Intro to ML + Recap

Freshers Introductory Meet for ML/AI
@ SSN Coding Club

What we will be covering..

Scan to ask questions anonymously in Slido or ask away in the Teams chat!

1. What machine learning is..
2. A timeline of events for the ML domain of this club.
3. How to get started learning ML as a fresher.
4. A quick recap of the first 2 meets.

Timeline of Events

Recording Links, Code and Resources
for both Meet 1 and 2:

github.com/ssncodingclub/Machine-Learning

1. Intro to ML - Scope, Roadmap and a simple analysis of a dataset - **Done**
2. Building a simple ML pipeline from scratch in Python - Linear Regression - **Done**
3. Freshers Intro + Recap Meet - **Now**
4. Building a binary classifier - Logistic Regression - **In 1 to 2 weeks**
5. Beginners Contest - based on what we have covered till now - **Sometime in Jan**

What is Machine Learning?

The README file here has a checklist-style roadmap to get started with ML:

github.com/ssncodingclub/Machine-Learning

- Machine learning allows computers to make intelligent decisions (i.e: predictions) by observing **real-world data**, *without* the need for explicit programming / hard coding of the decision logic.
- Many ML models derive from ideas in **statistics** that have existed for a long time. Hence, knowledge of fundamental statistics is a requirement before getting started with ML.
- To get started, knowledge of 12th grade math (linear algebra, probability distributions and stats) and basic Python (understanding functions, loops, arrays, classes, installing packages) should do.

Upcoming sessions

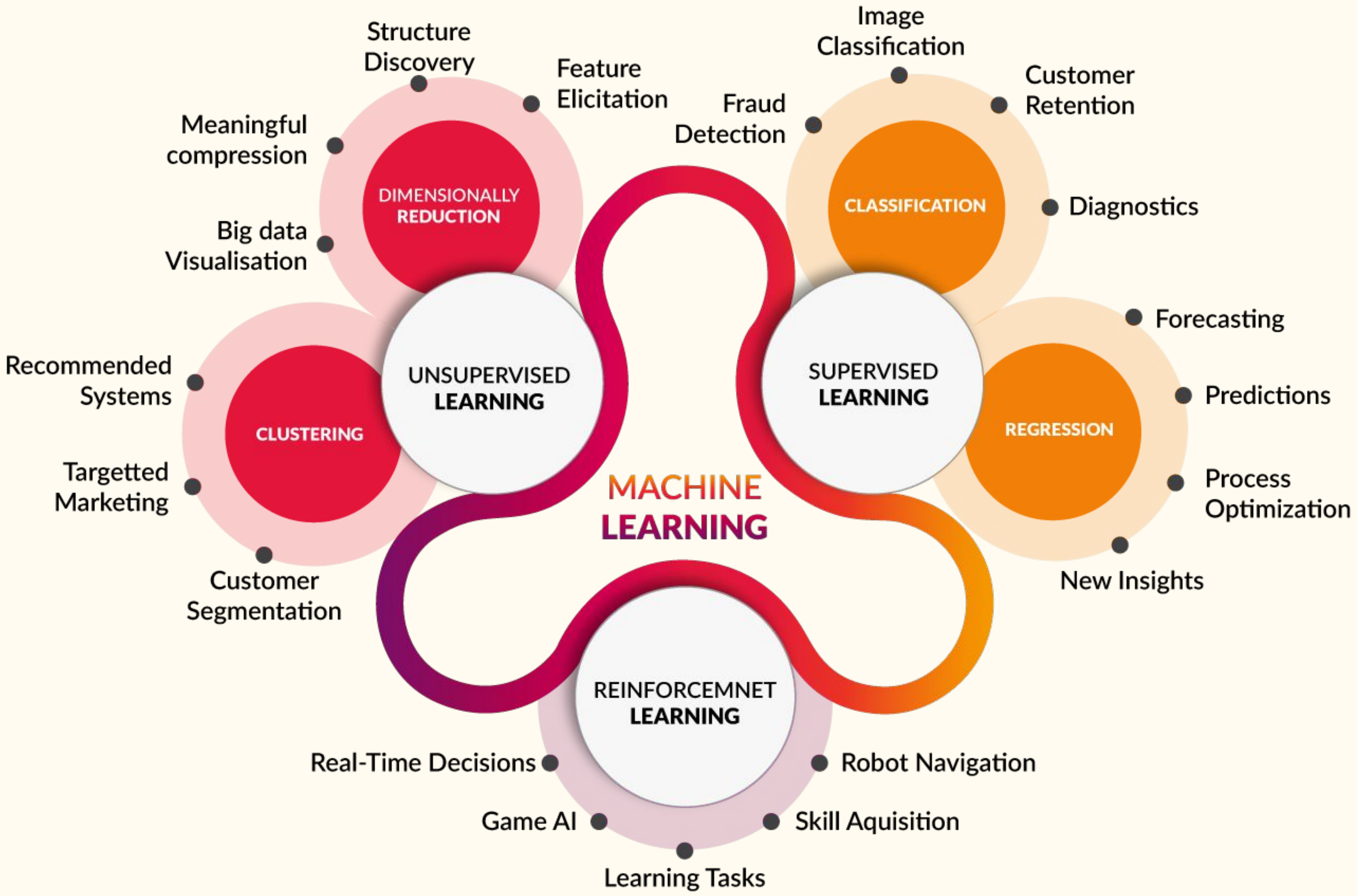
Tentatively, the upcoming sessions will focus on:

- Teaching core ML concepts
- Building simple ML models
- Understanding how to analyse and improve their performance, why deep learning is needed
- The current state of ML research and how ML models are deployed.

