LESSON 2: INTRODUCTION TO MACHINE LEARNING

MODULE 1: Lesson Overview

We'll cover the following modules in this lesson:

- What machine learning is and why it's so important in today's world
- The historical context of machine learning
- The data science process
- The types of data that machine learning deals with
- The two main perspectives in ML: the *statistical* perspective and the *computer science* perspective
- The essential tools needed for designing and training machine learning models
- The basics of Azure ML
- The distinction between models and algorithms
- The basics of a linear regression model
- The distinction between parametric vs. non-parametric functions
- The distinction between classical machine learning vs. deep learning
- The main approaches to machine learning
- The trade-offs that come up when making decisions about how to design and training machine learning models

MODULE 2: What is Machine Learning?

- **Machine Learning**
  A data science technique used to extract patterns from data, allowing computers to identify related data, and forecast future outcomes, behaviours, and trends.

- **Traditional Programming Paradigm**

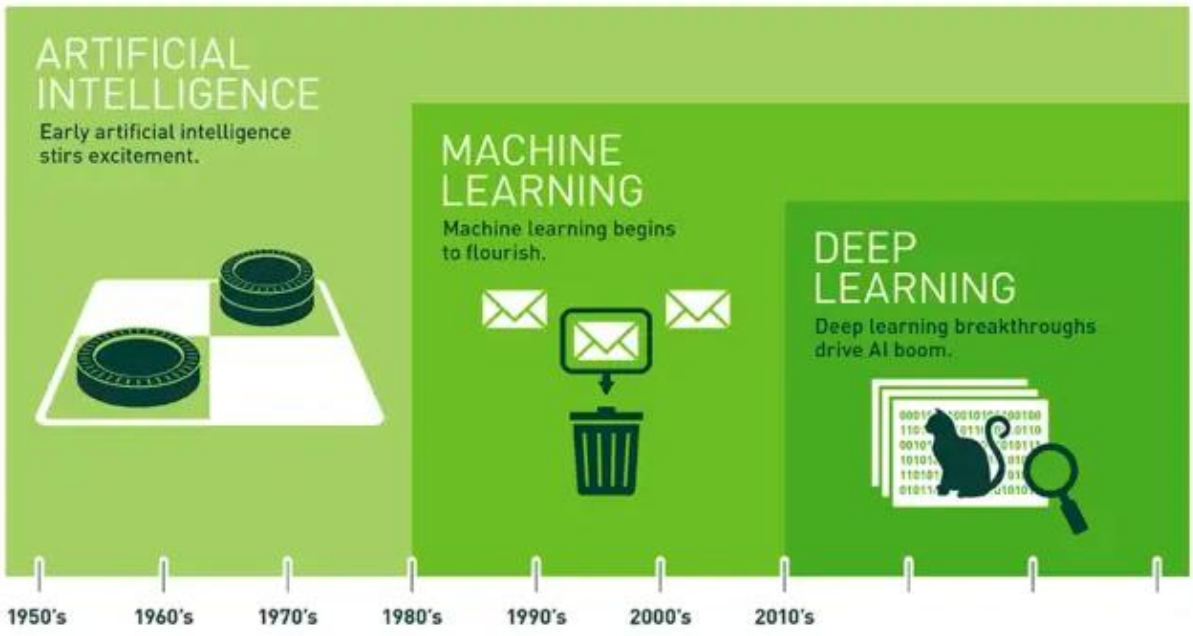
- **Machine Learning Paradigm**

- Machine Learning uses historical data to generate rules that we have not thought of.
- Machine Learning is best suited for tasks like pattern recognition, anomaly detection, time series forecasting and recommendation systems.

MODULE 3: Applications of Machine Learning

- Machine Learning/ Deep Learning/ Reinforcement Learning
    - Natural Language Processing (NLP)
        - Text: summarization, topic detection, similarity, search
        - Speech: speech-to-text, text-to-speech, translation
    - Computer Vision (CV)
        - Self-driving cars
        - Image classification
        - Object detection
        - Object identification
        - LIDAR and Visible Spectrum
    - Analytics
        - Regression
        - Classification
        - Forecasting
        - Clustering
    - Decision Making
        - Sequence decision making problems
        - Recommenders
- Examples of Machine Learning
    - Automating the recognising the disease.
        - Google has trained a deep learning model to detect breast cancer
        - Stanford researchers have used deep learning models to diagnose skin cancer
    - Recommend next best actions for individual care plans using patient's digital health footprint.
        - EMRs (Electronic Medical Records) and EHRs (Electronic Health Records)
        - IBM Watson Oncology
    - Enabling real-time, personalized and interactive banking experience with chat bots. This allows resolving simple issues without the need of human intervention.
        - https://www.drift.com/learn/chatbot/ai-chatbots/
    - Identify next best action for the customer (ex: showing relevant deals).
        - Sentiment analysis
    - Capture, prioritise and route service requests to correct employee to improve response times (ex: feedback mails received from the customers can be forwarded to the concerned department by looking at the content of the mail.)
        - Introduction to Ticket Routing using AI https://monkeylearn.com/blog/ticket-routing/

MODULE 4: HISTORY OF MACHINE LEARNING



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

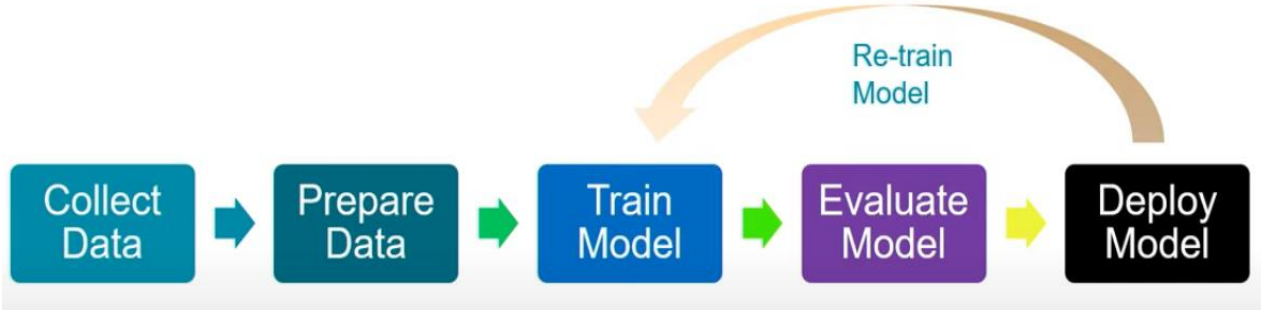
- **Artificial Intelligence**: A broad term that refers to computers thinking more like humans.
- **Machine Learning**: A subcategory of artificial intelligence that involves learning from data without being explicitly programmed.
- **Deep Learning**: A subcategory of machine learning that uses a layered neural-network architecture originally inspired by the human brain.

- Artificial Neural Network
  - A class of Machine Learning algorithms inspired by the functioning of brain.
  - Development was stagnant because of compute challenges.
  - Research & development was boosted with the emergence of GPU in 2000's and 2010's.
- Further readings:
  What's the Difference Between Artificial Intelligence, Machine Learning and Deep Learning? by Michael Copeland at NVIDIA

MODULE 5: THE DATA SCIENCE PROCESS

- Data is being generated at a very high rate on a very large scale. Most of the generated data remains unused.
- Big Data is a term which is used to define the data which cannot be processed locally using traditional methods.
- To process the large amount of data, new concepts like Cloud Computing, Distributed Processing have emerged.

