SAMARTH SHARMA

INTRODUCTION TO ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

PREFACE

Ok, so this is the part everyone skips! However it would be nice to read the preface as here we will discuss what this book is about! Firstly, this is the first book that I am writing so if I make any “author” mistakes, feel free to share with me. I will be happy to work more on that :)

Ok ok now lets talk about the book. The basic question occurs : Who is this book for ? Well this book is for both beginners and intermediate players in the field of Machine Learning. Some of the information may be known or may be absolutely new to you, but I do promise the readers something : It will definitely improve your confidence and knowledge in the Machine Learning World

I have seen many many complex books, which includes so so many complex mathematical formulas as well as ancient English definitions, which will even leave Shakespeare in a confused state. The above factors makes many people have a ML phobia

I have tried my best to explain the depths of ML in a fun manner in a readable format which will not leave you with a feeling of wonder thinking what the heck is ML ? I would like to thank

I do have to give you an important disclaimer. No matter from where you study ML, it does require two things : Passion and Patience. The subject is just like any living being, if you treat it with love and respect, it becomes more easy for you to understand it.

That being said, I do have a simple request : DO NOT RUSH INTO THIS BOOK. Reading everything with patience is the key to conquer this book.

Lastly I would like to thank

CH 1 - A BRIEF INTRODUCTION

WHAT'S MACHINE LEARNING ?

Now. We all have heard the term 'Machine Learning' time to time. Whether it is in a high tech conversion among friends or the fear of losing jobs to robots, Machine Learning is a definite discussion of the 21st century

So what exactly is Machine Learning ? Well what Machine Learning is nothing short of magic but with a certain logic behind it. Machine Learning is literally what the name is. It is the ability of a Machine learn to perform a certain task without explicitly being programmed. Here explicitly means that we are not making a code which a Machine will implement. No no no. We are making a Machine learn and use its learning experience for future scenarios

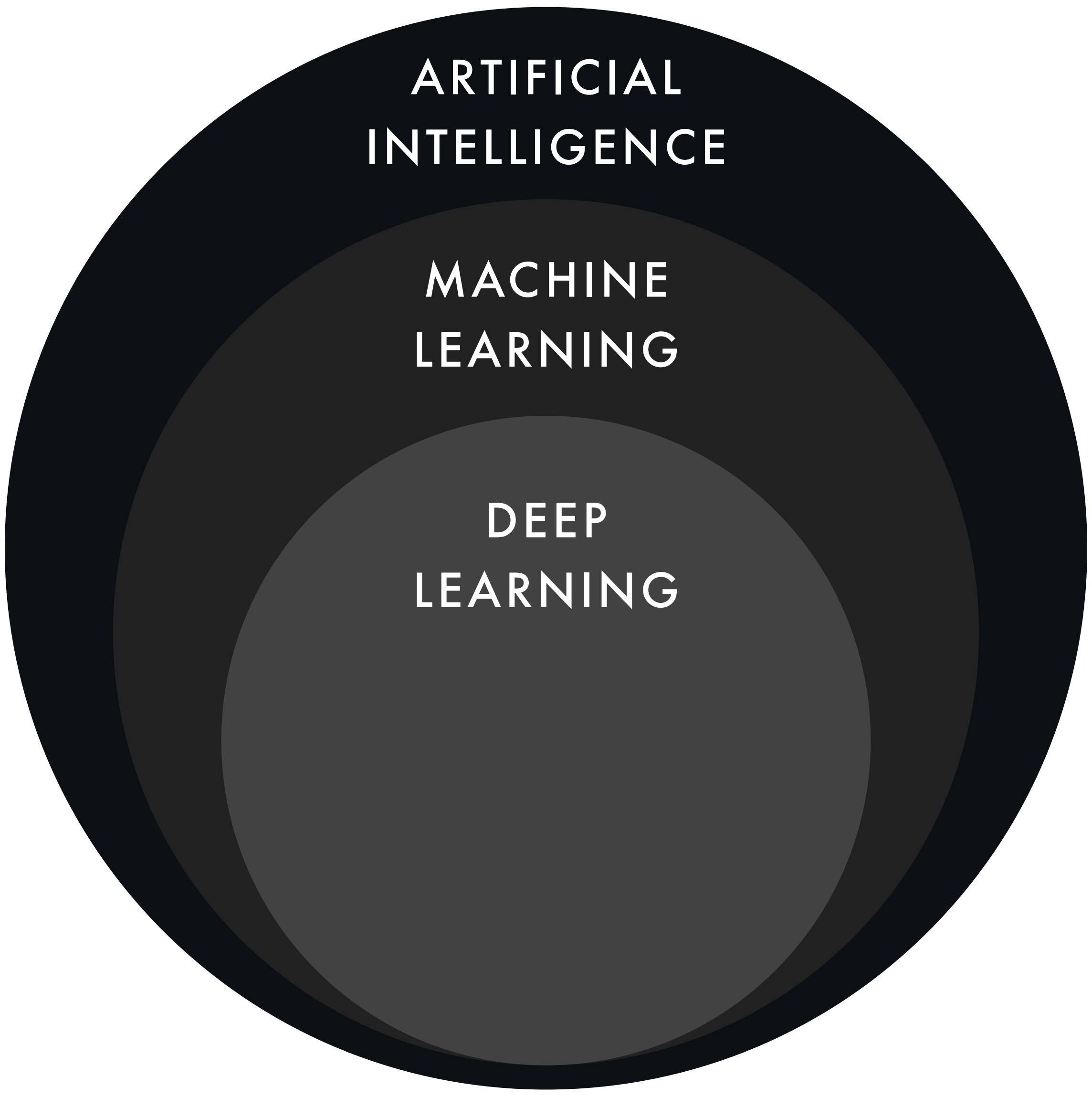
But if we want to formally define what Machine Learning is, I think the best way to do that is to have a look at the beautiful definition provided be IBM