Areas where ML is useful

- Computer Vision / Image Processing
- Speech Recognition
- Traffic prediction
- Product recommendations
- Email Spam and Malware Filtering
- Online Fraud Detection
- Stock Market trading
- Automatic Language Translation (NLP)

Types of ML

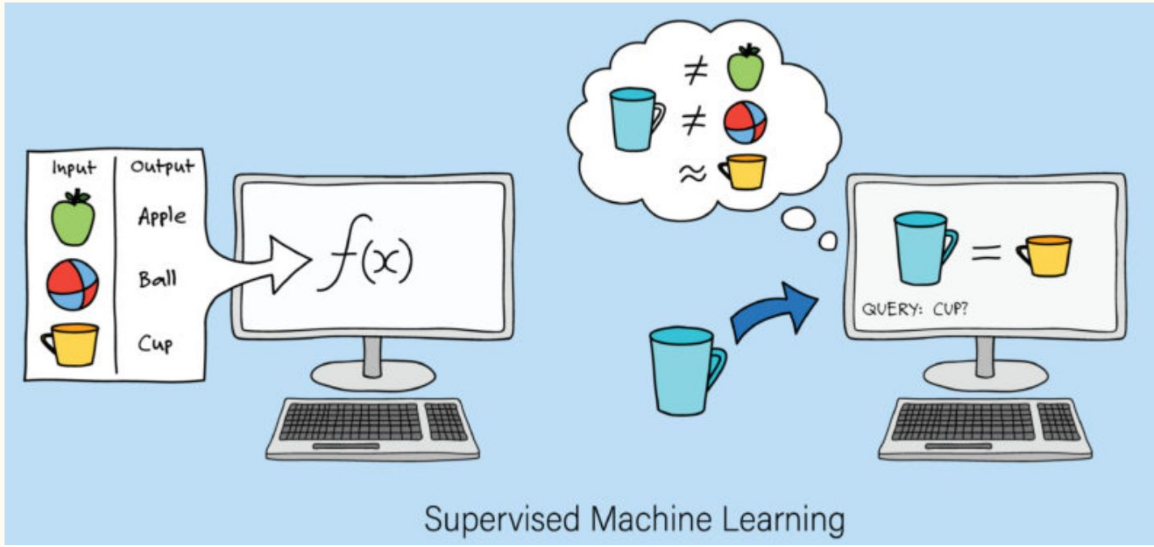

MACHINE LEARNING
UNSUPERVISED LEARNING
DIMENSIONALLY REDUCTION
Structure Discovery
Feature Elicitation
Meaningful compression
Big data Visualisation
CLUSTERING
Recommended Systems
Targetted Marketing
Customer Segmentation
SUPERVISED LEARNING
CLASSIFICATION
Image Classification
Customer Retention
Fraud Detection
Diagnostics
REGRESSION
Forecasting
Predictions
Process Optimization
New Insights
REINFORCEMNET LEARNING
Real-Time Decisions
Robot Navigation
Game AI
Skill Aquisition
Learning Tasks

1. Supervised Learning

Learning a function that maps an input to an output based on example labelled input-output pairs.

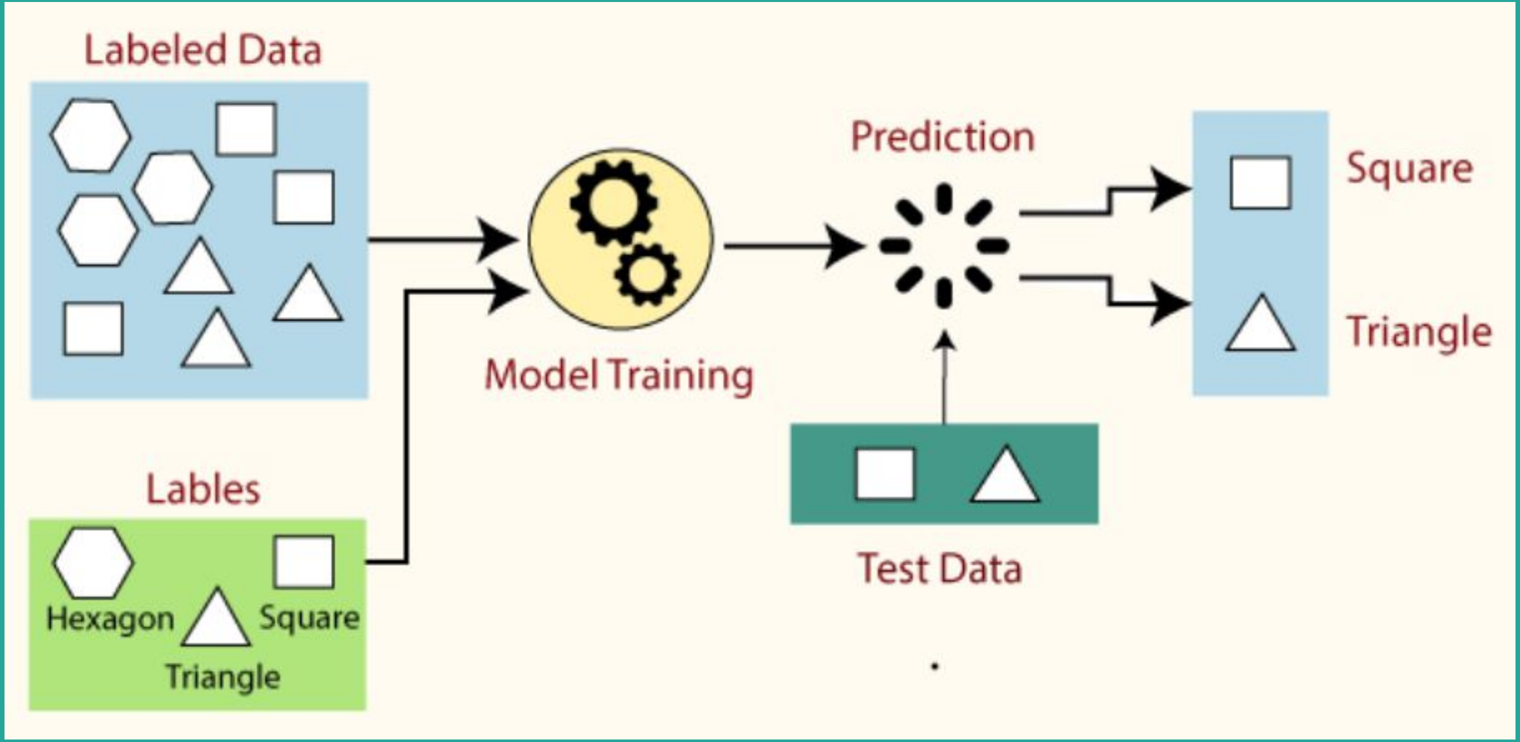
It is used in the context of two types of problems typically:

1. Classification
2. Regression



A. Classification

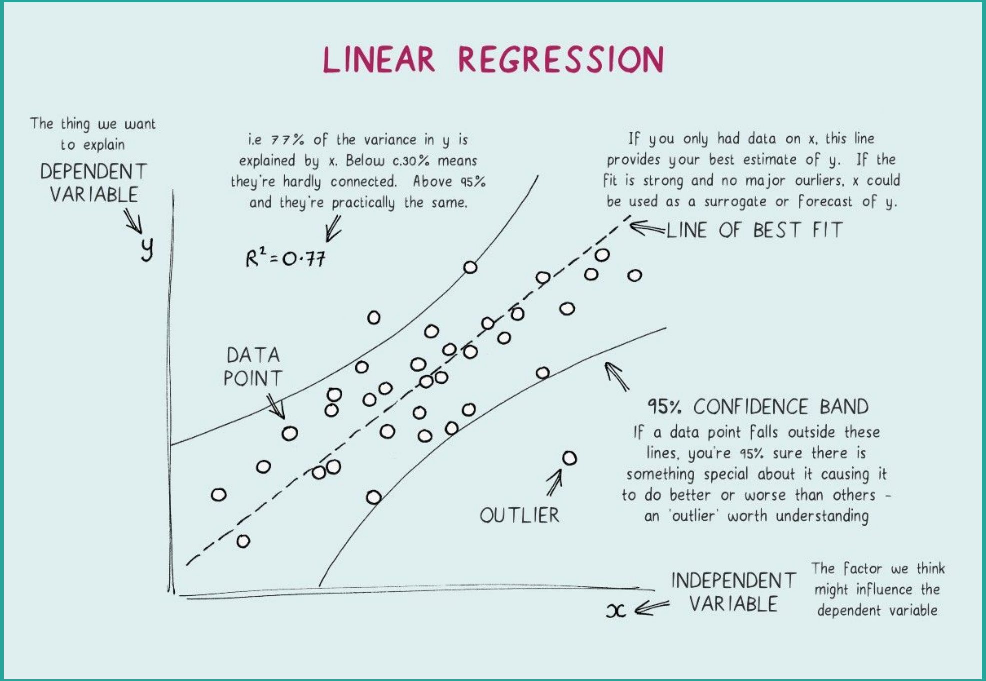
Uses a learning algorithm to learn a function that maps the input data to one of the output classes or categories.



B. Regression

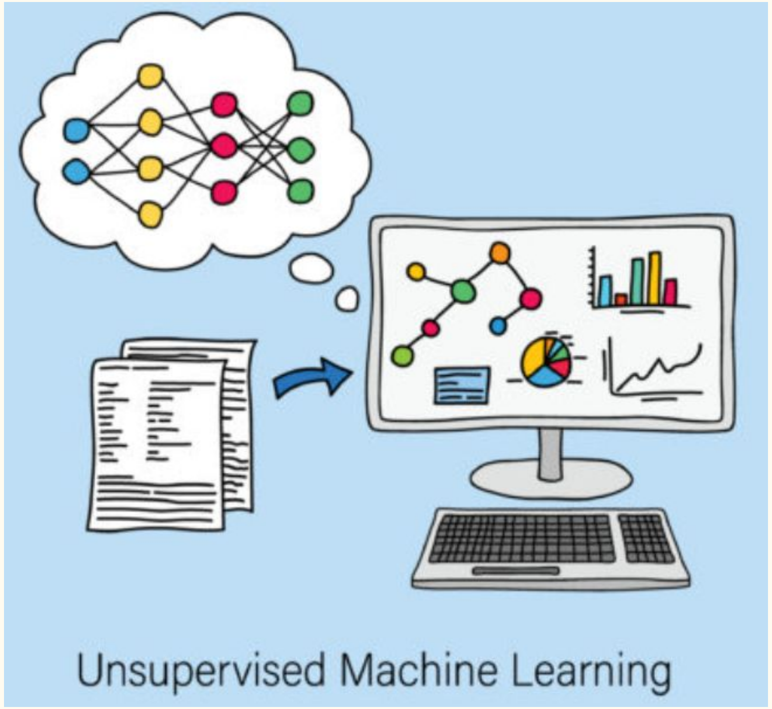
Learns the relationship between dependent and independent variables.

(You will learn to implement 2 fundamental regression techniques from scratch using Python in this session!)


LINEAR REGRESSION
The thing we want to explain
DEPENDENT VARIABLE
y
i.e 77% of the variance in y is explained by x. Below c.30% means they're hardly connected. Above 95% and they're practically the same.
R² = 0·77
If you only had data on x, this line provides your best estimate of y. If the fit is strong and no major ourliers, x could be used as a surrogate or forecast of y.
LINE OF BEST FIT
DATA POINT
95% CONFIDENCE BAND
If a data point falls outside these lines, you're 95% sure there is something special about it causing it to do better or worse than others - an 'outlier' worth understanding
OUTLIER
INDEPENDENT VARIABLE
x
The factor we think might influence the dependent variable