- Today, companies are making every effort to gain insights from the data in order to improve their profitability.
- Big Data has the following 4 main characteristics:
    - Volume
    - Variety
    - Velocity
    - Veracity
- However, this huge amount of raw data cannot be directly used to derive insights or train ML models because of issues like missing values, noise in the data, unsupported format, etc.
- In order to derive any meaningful insights or feed this data to an ML model, this data first needs to be cleaned and processed.
- Today, the ability to combine large, disparate data sets into a format more appropriate for analysis is an increasingly crucial skill.
- The data science process typically starts with collecting and preparing the data before moving on to training, evaluating, and deploying a model.
- Below are the steps involved in a standard data science process:
    - **Collect Data**: This step involves collecting data from different sources like mobile devices, IoT devices, sensors, software, etc. A developer may have to write queries and code to extract data from databases and webpages.
    - **Prepare Data**: This step involves cleaning the data and converting it into a desired format. This step involves activities like handling missing values, noisy data, creating new features, etc. A developer may have to write code to remove noisy data, handle missing values and perform data visualization.
    - **Train Model**: This step involves deciding an algorithm, splitting our data into train, validation and test sets, and training a model. A developer may have to write code to create and train the model.
    - **Evaluate Model**: This step involves evaluation of the performance of our model using different metrics like accuracy, loss, speed, etc.
    - **Deploy Model**: Once you're satisfied with the performance of your model, you can deploy your model using different techniques to derive useful insights and outputs.
    - **Re-train Model**: This is an iterative step which involves training the model on fresh data at regular intervals to make sure the performance of your model is in sync with the changing data environment.



MODULE 6: COMMON DATA TYPES

- Numerical
- Time-Series (numeric data, but in specific order)
- Categorical (represents different categories in real life)
- Text
- Image

All data in machine learning eventually ends up being numerical, regardless of whether it is numerical in its original form, so it can be processed by machine learning algorithms.

MODULE 7: TABULAR DATA

- This is the most common type of data encountered in Machine Learning,
- In tabular data, typically each cell describes a single value, each row describes a single item, while each column describes different properties of the item.
- A **vector** is simply an array of numbers, such as `(1, 2, 3)`—or a nested array that contains other arrays of numbers, such as `(1, 2, (1, 2, 3))`
- Khan Academy: Introduction to Linear Algebra
  https://www.khanacademy.org/math/linear-algebra
- Linear Algebra Refresher Course
  https://www.udacity.com/course/linear-algebra-refresher-course--ud953
- All non-numerical data types (such as images, text, and categories) must eventually be represented as numbers.

MODULE 8: SCALING DATA

- Scaling the data means transforming it in way that it fits within some range or scale, like 0-100 or 0-1.
- Methods of scaling data:
  - Standardization
    - Scales the data to have Mean = 0 and Variance = 1.
    - Scaling is done using the formula: **(x-Mean)/Variance**
  - Normalization
    - Scales the data in the range 0-1.
    - Scaling is done using the formula: $\mathbf{(x - x_{min}) / (x_{max} - x_{min})}$

MODULE 9: ENCODING CATEGORICAL DATA

- Machine Learning algorithms required data in the numerical format. Hence, it becomes important to convert our data in the required format.
- Ex: Converting categorical data (male, female, others) into numerical data.
- There are two common approaches for encoding categorical data:
    - Ordinal encoding
    - One hot encoding

  Let's take a look at them one by one.
- **Ordinal Encoding**
    - Converts the categories into integer code ranging from 0 to (number of categories – 1).
    - Ex:

Colour	**Encoding**
Green	0
Blue	1
Red	2

    - This method has one major drawback that it assumes that there is a particular order in the categories, like the colour Green is more important than Blue and Blue is more important than Red, or vice-versa. This may or may not be the case in reality.
    - In order to overcome this drawback, lets take a look at One hot Encoding.
- **One hot Encoding**
    - A new column is added for each distinct category in the data.
    - If a row belongs to a particular category, the value of column corresponding to that category will me marked as 1, while all other columns corresponding to all other categories will be marked as 0.
    - Ex:

Name	**Gender**
Rayan	M
Jessy	F
Liz	F

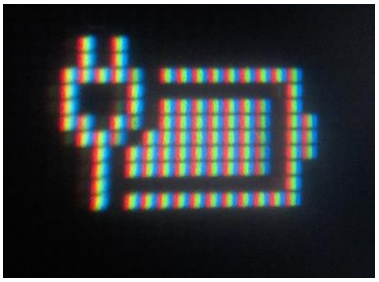
    - The above column can be converted into the following format after applying one hot encoding:

Name	**Male**	**Female**	**Others**
Rayan	1	0	0
Jessy	0	1	0
Liz	0	1	0

    - This approach gets rid of the drawback created by the ordinal encoding.
    - However, this approach gives rise to another problem of having large number of columns in case you have more categories.

MODULE 10: IMAGE DATA

- Zooming in on the below image, you'll find that this image is made of small square titles called a "Pixel".



- Digitally, images are represented in form of pixels. A pixel is a smallest unit of an image.
- Images are described in terms of total number of pixels i.e.,
  Height x Width x Number of channels
- In machine learning, square images are most commonly used.
- The colour of each pixel can be represented in different format:
  - **Greyscale**: Each pixel is represented by a single value ranging between 0-255. Here the number 0 represents black and 255 represents white colour.
  - **Coloured**: Each pixel is represented by a vector of 3 number, where each number ranges between 0-255.
- The number of channels required to represent a colour is called **colour depth** or simply, **depth**.
  - In case of a **greyscale** image, the **colour depth** is **1**.
  - While in the case of an RGB image, the **colour depth** is **3.**
- We can fully encode an image numerically by using a vector with three dimensions. The size of the vector required for any given image would be the **height * width * depth** of that image.
- We may want to perform other processing operations on an image after encoding it:
  - Normalization: Subtracting the mean pixel value in a channel from each pixel value in that channel.
  - Rotation
  - Cropping
  - Resizing
  - Denoising
  - Centring

MODULE 11: TEXT DATA (DAY 1/50)

- Text is another form of data that is non-numerical initially and must be processed before feeding it to the machine learning algorithms.
- **Normalization**
  - Normalization means converting a piece of text into a canonical/official form.
  - It is often seen that many different words used in text mean the same thing:
    - Ex: the verb **to be** may show up as **am, is, are**.
  - Also, many words have 2 different spellings.
    - Ex: **behavior** and **behaviour**
  - Thus, it becomes necessary to perform normalization to resolve all the above-mentioned inconsistencies.
- **Lemmatization**
  - **Lemma** is the dictionary form of a word.
  - Lemmatization is a form of Normalization which involves reducing multiple inflections to the dictionary form of the word.
  - This can be understood with the following example:

Original Word	Lemmatized Word
am	be
is	be
are	be

- **Removing Stop words**
  - Stop words are high-frequency words which add little meaning during analysis.
  - For example, after removing the stop words, the phrase **how to reach the Mount Everest** is reduced to **reach Mount Everest**, which still conveys pretty much the same meaning.
- **Tokenization**
  - Tokenization is a very common practice in text processing where we split each string into smaller parts, called **tokens**.
  - The below examples demonstrate tokenization:

Original String	Tokenized text
I like mangoes	[I, like, mangoes]
The train left the station	[The, train, left, the, station]

- **Vectorization**
  - After the normalization of text, next we convert it into a numerical vector, and the process is called vectorization.
  - There are many different methods of vectorization, but the 2 most common ones are as follows:
    - Term-Frequency Inverse Document Frequency (TF-IDF) https://en.wikipedia.org/wiki/Tf-idf
    - Word Embedding
      Word2Vec: https://en.wikipedia.org/wiki/Word2vec
      Global Vectors (GloVe): https://nlp.stanford.edu/pubs/glove.pdf

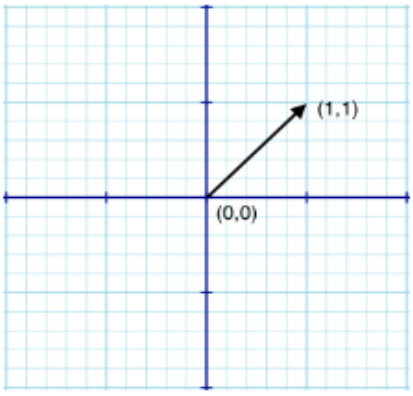
- **TF-IDF**
  - This approach gives lesser importance to words (like **is, the, am, will**) which are most common in the document and represent little information.
  - Words which contain more information and appear less frequently are given more importance using this approach.
  - This approach can be understood with the example below:

python	**is**	**a**	**general**	**purpose**	**programming**	**language**
0.32	0.0	0.0	0.10	0.13	0.45	0.25

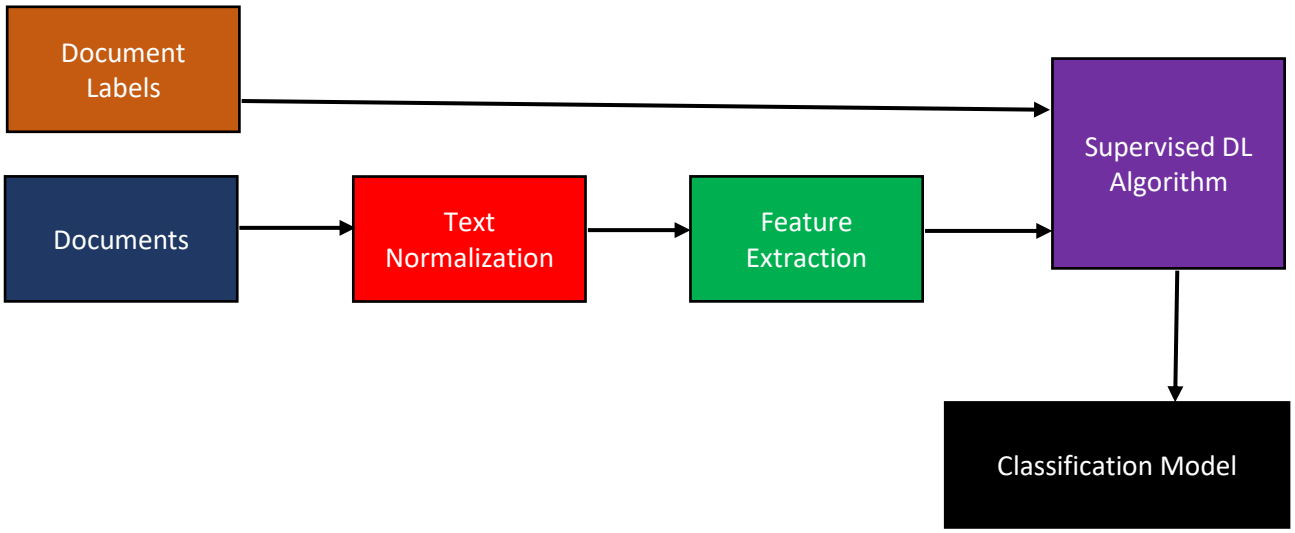
  - The above table contains the phrase **python is a general-purpose programming**
  - **language** split up into tokens and given weightage depending upon their importance in the document.
  - Taking a chunk of above text would result in the following table:

	python	general	purpose	programming	language
[python, general]	0.32	0.10	0.0.	0.0	0.0
[purpose, programming]	0.0	0.0	0.13	0.45	0.0
Language	0.0	0.0	0.0	0.0	0.25

  - It can be noticed that the words **'is'** and **'a'** are not a part of the above table since they do not contain any important information.
- **Feature Extraction**
  - The text in the previous example can be represented in form of a single vector containing 5 elements (since there are 5 total words after removing stop words).
    [python, general] = [0.32, 0.10, 0, 0, 0]
    [purpose, programming] = [0.0, 0.0, 0.13, 0.45, 0.0]
    [language] = [0.0, 0.0, 0.0, 0.0, 0.25]
  - Vectors of length **n** can be represented in **n-dimensional** space.
  - For example, a vector (1,1) can be viewed as a line starting from (0,0) and going till (1,1).



- How close two vectors are can be calculated using vector distance.
- When two vectors are close to each other i.e., they have a small vector distance, it can be understood that those two vectors either have similar meaning or are closely related to each other.
- Below is the end-to-end pipeline for classification model using text data.



MODULE 12: TWO PERSPECTIVES ON ML

- Machine Learning can be described from two different perspectives.
  - Computer Science
  - Statistical
- From the **Computer Science** perspective, we may state that we're using input features to create a program that can generate the desired output.
- From a **Statistical perspective**, we may state that we're trying to find a function which that can generate the values of the dependent variables given the values of the independent variables.
- We can map the two perspectives in the following manner:

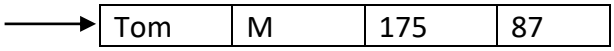
Computer Science	**Statistics**
Program	Function
Input	Independent variables
Output	Dependent variables

MODULE 13: THE COMPUTER SCIENCE PERSPECTIVE

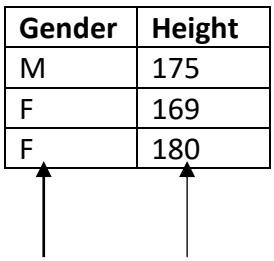
- Data can be present in form of rows and columns in a spreadsheet, as displayed below.

Name	Gender	Height	Weight
Tom	M	175	87
Vic	F	169	65
Laura	F	180	76

- From the Computer Science perspective, each row can be considered as an **entity** or an **observation about an entity**.



- Each column in the spreadsheet can be then considered as an **attribute** or a **feature** of the aforementioned entity.



