Data $\rightarrow$ $\boxed{ML}$ $\rightarrow$ Patterns
ML model

Features + Target $\xrightarrow{ML}$ ML Model

output data

eg:
Car price prediction

Features + ML Model $\rightarrow$ Predictions

ML
The process of extracting pattern from data

Data is of 2 types:
i) Features
ii) Target

A model is an artifact of training that contains all the patterns learned from the trained examples.

ML v/s Rule-Based System

eg: Email System
Spam classification

Rule-Based
Add conditions, if-else statements

ML
Get data — click $\boxed{spam}$ button for each email
Define & calculate features
Train & use the model

length of title
length of body
sender
sender domain
Description

All the features' values are converted to a numerical matrix & used to produce prediction

Prediction output comes as decimal value & represents a probabilistic value

Based on a threshold value, the classification output spam or not spam is evaluated

| Features (data) | Prediction (output) | Final outcome (decision) |
| --- | --- | --- |
| [0, 1, 0, ....] | 0.8 | Yes (≥ 0.6) |
| [1, 1, ...] | 0.3 | No (< 0.6) |

is spam

# Supervised Machine Learning → Teaching a model with examples & making predictions

Features (Data) $\xrightarrow{\text{Model}}$ Target (desired output)

$g(X) \approx y$

- $X$ → Matrix → features
- $y$ → Target variable
- $g$ → model
- $\approx$ → as close as target variable

Regression
o/p is continuous value

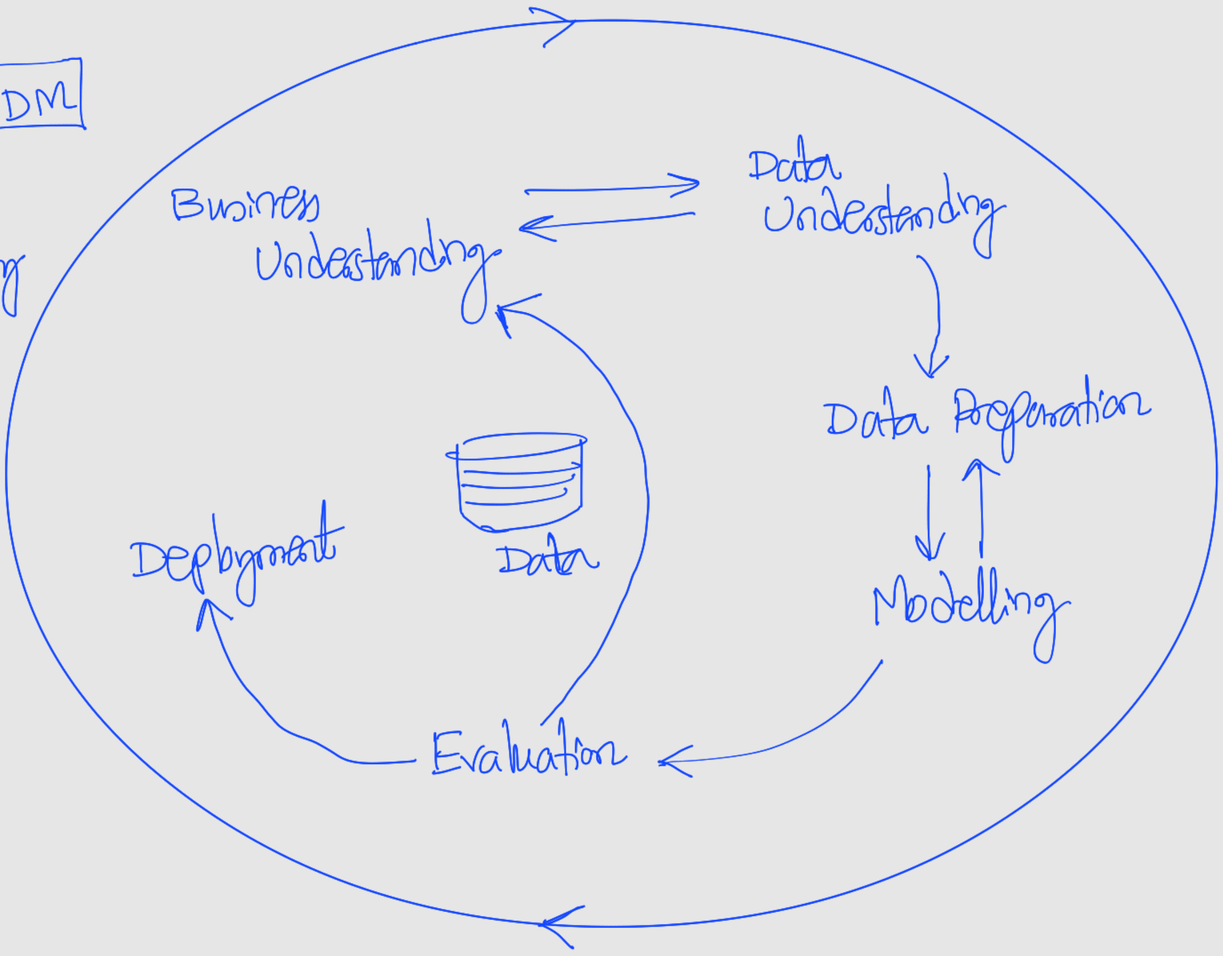
Classification
o/p is category
multiclass classification*** {cat, dog, horse}
Binary classification*** {spam, Not Spam}