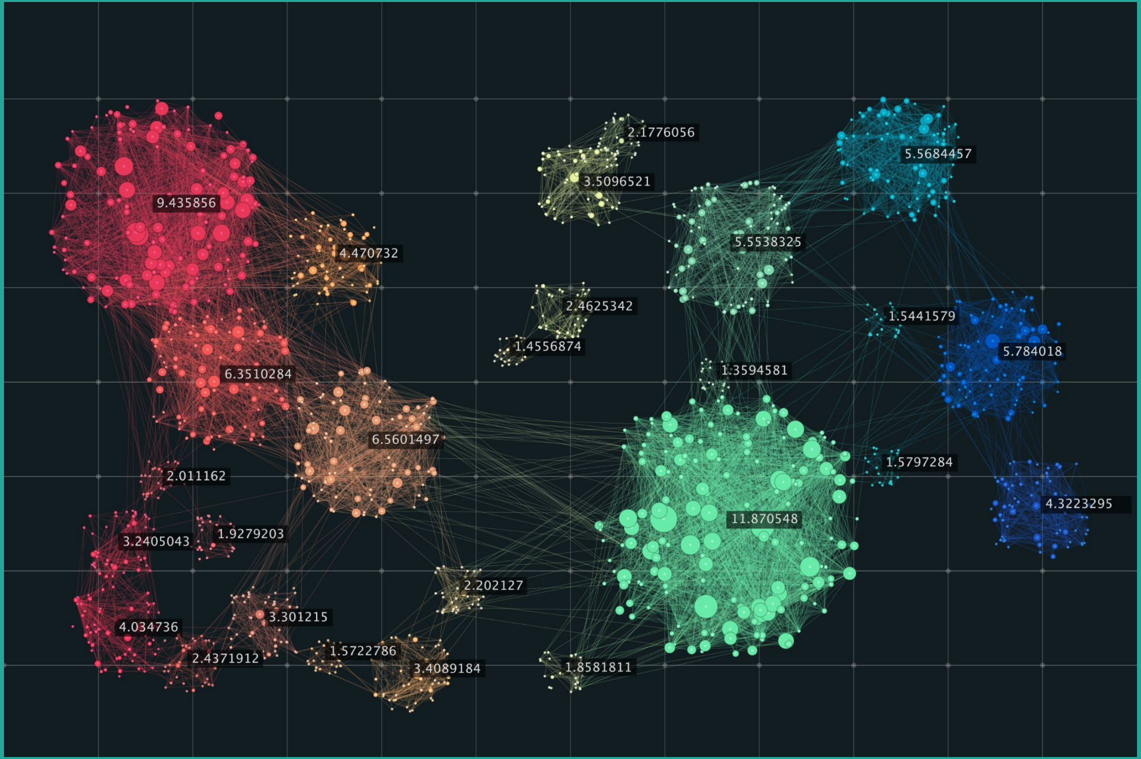
2. Unsupervised Learning

1. The data is not labelled
2. Self-discover any patterns in the training data set
3. It is used in the context of two types of problems typically-
    a. Clustering
    b. Dimensionality Reduction



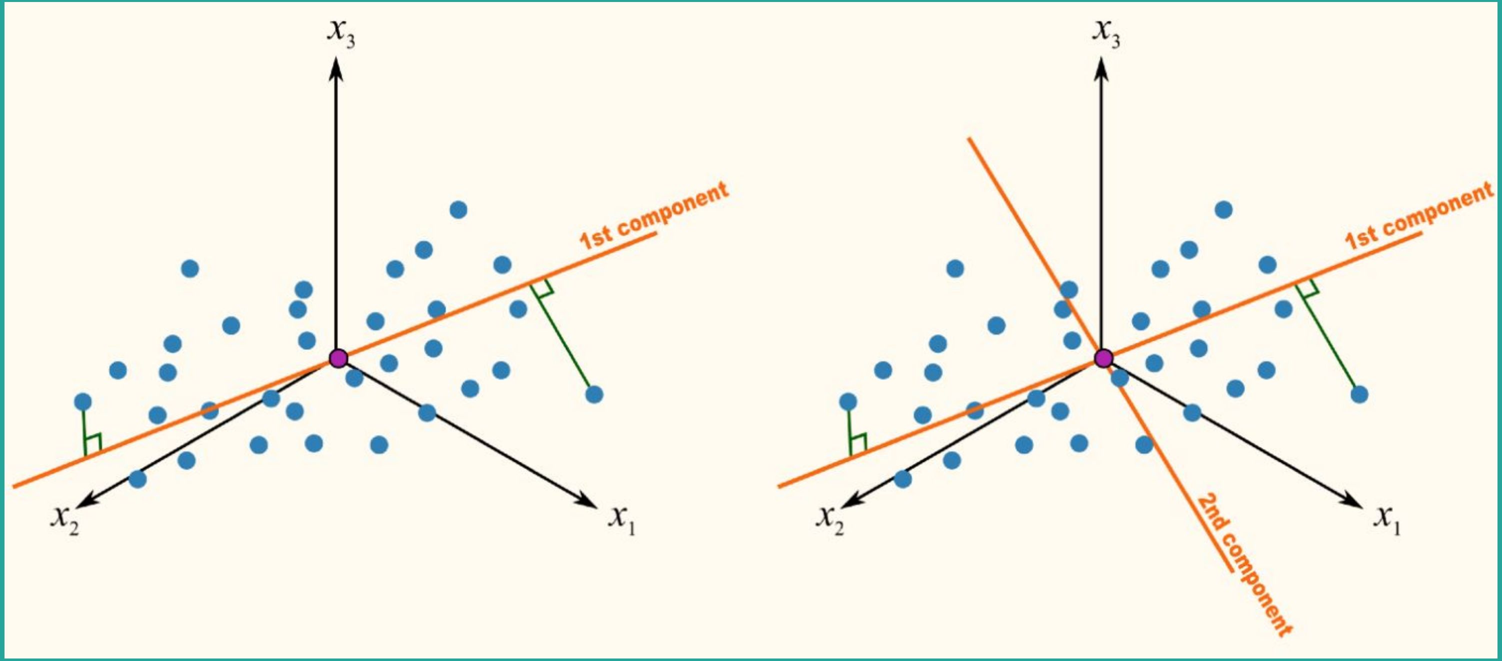
A. Clustering

Clusters training examples into categories with similar features using various metrics