- A row may also be called an **instance.**
- **INPUT VECTOR**: A group of input variables
- In the computer science terms, we can understand machine learning as:

Output = Program (Input Features)

MODULE 14: THE STATISTICAL PERSPECTIVE

- In the Statistical perspective, the machine learning algorithm is trying to find a hypothetical function, f, such that:

Output Variables = f (Input Variables)

- The input variables are also called the **independent variables** while the output variables are also known as the **dependent variables**. Thus, the above equation can be rewritten as:

Dependent Variables = f (Independent Variables)

- Often the output variable is represented with the alphabet **Y** and the input variable is represented with the alphabet X. So, the above equation is commonly represented with the shorthand as:

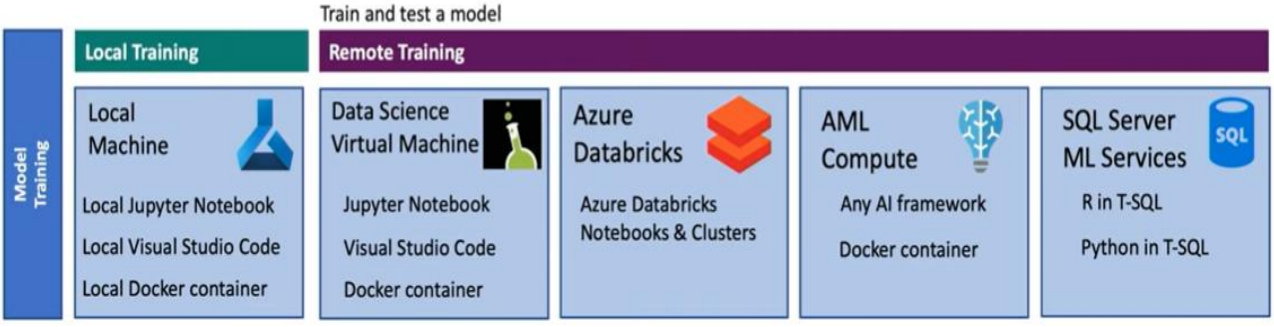
Y = f (X)

MODULE 15: THE TOOLS FOR MACHINE LEARNING

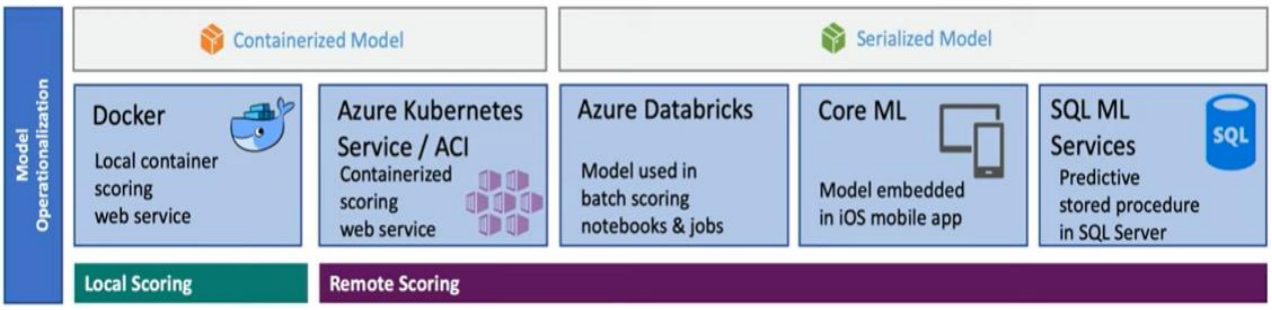
- Let's take a look at some of the most popular libraries and tools which form an integral part of any Machine Learning Ecosystem.

Tool/Library Category	Tool/Library Name	Description
Libraries	Scikit-Learn	Classical ML library
	• TensorFlow • Keras • PyTorch	Deep Learning libraries
Development Environments	• Jupyter Notebooks • Azure Notebooks • Azure Databricks • Visual Studio • Visual Studio Code	Provide interface to build, train and test your model by writing code.
Cloud Services	• Microsoft Azure Machine Learning • AWS • GCP	Service providers which allow you to develop and deploy your ML models on Cloud.

- Microsoft Azure provides environment for both development and deployment (operationalization) of your model.
- Below are the different tools/options provided for training your ML model.



- After training your model, you can also deploy/operationalize your model using different tools/options provided by Microsoft Azure.



MODULE 16: LIBRARIES (DAY 2/50)

- **Core Framework and Tools**
  - **Python**:
    - Python is a general-purpose programming language which is widely used as a preferred language for developing Machine Learning models.
    - Python provides a plethora of extremely useful libraries which can be very easily brought into use for ML-related operations.
  - **Pandas**:
    - Pandas is one of the most popular open-source, high performance, easy-to-use data structures and data analysis tools available in Python programming language.
    - https://pandas.pydata.org/
  - **NumPy**:
    - NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
    - https://numpy.org/
  - **Jupyter:**
    - Jupyter provides interactive and modular **Notebook-type** development environment for Python.
    - https://jupyter.org/
- **Machine Learning & Deep Learning Libraries**
  - **Scikit-Learn**
    - One of the most popular Python libraries for Machine Learning.
    - Can be integrated with SciPy, NumPy and Matplotlib.
    - Provides out-of-the-box implementation of most common ML algorithms like Regression, Clustering, SVM, etc.
    - https://scikit-learn.org/stable/
  - **Apache Spark**
    - Open-source analytics engine which supports cluster-computing.
    - Most often used for large-scale data processing.
    - https://spark.apache.org/
  - **TensorFlow**
    - Open-source Deep Learning library developed by Google Brain.
    - Most popular library for Deep Learning on GitHub.
    - Has libraries for running complex algorithms in browser (TensorFlow.js) and mobile phones (TensorFlow Lite) in real-time.
    - https://www.tensorflow.org/
  - **Keras**
    - Very popular in the ML community as a tool which allows rapid prototyping and development.
    - Provides an Application Programming Interface (API) for the more complex libraries like TensorFlow.
    - https://keras.io/

- **PyTorch**
  - A Deep Learning library developed largely in part by Facebook's AI Research Lab.
  - Provides more beginner-friendly and pythonic implementation of various ML algorithms.
  - Much more popular in the research community.
  - https://pytorch.org/

- **Visualization**
  - **Plotly**
    - Plotly is not a library but a company which provides different visualization libraries in different programming languages like Python, JavaScript, etc.
    - https://plotly.com/
  - **Matplotlib**
    - One of the most popular visualization libraries available in Python.
    - Provides numerous 2D charting and graphing options.
    - https://matplotlib.org/
  - **Seaborn**
    - Python library built specifically for data visualization.
    - Built on top of Matplotlib, but provides a much higher-level interface along with more features to make your visualizations more attractive and informative.
    - https://seaborn.pydata.org/
  - **Bokeh**
    - Interactive visualization library.
    - As opposed to Matplotlib which provide static visualization, Bokeh generates visualizations in HTML and JavaScript.
    - This makes web-based visualization more interactive and attractive.
    - https://bokeh.org/

MODULE 17: CLOUD SERVICES

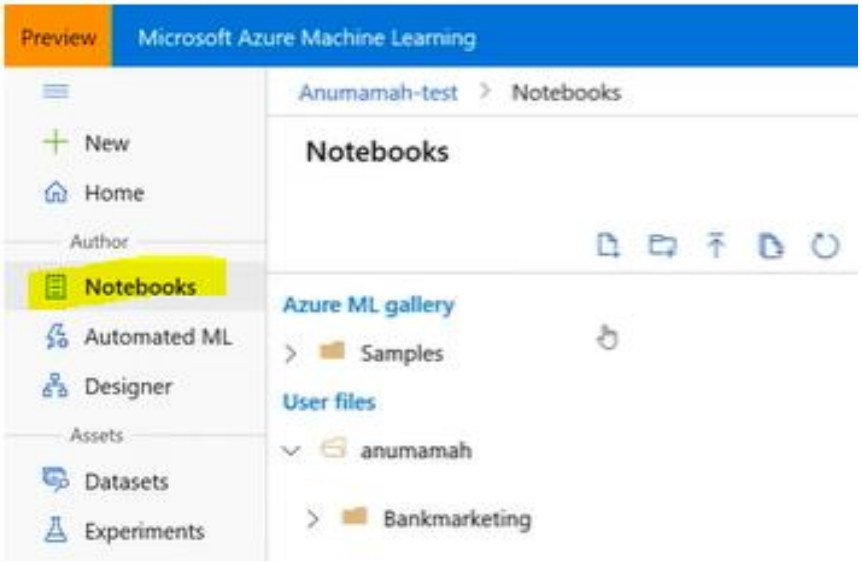
- A typical cloud service for Machine Learning provides support for managing the core assets of a Machine Learning project.
- Some of the common core assets of an ML project are as follows:

Feature	Description
Datasets	Define, version, and monitor datasets used in machine learning runs.
Experiments / Runs	Organize machine learning workloads and keep track of each task executed through the service.
Pipelines	Structured flows of tasks to model complex machine learning flows.
Models	Model registry with support for versioning and deployment to production.
Endpoints	Expose real-time endpoints for scoring as well as pipelines for advanced automation.

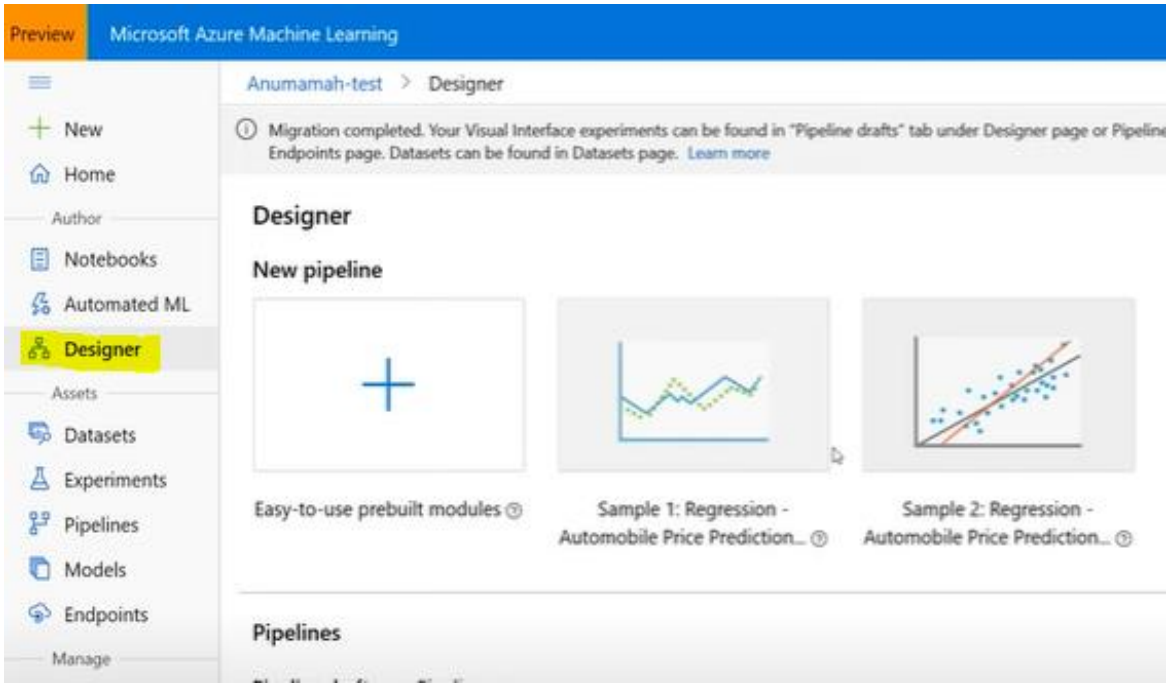
- Apart from this, a cloud service also needs to provide support for managing different resources required to run the machine learning tasks.

Feature	Description
Compute	Manage compute resources used by machine learning tasks.
Environments	Templates for standardized environments used to create compute resources.
Datastores	Data sources connected to the service environment (e.g. blob stores, file shares, Data Lake stores, databases).

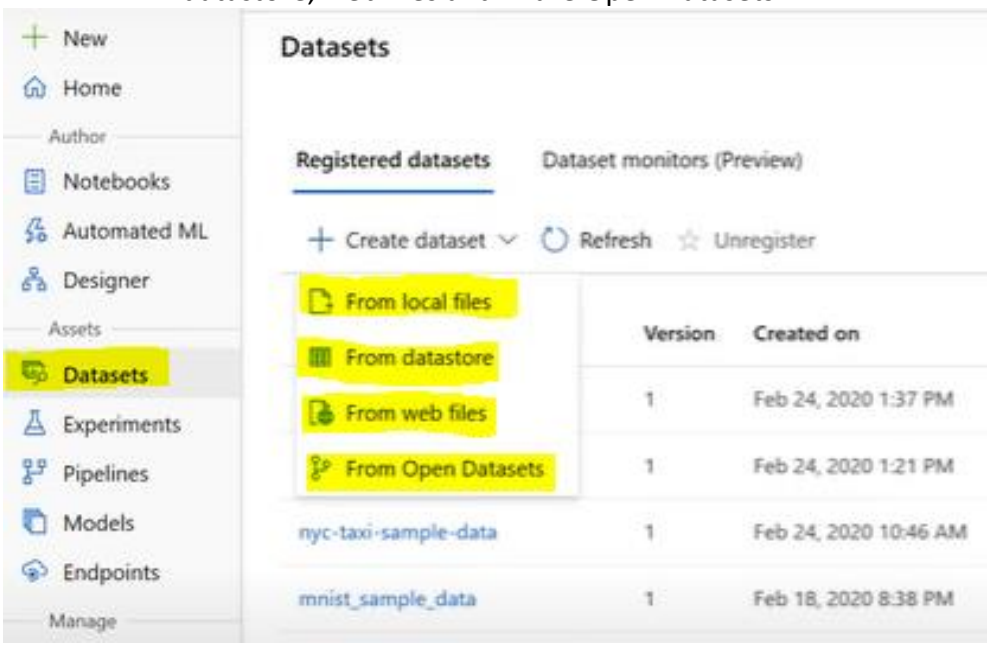
- **Introduction to Azure Machine Learning**
  Below are different sections in the Azure Machine Learning Studio
  - **Workspace**
    - It is a top-level resource for managing all your artifacts that you create while working on Azure ML.
    - It contains several sections like: quick links, recent items and tutorials.
  - **Notebooks**
    - This section allows you to create new notebooks in which you can write some code and develop you ML model.
    - This section contains some sample notebooks for your reference.