*Ranking
eg: Recommender system
A function ranks the items based on priority or some ranking based on a value.
eg: how google search shows all the websites.

# Spam Detection

Mail $\xrightarrow{\text{model}}$ 0.6 → spam ✓ ($\geq 0.5$) / Not spam ×

CRISP-DM → methodology for organizing ML projects



1) Business Understanding

- Do we really need ML?

  Analyze to what extent it is a problem?

  if not, propose an alternate solution

- Define the Goal

  ↳ The goal has to be measurable

2) Data Understanding:

- Analyse available data sources
- Decide if we need to get more data

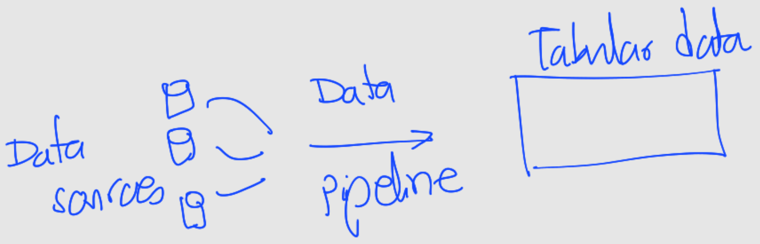
Spam detection:

we have a report spam button
is the data behind this button good enough?
is it reliable?
Do we track it correctly?
Is the dataset large enough?
Do we need to get more data?

* Identify the data sources:
  → If may influence the goal
  → we may need to go to previous step & adjust it

3) Data Preparation:

- clean the data
- Build the pipelines
- Convert into tabular form




4) Modelling:

$$\begin{bmatrix} 0, 0, 0, 1 \\ 1, 0, 1, 0 \\ \vdots \; \ldots \end{bmatrix}$$

Once the data is in the form of matrix (of 0 or 1), it is good to be under a model.

5) Evaluation:

- Is the model good enough?
  - → Have we reached the goal?
  - → Do our metrics improve?

Evaluate on the test group

- Do a retrospective:
  - → Was the goal achievable?
  - → Did we solve/measure the right thing?

- After that we may decide to:
  - → Go back and adjust the goal
  - → Roll the model to more users/all users
  - → Stop working on the project