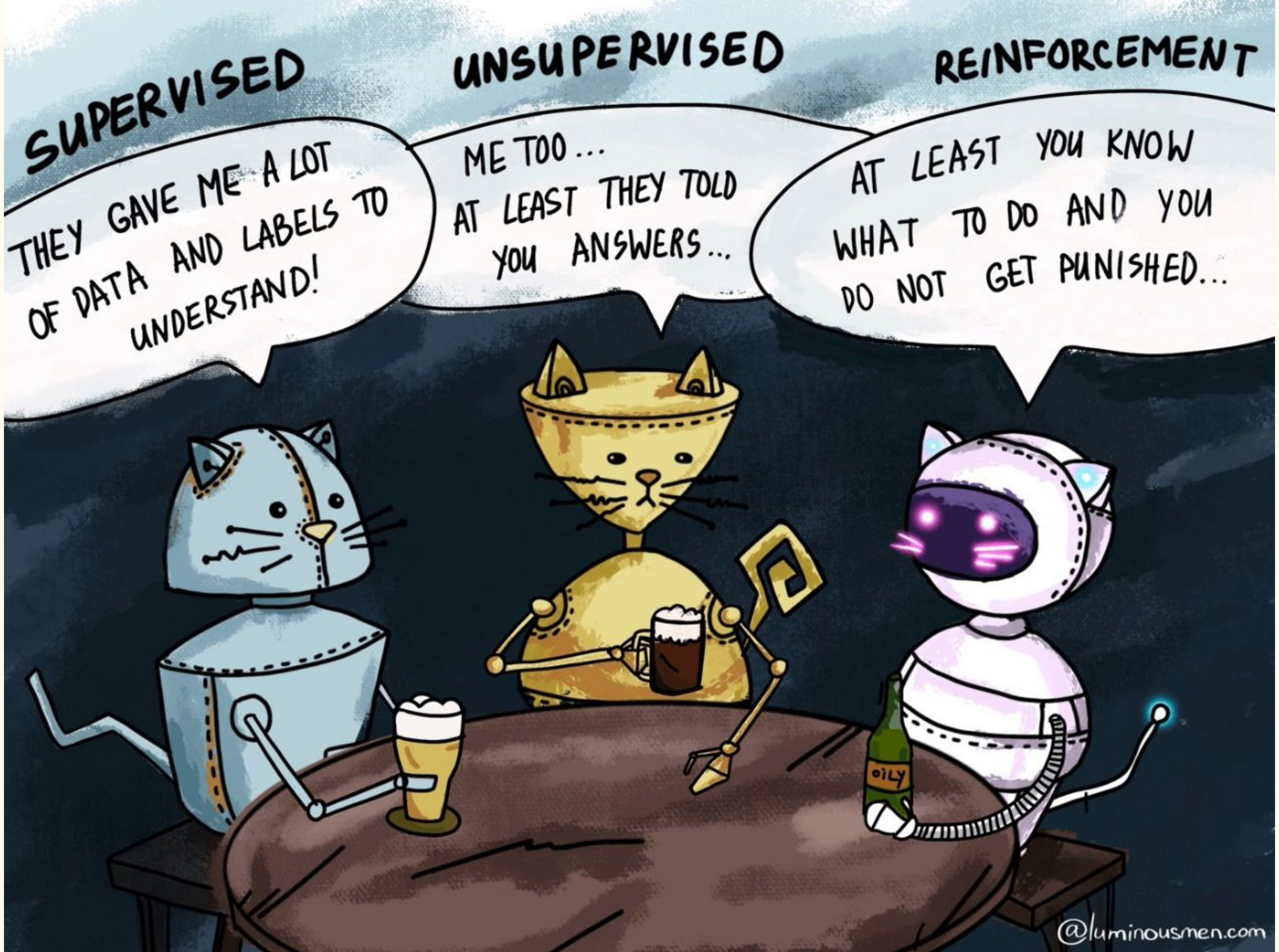


B. Dimensionality Reduction - Principal Component Analysis

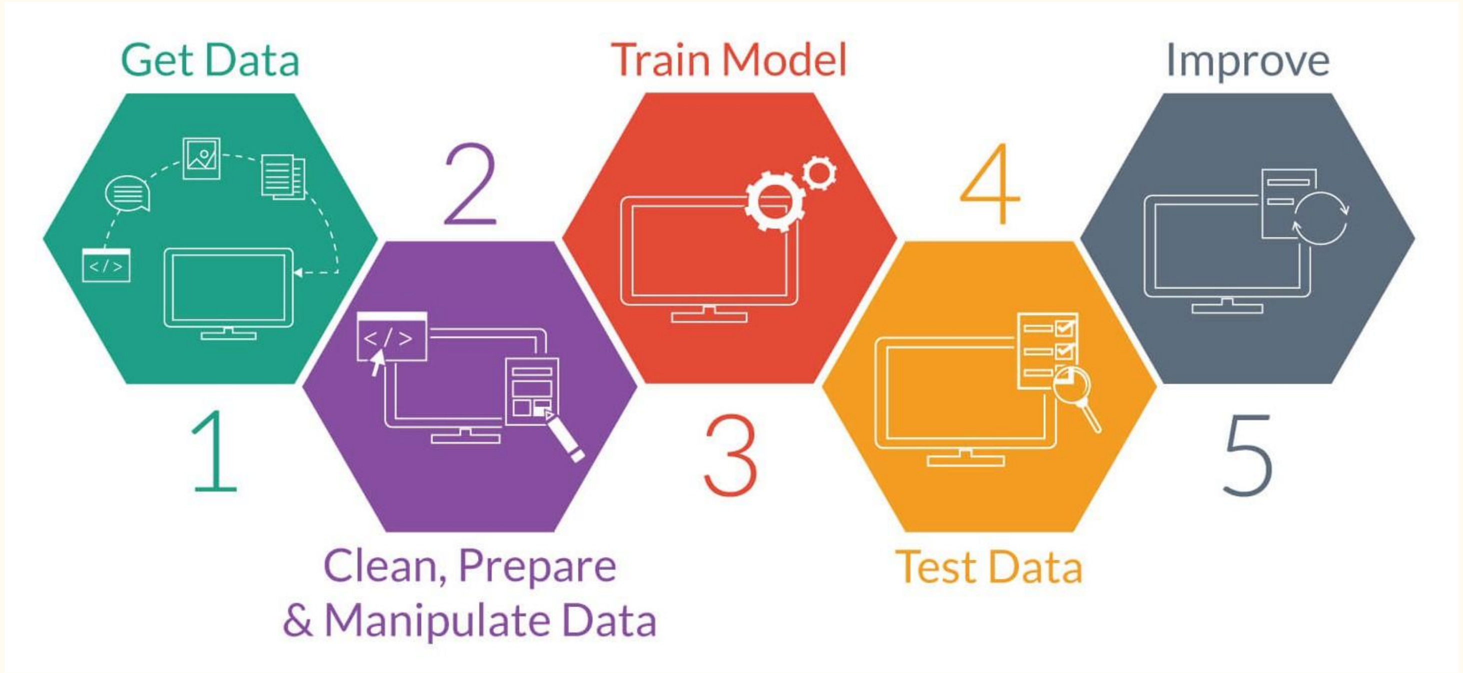
Finds ways to compress the training data set by identifying which features are most useful for discriminating between different training examples, and discarding the rest.