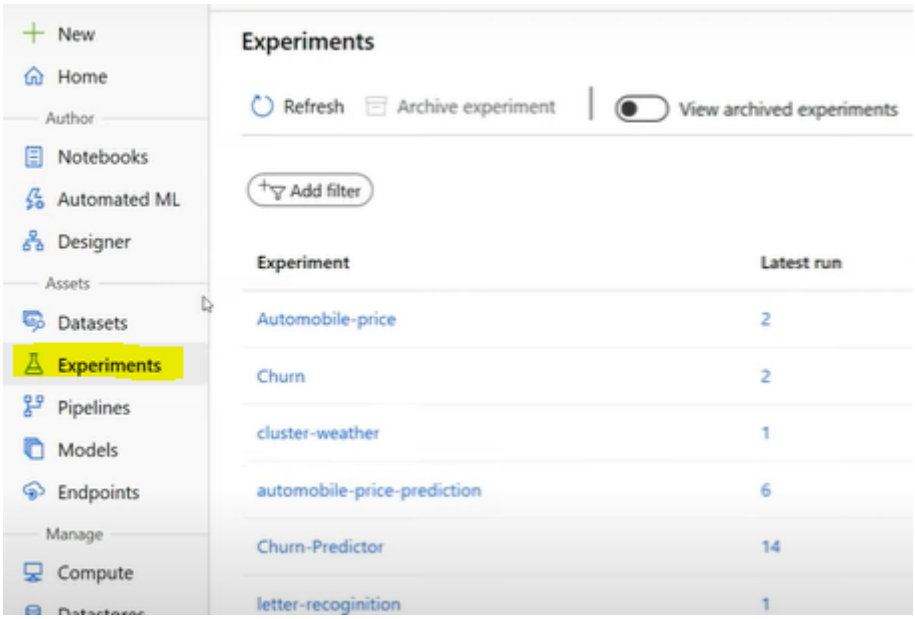
  - **Automated ML**
    - Auto-ML continuously iterates over different parameters of an ML algorithm to provide you with the best performing model according to the success metric defined by you.
    - In this section, you can create new AutoML models or view existing ones.
  - **Designer**
    - It is a drag-and-drop tool which allows you to build end-to-end ML pipelines without need of any code.
    - You can create new pipelines or view existing pipelines in this section.



- **Datasets**
  - This section allows you to create and manage datasets in your ML project.
  - You get multiple ways to create your dataset like: from local files, datastore, web files and Azure Open Datasets.



- **Experiments**
  - This section help you organize your runs.
  - Every experiment that you create shows up in this section.
  - When you submit a new run, you must either create a new experiment for that run or associate it with an existing experiment.
  - You can view all the details regarding your runs in this section.



- **Models**
  - A model is produced as a result of a run in Azure ML.
  - You can also train your model outside Azure ML and import it in this section to be used later on.
- **Endpoints**
  - This section provides real-time endpoints for scoring and pipelines for advanced automation.
  - Every endpoint that is deployed in the workspace shows up in this section.
- **Computer**
  - A designated compute resource where you run the training script or host the service deployment.
- **Datastores**
  - This section provides you an attached storage where you can store your datasets.

MODULE 18: MODELS v/s ALGORITHMS

- It is extremely important to learn the difference between the two terms, models and algorithms.
- We can understand these two terms in simple words in the following way:
  - **Model**: A model is a specific representation learned from data.
  - **Algorithms:** An algorithm can be understood as a process of learning.
- The choice of your algorithm can heavily affect the internal representation and the structure of the resultant model.
- In other words,

Model = Algorithm (Data)

- Examples of models:
  - Set of coefficients
  - Equation learned from the data.

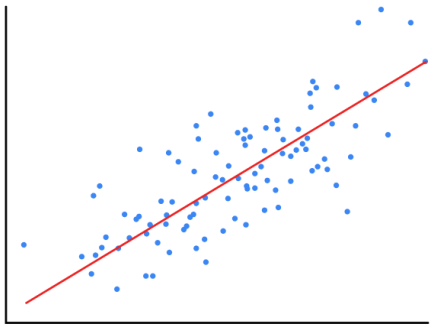
- Examples of algorithms:
    - Convolutional Neural Networks
    - Least Squares Linear Regression

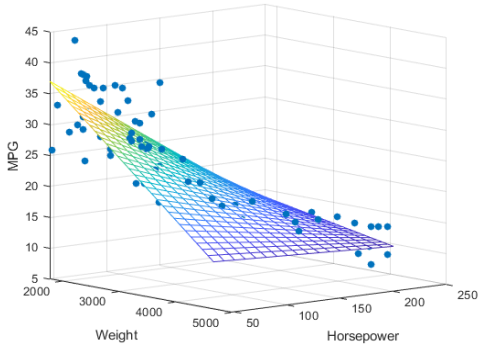
MODULE 19: PRELAUNCH LAB

LAB RESERVED. MOVE ON TO THE NEXT MODULE.

MODULE 20: LINEAR REGRESSION

- Linear Regression is an algorithm that uses a straight line or a plane to describe the relationship between the variables.





- The general equation of a line is:

  y = mx + b

  where m = slope of the line

  b = y-intercept

- In Machine Learning, a Linear Regression is often represented as:

  $y = B_0 + B_1 * x$

  where B_0 is called the **Bias**

  and B_1 is called the **coefficient**

  This is pretty similar to the equation of line, just with different alphabets used for representation.

- On the similar notes, a Multiple Linear Regression is often represented as:
    - **$y = B_0 + B_1 * x_1 + B_2 * x_2 + B_3 * x_3 + B_4 * x_4$ …**
- Training a regression model simply means finding the best values of bias and coefficient for your model.
- **Cost Function**
    - While training your model, you start with a random value of bias and coefficient.
    - This leads to some difference in the actual value and the predicted value. This is what we call **error**.
    - The purpose of the cost function is to minimize the value of this error so as to push the values of the bias and coefficient in the direction which would lead to lower loss.
    - Doing this process iteratively will eventually result in the loss of your model reaching to the lowest value (a minima).

- The most commonly used cost function is the **root mean squared error (RMSE)**.

- There are certain assumptions or conditions that need to be kept in mind while using Linear Regression.
  - **Linear Assumption**
    - Needless to say, this algorithm assumes that there is a linear relationship between the input and the output variables.
    - If your data doesn't appear to be linear, you may want to transform it to introduce linearity in your data prior to feeding your data into the model.
  - **Remove Collinearity**
    - When two variables are collinear, they can be modelled by the same line.
    - In other words, one input variable can be used to accurately predict the other variable.
    - **Example**: If you wish to predict the education level using the factors like **number of years spent at school, if an individual is male, if an individual as a female**.
    - In the above case, **if an individual is a female** can be accurately predicted by the factor **if an individual is a male**.
    - Collinearity makes your model less consistent.
  - **Gaussian Distribution**
    - Linear Regression assumes that the distance between the output variable and the real data (residual) is normally distributed.
    - If this is not the case with your data, you might want to apply some transformations on your data to make it normally distributed.
  - **Rescale Data**
    - Linear Regression is very sensitive to the distance among the data points.
    - Thus, it's a good idea to rescale your data using standard practices like **normalization** and **standardization**.
  - **Remove Noise**
    - Linear Regression is very sensitive to outliers and noise in data.
    - So, it's a good practice to clean your data and remove any noise prior to feeding the data to the model.
  - **Formulae**
    - To calculate the slope of the line:

$$B1 = \frac{\sum_{i=1}^{n}(x_i - mean(x)) \times (y_i - mean(y))}{\sum_{i=1}^{n}(x_i - mean(x))^2}$$

    - To calculate the intercept (or bias) of the line:

$$B0 = mean(y) - B1 \times mean(x)$$

    - To calculate the Root Mean Squared (RMS) error:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(p_i - y_i)^2}{n}}$$

MODULE 21: LINEAR REGRESSION (Revision) (DAY 3/50)

- The general equation of a line is **y = mx + b**.
- Simple Linear Regression uses a straight line to describe the relationships between the variables.
- Multiple Linear Regression uses a plan to describe the relationships between the variables.
- Multiple Linear Regression involves more than one input variable.
- Linear Regression assumes that the input and output variable follow the linear relationship.
- In Machine Learning and Linear Regression, different terms are used to represent the same (or similar) concept. Some of them are:

Linear Regression	**Machine Learning**
Slope	Coefficient
Intercept	Bias
RMSE (Root Mean Squared Error)	Cost Function

- The training process is a process of minimizing the error.
- A **cost function** is used to calculate the error of a model.
- A Linear Regression model is sensitive to noise and outliers, and the correlated variables should not be used as input while training.
- It is a good practice to scale your data using standard methods like **Normalization** and **Standardization**.