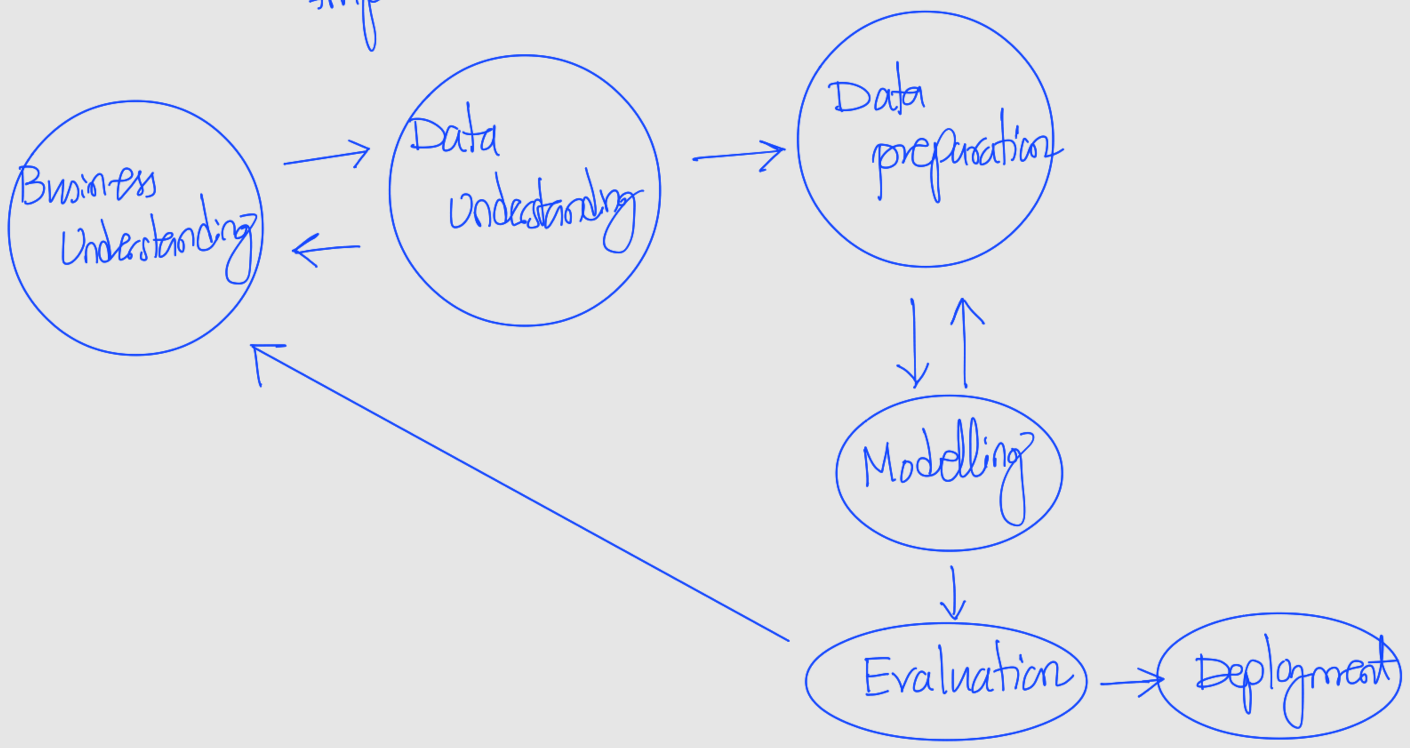
6) Deployment:

Evaluation + Deployment

- Roll the model to all the users
- Proper monitoring
- Ensuring the quality & maintainability

7) Iterate! ML projects require many iterations.

- Start simple
- Learn from feedback
- Improve



# Modelling / Model Selection

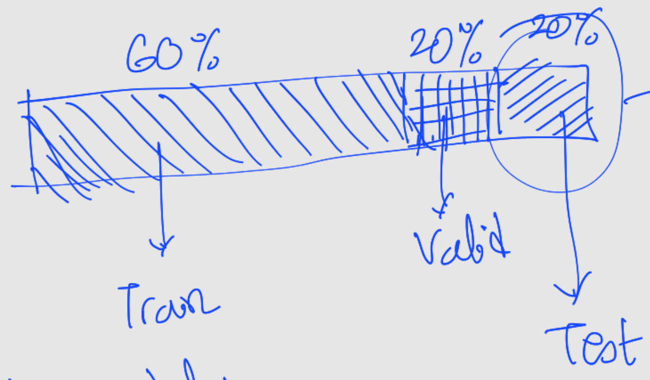
which model to choose?