According to IBM

MACHINE LEARNING IS A BRANCH OF ARTIFICIAL INTELLIGENCE (AI) AND COMPUTER SCIENCE WHICH FOCUSES ON THE USE OF DATA AND ALGORITHMS TO IMITATE THE WAY THAT HUMANS LEARN, GRADUALLY IMPROVING ITS ACCURACY.

Wait a minute…What do you mean by 'a branch of artificial intelligence' ? I thought that ML and AI is the same thing! Believe me, this is a very common question. Well the answer is No, as said above, Machine Learning is a branch of Artificial Intelligence. Do you know that even Machine Learning has a subset, and even though it is not very talked about, it is one of the leading research topics.

Lets have a look at a detailed structure in order to understand the relation between AI, ML and its subset as well



A VENN BREAKDOWN OF ARTIFICIAL INTELLIGENCE AND IT SUBSETS

THE ARTIFICIAL INTELLIGENCE BREAKDOWN

As seen the structure above, we have the Artificial Intelligence as the biggest category. Then we have Machine Learning as it's subset and finally we have a category called Deep Learning which is the lowest sub category but with the leading research topics

Before we move forward it is better to understand each of the categories so as to distinguish between them as from now on you would have an edge over others in debates and other discussions where you can use the appropriate term where suitable rather than use them in the air without truly understanding what AI, ML or DL is

• ARTIFICIAL INTELLIGENCE : THE PROUD PARENT

So as mentioned in the title Artificial Intelligence is the proud parent of Machine Learning and its subset Deep Learning. So the question is what exactly is Artificial Intelligence ?

Well according to the encyclopaedia giant Brittanica, :

ARTIFICIAL INTELLIGENCE (AI) IS THE ABILITY OF A DIGITAL COMPUTER OR COMPUTER-CONTROLLED ROBOT TO PERFORM TASKS COMMONLY ASSOCIATED WITH INTELLIGENT BEINGS.