MODULE 22: LAB INSTRUCTIONS

MODULE 23: TRAIN A LINEAR REGRESSION MODEL IN AZURE ML STUDIO

MODULE 24: LAB WALKTHROUGH

Below are the steps followed to train a Linear Regression model on Azure ML Studio:

- In the **Assets** section, select the **Datasets** option. Click on **Create dataset** and select '**From web file'** option from the drop-down menu.
- Paste the web-link of your dataset and select different options to set up your dataset. Once your dataset is imported, it will be seen as an entry in the **Datasets** section.
- After setting up your dataset, move on to the **Designer** option under the **Authoring** section.
- Click on the **+** icon to create a new pipeline. Once your pipeline is successfully created you will see an entry in the **Designer** section.
- Clicking on the **+** icon will take you to your designer window where your first step will be to set up a computer for your designer on which your model can be trained.
- After this, drag your dataset from under the **Datasets** tab and drop it on to the empty canvas.
- Next, drag the **Split Data** tool from under the **Data Transformations** tab and drop it to the canvas next to your dataset.
- Select the train/test split for your dataset and connect your dataset to the **Split Data** tool.

- Next, drag the **Linear Regression Algorithm** from under the **Machine Learning Algorithms** tab and drop it to the canvas.
- Next, drag the **Train Model** module from under the **Model Training** tab and drop it to the canvas.
- Now, on your canvas, connect the output of the **Linear Regression** module to the first input of the **Train Model** module. Connect the first output of the **Split Data** tool to the second input of your **Train Model** module.
- After setting up the Train Model module, next drag the **Score Model** module from under the **Model Scoring & Evaluation** tab.
- Connect the output of the **Train Model** module to the first input of the **Score Model** module. Connect the second output of the **Split Data** module to the second input of the **Score Model** module.
- Next, drag the **Evaluate Model** module from under the **Model Scoring & Evaluation** tab and drop it on to the canvas.
- Now, connect the output of the **Score Model** module to the input of the **Evaluate Model** module.
- After setting up the entire pipeline, click on **Submit** button to submit your pipeline for run.
- After your model is trained successfully, you can browse through the results by clicking on the **Score Model** and the **Evaluate Model** modules.

MODULE 25: LEARNING FUNCTIONS

- Machine Learning algorithms aim to learn the function (**f**) which describes the mapping between data input variables (**X**) and an output variable (**Y**).

$$Y = f(X)$$

- Since the process extrapolates from a limited set of values, there will always be an error e which is independent of the input data (**X**) such that:

$$Y = f(X) + e$$

- The variable **e** in the above equation is called the **irreducible error** because no matter how good we get at estimating the target function, we cannot reduce this error.
- The irreducible error is different from the model error.
- Irreducible error is caused by the data collection process—such as when we don't have enough data or don't have enough data features.
- In contrast, the model error measures how much the prediction made by the model is different from the true output. The model error is generated from the model and can be reduced during the model learning process.
- In practice, its best to try a variety of algorithms and compare the results to see which produces the function with the least error.

MODULE 26: PARAMETRIC vs NON-PARAMETRIC (DAY 4/50)

- Machine Learning algorithms can be divided into **Parametric and Non-Parametric** based on the assumptions about the shape and structure of the functions they try to learn.
- **Parametric ML Algorithms**
    - Make assumptions about the mapping function and have a fixed number of parameters.
    - Any amount of data does not change the assumption about the mapping function since the form of the function is already selected and the data is only used to learn the coefficients.
    - Ex: Linear Regression algorithm of the form:

      $$\mathbf{Y = B_0 + B_1 * X_1 + B_2 * X_2 + B_3 * X_3}$$

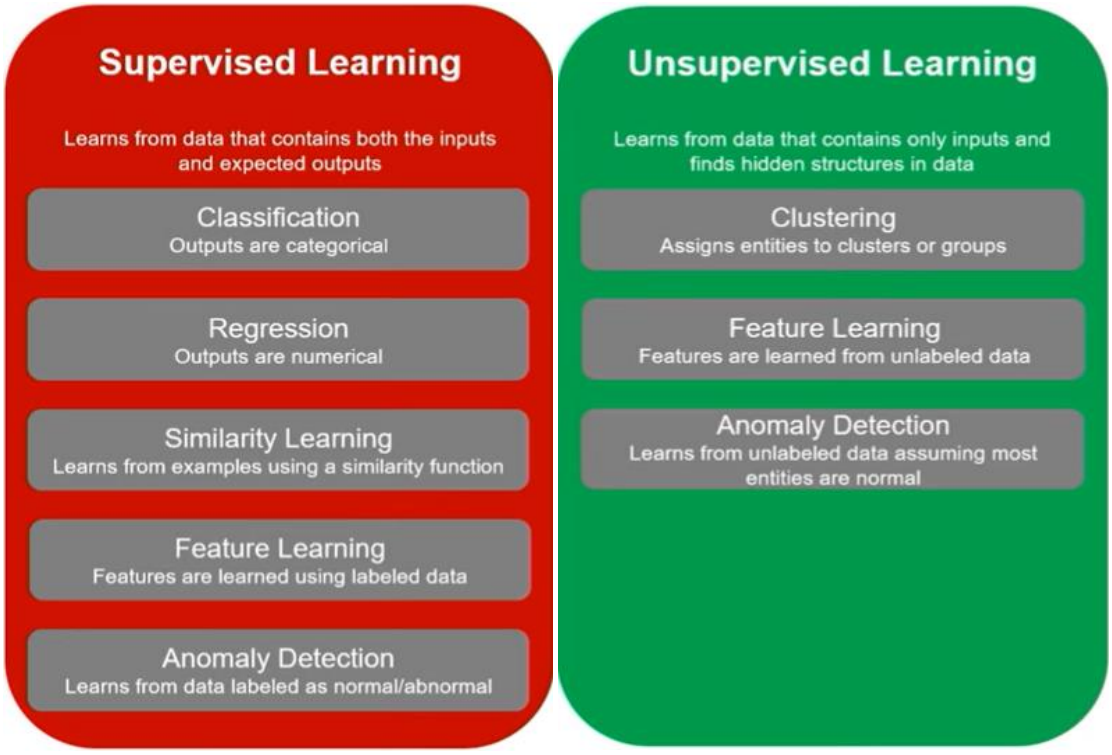
      Here, the form of the function is already decided by having 3 variables (X_1, X_2 and X_3). What we are learning using the training data is the values for the coefficients B_1, B_2 and B_3 which would best fit the data.
    - **Benefits**
        - Simpler and easier to understand, easier to interpret the results.
        - Learns faster from your data.
        - Less data is required to learn the mapping function.
    - **Limitations**
        - Highly constrained to the specific forms of the simplified functions.
        - Suitable for problems with limited complexity.
        - Poor fit in practice, unlikely to fit the underlying data.
- **Non-Parametric ML Algorithms**
    - Do not make any assumptions about the mapping function and are free to learn any functional form of the training data.
    - Ex: K Nearest Neighbours (KNN)
        - This algorithm does not make assumptions about the functional form, but learns the functional form of the data by using patterns found in the data.
    - **Benefits**
        - High flexibility i.e., these algorithms are capable of fitting wide variety of functional form, from simplest to very complex ones.
        - Provides more power by making least or no assumptions about the functional form of the training data.
        - Produces comparatively high performant model.
    - **Limitations**
        - More training data is required for training.
        - Slower to train in general because of the large number of parameters.
        - Overfitting on data is a big risk in which case explaining the prediction becomes difficult.