Supervised Learning	Unsupervised Learning
● Trained using labeled data, essentially learns a mapping function from inputs to outputs	● Trained using unlabeled data, discovers patterns within the data to extract information
● Could take direct feedback	● Doesn’t take any feedback
● Requires labelled data (usually done manually) which is sufficiently pre-processed.	● Lesser preprocessing of data but coming at the cost of larger training data, slower convergence to results, increased computational and storage requirements, etc.


SUPERVISED
THEY GAVE ME A LOT OF DATA AND LABELS TO UNDERSTAND!
UNSUPERVISED
ME TOO... AT LEAST THEY TOLD YOU ANSWERS...
REINFORCEMENT
AT LEAST YOU KNOW WHAT TO DO AND YOU DO NOT GET PUNISHED...
OILY
@luminousmen.com

General building blocks of any ML pipeline



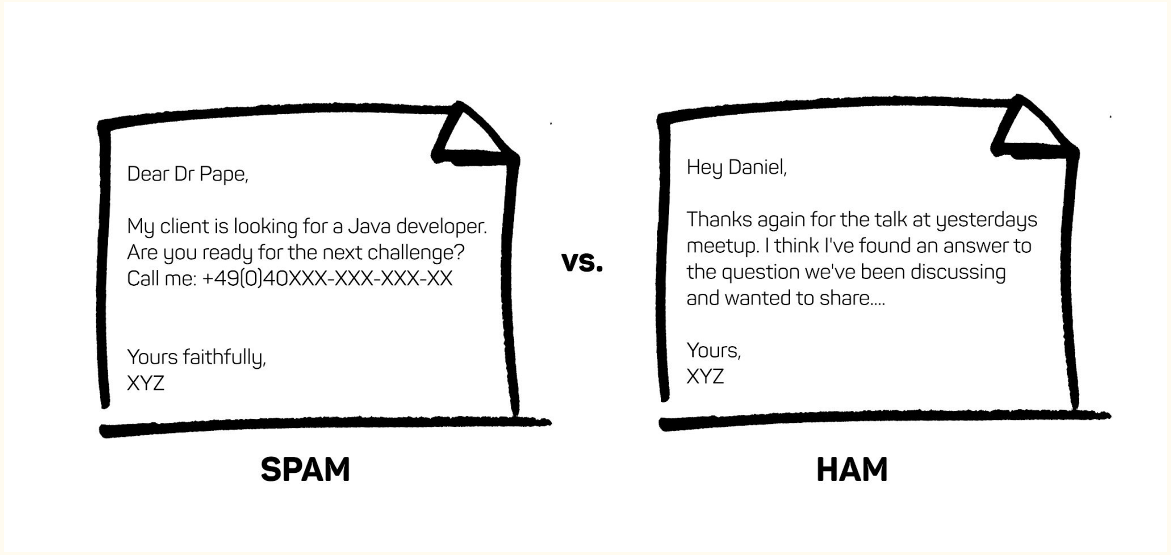
Fundamentals in ML:

Every ML (and even DL) model share in common a few fundamental concepts.

1. **Model** - A function that maps the input to our expected output: y = f(x), it's characterised by a set of **parameters** (or **weights**) that our algorithm has to learn (eg: y = w1.x + w2)
2. **Train and Validation Data:** The entire dataset of (x, y) pairs is split in a 80-20 fashion, 80% is used for training the model and 20% is used to validate if the performance is good.
3. **Loss Function:** Given a set of weights and a dataset, it gives a number that says how close the models predictions are to the expected y values. (the lower, the better)
4. **Optimizer**: An algorithm that tweaks the weights of the model so as to minimise the loss
5. **Metrics**: Other metrics like accuracy that can also be used to judge how a model is doing, in addition to the loss function.

Examples:

You’re given a dataset of emails, with each email labeled as spam or non-spam.



Dear Dr Pape,

My client is looking for a Java developer.
Are you ready for the next challenge?
Call me: +49(0)40XXX-XXX-XXX-XX

Yours faithfully,
XYZ

vs.

Hey Daniel,

Thanks again for the talk at yesterdays meetup. I think I've found an answer to the question we've been discussing and wanted to share....

Yours,
XYZ

Examples:

You’re given a dataset of emails, with each email labeled as spam or non-spam.

1. Supervised or unsupervised?

Examples:

You’re given a dataset of emails, with each email labeled as spam or non-spam.

