% Use the trained KNN model to predict the artists for the verification data
% and name the variable predictions
%%% Your Code Goes Here %%%

% Check if the predictions are correct and calculate accuracy
%%% Your Code Goes Here %%%
%%% Your Code Goes Here %%%

% Calculate misclassification rate
%%% Your Code Goes Here %%%
%%% Your Code Goes Here %%%

% Generate a confusion chart to visualize the performance of the classifier
%%% Your Code Goes Here %%%
```

Run

# Ethical and Technological Implications of AI

## What are the Impacts of Machine Learning?

**Watch the following videos before having a discussion:**

## Let’s discuss the following questions:

1. How might the use of KNN and CNN models in decision-making processes raise ethical concerns, particularly in areas like hiring, lending, or law enforcement?

2. In what ways have KNN and CNN models revolutionized industries such as healthcare, finance, or transportation? What are the potential benefits and drawbacks of these technological advancements?

3. How do KNN and CNN algorithms contribute to the phenomenon of filter bubbles and echo chambers on social media platforms? What implications does this have for societal discourse and polarization?

4. What do you envision as the future trajectory of AI research and development, particularly in the context of KNN and CNN models? How might these technologies evolve to address emerging challenges and opportunities?

Link: What are Neural Networks?


Link : What are KNNs?


Link: What are CNNs?

# EXTRA Practice/Reference Material: Google Net Model

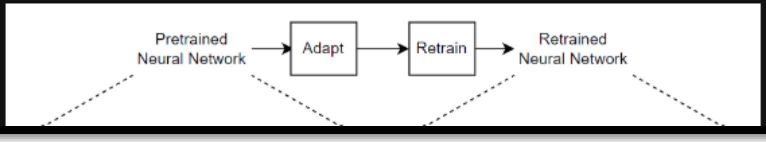
**Please copy over the files for Section 05 from the MATLAB Drive, the “Google Net Model” live script should be in the same folder.**

## Google Net Model

**Table of Contents**



### Part A: What is a Google Net Model




[The MATLAB files for this section can be found at this link.]