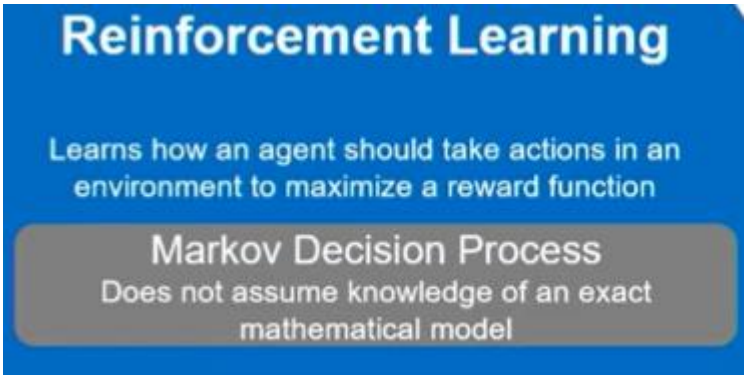
MODULE 27: CLASSICAL ML vs DEEP LEARNING

- It is important to note that all Deep Learning algorithms are Machine Learning algorithms, but not all Machine Learning algorithms are Deep Learning algorithms.
- Deep Learning algorithms are based on the concept of Neural Network while the Machine Learning algorithms are based on classical mathematical algorithms.
- **Deep Learning Advantages**
    - Provides better accuracy as compared to classical ML.
    - Provides better support for big data.
    - Suitable for high complexity problems.
    - Complex features can be learnt.
- **Deep Learning Disadvantages**
    - Difficult to explain the predictions
    - Require more computational power
- **Classical ML Advantages**
    - Required less computation
    - Suitable for smaller datasets
    - Easier to interpret the outcomes
- **Classical ML Disadvantages**
    - Inefficient on complex dataset
    - Requires feature engineering

MODULE 28: APPROACHES TO MACHINE LEARNING

- There are mainly 3 different approaches to Machine Learning:
    - **Supervised learning**
    - **Unsupervised learning**
    - **Reinforcement learning**



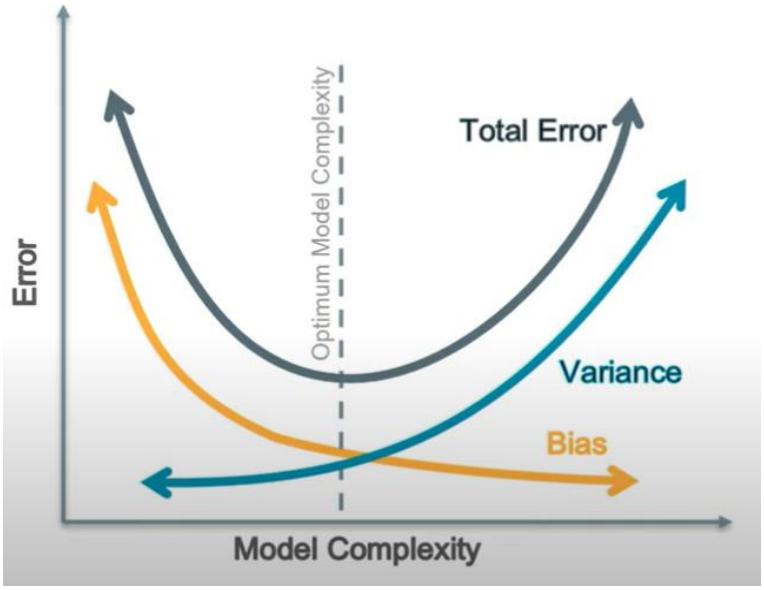


MODULE 29: THE TRADE-OFFS

- Machine Learning involves two major trade-offs that we need to deal with:
  - Bias vs Variance
  - Overfitting vs Underfitting
- **Bias vs Variance**
  - Bias
    - Bias measures how inaccurate model prediction is as compared to the actual/true output.
    - It is because of the erroneous assumptions made in the machine learning process to simplify the model and make the target function easier to learn.
    - High model complexity tends to have low Bias.
    - **Parametric** algorithms have high bias and low variance.
  - Variance
    - Variance measure how much the target function would change if a different training data is used.
    - Variance can be caused by modelling random noise in the training data.
    - High model complexity tends to have high Variance.
    - **Non-parametric** algorithms have low bias and high variance
- **Overfitting vs Underfitting**
  - Overfitting
    - This refers to the condition when the model performs very well on the training data, but does not perform well on the test data.
    - In other words, the model learns the training data rather than learning the function.
    - This type of model does not generalize well.
  - Underfitting
    - This refers to the condition when the model neither fits on the training data nor generalize on the test/new data.
- The **prediction error** can be considered as the sum of the **model error** (error produced by the model) and the **irreducible error** (error due to data).

  Prediction Error = Bias Error + Variance Error + Irreducible Error
- **Low Bias**
  - This means fewer assumptions about the target function.
  - Some of the examples are:

        - KNN
        - Decision Trees
    - Having fewer assumptions can allow the algorithm to generalize relevant relations between features and the target outputs.
- **High Bias**
    - This means more assumptions about the target function.
    - Some of the examples are:
        - Linear Regression
    - Having more assumptions can cause the model to miss some important relations between features and target outputs.
    - This can cause Underfitting.
- Low Variance
    - This means changes in the training data would lead to similar target functions.
    - Linear Regression is a good example of an algorithm having low variance.
- High Variance
    - This means changes in the training data would lead to a large change in the target functions.
    - Support Vector Machine (SVM) is a good example of an algorithm having high variance.
    - High variance causes a model allows a model to even learn the noise and thus causing overfitting.
- Increasing model complexity will reduce the bias as the model has more flexibility to learn the target function. However, this leads to an increase in the variance.
- On the other hand, decreasing model complexity will reduce the variance as the model will be less affected by the changing data. However, this would lead to high bias.
- The aim of the training process is to find the optimal model complexity where the bias error and the variance error are at equilibrium.



- Some of the ways of reducing Overfitting are as follows:
    - Reduce model complexity
    - Train on more data

- Early stopping of the training process
- Reduce dimensionality of the training data (using methods like PCA)
- k-fold cross-validation, where you first divide the dataset into k subsets and then train on k-1 subsets and test on 1 subset. Every time a new subset is used for testing while the other k-1 subsets are used for training.
- Hold out a Validation dataset to estimate how well your model is generalizing on the unseen data.