In simpler terms you can say that Artificial Intelligence is the capability of a computer to perform assignments which were and still are usually performed by human beings due to their requisite of human intelligence or human judgement

In the modern world, the form of A.I. that we see is called weak A.I or more specifically ‘Narrow AI’. As the name suggest, Narrow AI is trained and used to perform specific tasks. A strong AI is made up of AGI and ASI, both which stand for Artificial General Intelligence and Artificial Super Intelligence respectively. AGI focuses on the current theoretical concept of machines possessing intelligence comparable to the current day humans while ASI believes in the concept of machines possessing an intelligence excelling that of humans

Strong AI is entirely a theoretical concept without backing from practical examples to this day but as we are progressing in the 21st century, those days are not too far where we finally have some practical demonstrations of strong AI

The question of its origin is often discussed by many people. What are the true roots of A.I ? Truth be told, it is still a matter of discussion when was the first AI discussed. Some believe it was John McCarthy, who have believed to coin the term ‘artificial intelligence’ and holding the first AI Conference in 1956 while some believe it is the Mathematical legend Alan Turing who saved the Britain and her allies in the Second World War by cracking the notorious German Enigma, then thought to be uncrackable.

But one thing is certain, no matter what the past is, AI is certainly a talk of future. Since the first talks in 1950s to more than 70 years later, we have seen many items which were never thought to exist, let alone possessing their own intelligence. These items have now become essentials of the day to day life. Smartphones, Laptops, Smart TVs and so many many more. In earlier days, caretakers would need to be hired to look over the house while you are sipping colas somewhere. Fast forward to present, we have smart assistants which take care of lightings, security, comfort and so on….

• MACHINE LEARNING : THE FAMOUS CHILD

Let's talk about Machine Learning now. As said in the title, Machine Learning is the famous child of Artificial Intelligence and currently the talk of the hour when it comes to learning skills in the A.I. industry.

Well as mentioned in the previous sections, Machine Learning is simply a branch of Artificial Intelligence which uses algorithms to mimic learning behaviour of humans. This has lead to many companies slowly adapting and evolving in terms of deploying Machine Learning in their products and services.