1. Supervised or unsupervised?
2. Classification or a regression problem?

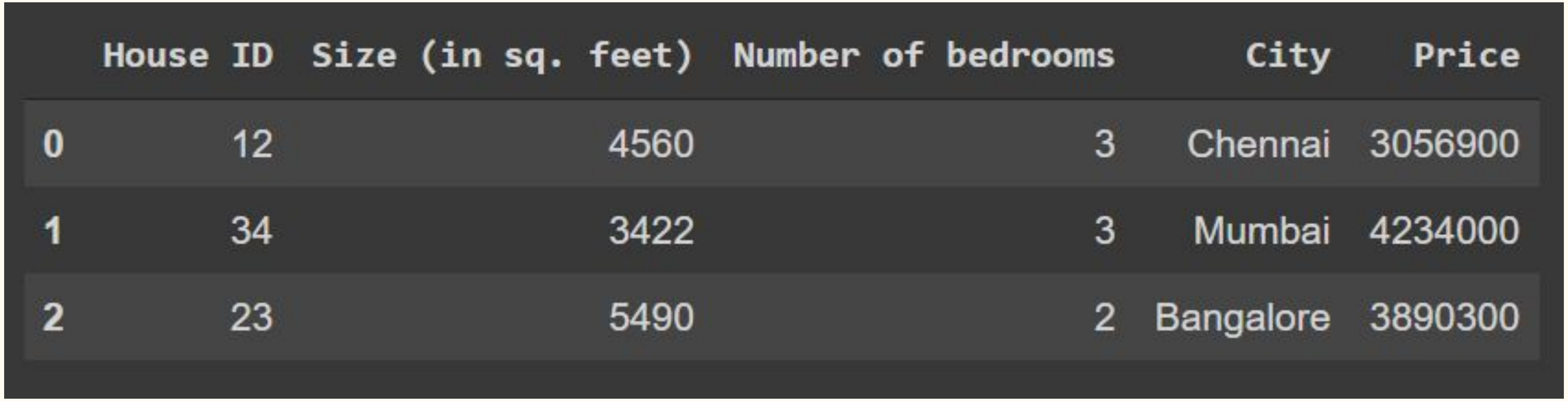
Examples:

You’re given a dataset of emails, with each email labeled as spam or non-spam.

1. Supervised or unsupervised?
2. Classification or a regression problem?
3. What metric would you use to get a general idea of how the classifier is doing?

Linear Regression

- Consider a problem where you want to predict the prices of different houses, given some information about them.
- So the price (target variable we wish to predict) can be **modeled** as a linear function of size and number of bedrooms. (is house ID relevant?)

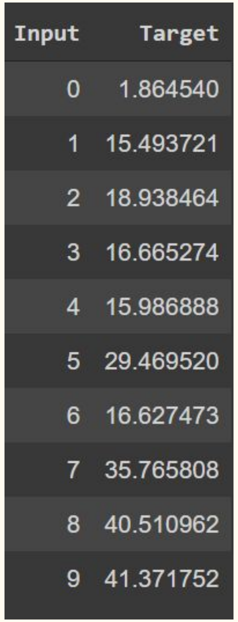


	House ID	Size (in sq. feet)	Number of bedrooms	City	Price
0	12	4560	3	Chennai	3056900
1	34	3422	3	Mumbai	4234000
2	23	5490	2	Bangalore	3890300

Source: ML Meet 2 Notebook

Linear Regression

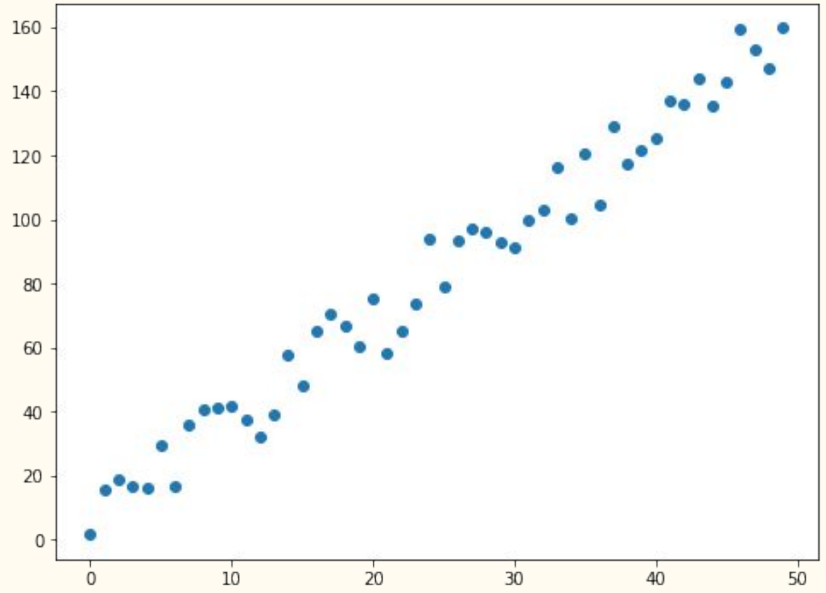
- Consider a simpler version of this problem, where the target variable ‘y’ only depends on one input variable ‘x’.
- Thus, a linear regression model would be: **y = mx + c**, where m and c are the slope and y-intercepts respectively.
- Now, given a list of such (x, y) pairs (the dataset), linear regression aims to find the m and c values that fit the data the best.



Input	Target
0	1.864540
1	15.493721
2	18.938464
3	16.665274
4	15.986888
5	29.469520
6	16.627473
7	35.765808
8	40.510962
9	41.371752