- Logistic regression
- Decision Trees
- ....

Train set | Validation set | Test set

$\hat{y}$ → predicted output

$y$ → target variable




→ Hide this & put it away

1) split the data
2) First we use the Train data to get the model(s)
3) we apply the models on the validation dataset to get the best model
4) we apply the best model on test dataset to compare the accuracies of valid & test datasets
5) The accuracies should be similar or close

packages required:

jupyter
numpy
pandas
scikit-learn
seaborn

# NUMPY

Creating Arrays
Multi-dimensional Arrays
Randomly Generated Arrays → np.random.
Element wise operations

Comparison Operations

Summarizing operations $\longrightarrow$ mean(), min(), max(), std().

# Linear Algebra Refresher

Vector Operations

Multiplications

Inverse

1) $2\begin{bmatrix}2\\4\\3\end{bmatrix} = \begin{bmatrix}4\\8\\6\end{bmatrix}$

↳ column vector

2) $\begin{bmatrix}2\\6\\3\end{bmatrix} + \begin{bmatrix}3\\1\\2\end{bmatrix} = \begin{bmatrix}5\\7\\5\end{bmatrix}$

Multiplication

→ vector-vector multiplication

→ matrix-vector mul

→ matrix-matrix mul

vector-vector multiplication → Dot product → $\Sigma$(element wise multiplication)

Note: in numpy, $u * v$ gives the array with element-wise multiplication.

$$\begin{array}{c}u\\ \begin{bmatrix}2\\3\\4\end{bmatrix}\end{array} \cdot \begin{array}{c}v\\ \begin{bmatrix}1\\3\\6\end{bmatrix}\end{array} = \begin{bmatrix}2\\3\\4\end{bmatrix} \vdots \begin{bmatrix}1\\3\\6\end{bmatrix}$$

$$= 2\cdot 1 + 3\cdot 3 + 4\cdot 6$$

$$= 2 + 9 + 24$$

$$= 35$$

$$\begin{array}{c}u^T\\ \begin{bmatrix}2 & 3 & 4\end{bmatrix}\end{array} \times \begin{array}{c}v\\ \begin{bmatrix}1\\3\\6\end{bmatrix}\end{array}$$

$$= u^T \times v = u \cdot v = \sum_{i=1}^{n} u_i \cdot v_i = u.\mathrm{dot}(v)$$

(in Numpy)

Matrix-Vector Multiplication ≡ u.dot(v) [in numpy]

$$
\begin{array}{c c}
 & u \\
\begin{array}{c} u_0 \\ u_1 \\ u_2 \end{array} &
\begin{bmatrix} 2 & 4 & 5 & 6 \\ 1 & 2 & 1 & 2 \\ 3 & 1 & 2 & 1 \end{bmatrix}_{m \times n}
\end{array}
\overset{v}{\begin{bmatrix} 1 \\ 0.5 \\ 2 \\ 1 \end{bmatrix}}_{n \times 1}
=
\begin{bmatrix} u_0 \times v \\ u_1 \times v \\ u_2 \times v \end{bmatrix}_{m \times 1}
$$

vector-vector multiplication

Matrix-matrix multiplication:

$$
\overset{u}{\begin{bmatrix} 2 & 4 & 5 & 6 \\ 1 & 2 & 1 & 2 \\ 3 & 1 & 2 & 1 \end{bmatrix}}
\begin{array}{c}
v \\
\begin{bmatrix} 1 & 1 & 2 \\ 0 & 0.5 & 1 \\ 0 & 2 & 1 \\ 2 & 1 & 0 \end{bmatrix} \\
\begin{array}{ccc} v_0 & v_1 & v_2 \end{array}
\end{array}
=
\overset{uv}{\begin{bmatrix} uv_0 & uv_1 & uv_2 \end{bmatrix}}
$$

Intro to Pandas

- DF
- Series
- Index
- Element-wise operations
- Accessing Elements
- String operations (.str)
- Grouping
- Filtering
- Sorting
- Summarizing operations
- Converting to dict
- Converting to ndarray