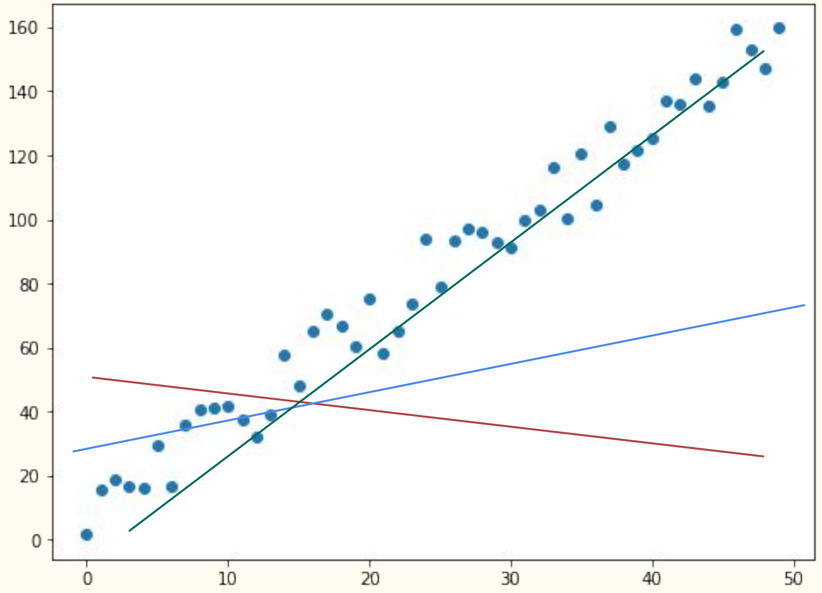
Linear Regression

- This shows the various (x, y) pairs plotted as a graph.
- Clearly, x and y have a linear relationship but they don't follow an exact straight line.
- But we can fit a straight line to this data.
- Why would we do this? Once we do this, the y values for any other x values can be easily calculated for future predictions.



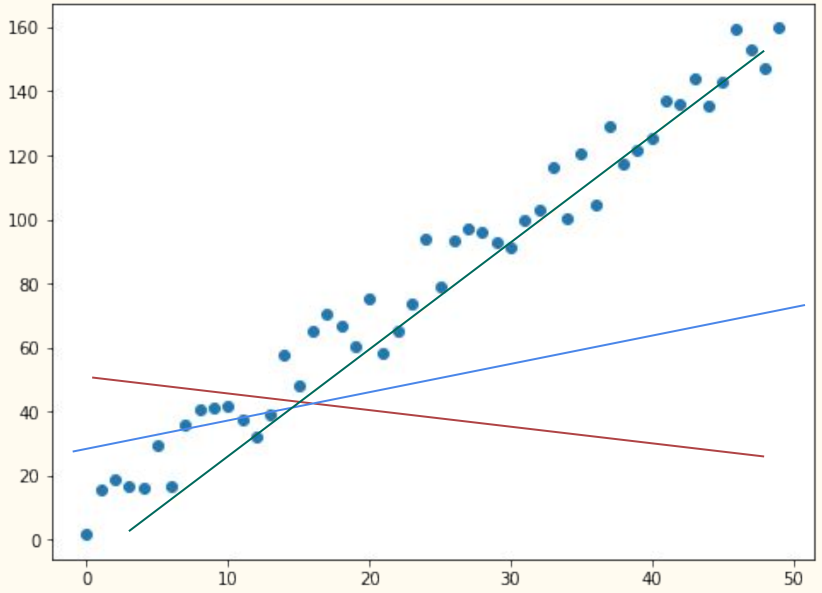
Linear Regression

- Each 'fit' is basically one configuration of the m and c values.
- Visually, the red line here is a bad fit, the blue line is relatively better and the green line is a good fit.
- What makes one fit better than another *mathematically*?
- We decide how good a fit (or a model configuration) is by looking at the **loss function** value.
- Lower the training loss, better the fit on the training data.
- The most common loss function for regression is **Mean Squared Error.**

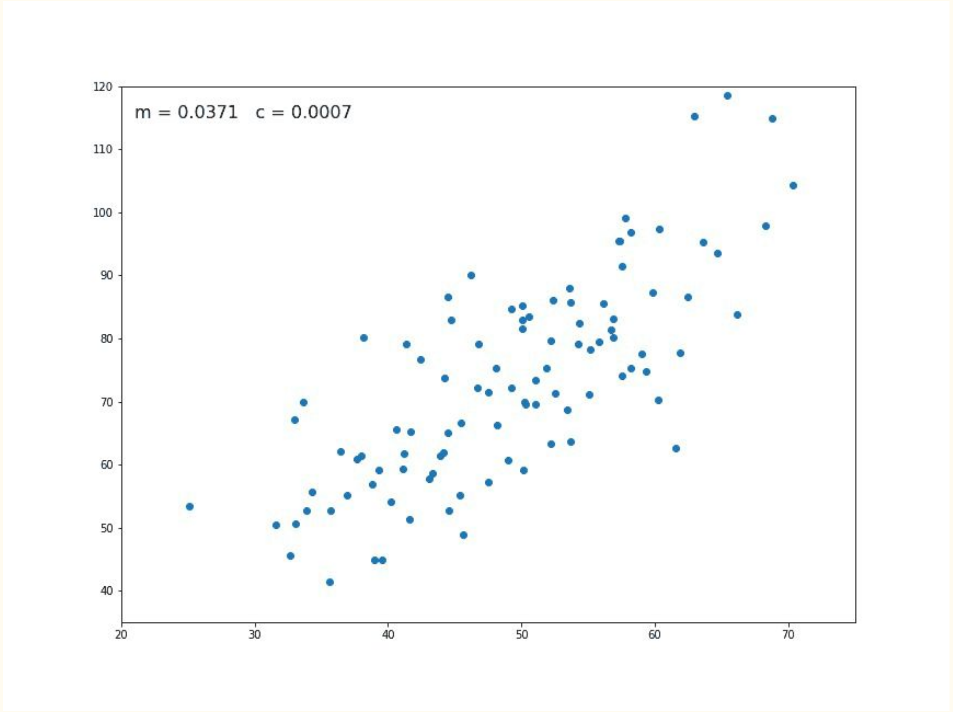


Linear Regression

- How do we make the computer get the best fit automatically?
- We start off with a random m and c value, then we iteratively improve it (change it so the loss gets lower and lower as time goes on)
- How do we know what optimal change to make that reduces the loss the most?
- This is where **optimizers** come in, they take a model and a dataset and tweak the model so as to minimize the given loss function.
- The most well known (family of) optimizers in ML is **Gradient Descent.**



Linear Regression


m = 0.0371   c = 0.0007
120
110
100
90
80
70
60
50
40
20
30
40
50
60
70