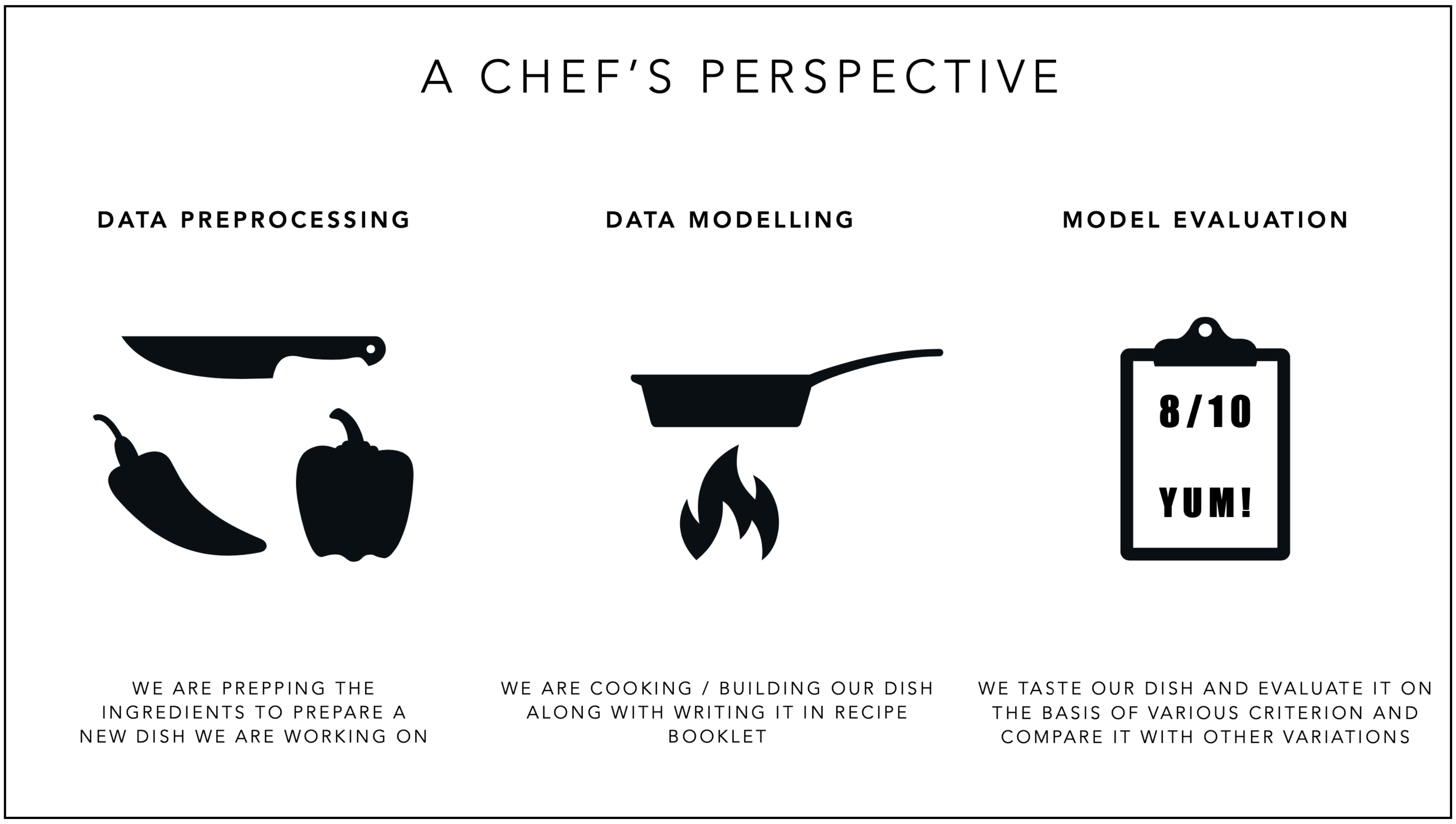
According to the London based IT giant Deloitte, a survey in 2020 reports that 67% of companies are using Machine Learning and 97% companies are planning to use it in the following year

Machine Learning basically involves three main steps :

1. **Data Preprocessing** : The world is full of data, so much so that many times it is not complete for Machine Learning algorithms to work on. Data Preprocessing is basically prepping the data to be used for algorithms as a reference to learn on It often involve cleaning of data, handling missing data, sorting out data for the upcoming steps, converting a data form into more suitable and accepted format

2. **Data Modelling** : This phase is what we call the 'fun part'. Here we take the data prepped in the Data Preprocessing phase and use it to train different kinds of models. Data Modelling involves building a model, training it and then using it to make predictions. Sometimes we also play with the boundaries and limits that we set for the models in order to work on them in the evaluation phase

3. **Model Evaluation** : This is the final phase and a crucial step where we evaluate the performance of our model. We use the various Performance Metrics such as a Confusion Matrices (explained later in the book) and then we make conclusions about our models. In this phase, we also compare the performances of other models built on the same dataset and find out the best model to deploy in the market



A CHEF'S PERSPECTIVE

DATA PREPROCESSING

WE ARE PREPPING THE INGREDIENTS TO PREPARE A NEW DISH WE ARE WORKING ON

DATA MODELLING

WE ARE COOKING / BUILDING OUR DISH ALONG WITH WRITING IT IN RECIPE BOOKLET

MODEL EVALUATION

WE TASTE OUR DISH AND EVALUATE IT ON THE BASIS OF VARIOUS CRITERION AND COMPARE IT WITH OTHER VARIATIONS

As mentioned before, many companies have begun on involving Machine Learning in their products or services. How many times do we say the following phrases 'Hey Siri' or for my Android gang out there 'Hey Google!'. Yes! Speech Recognition is a big part of Machine Learning, well, Natural Language Processing to be precise, but we won't discuss it in this book. Whenever you open your mails, how many times do you still see spam in your inbox ? We have really made you think back haven't we ? Well, that's all thanks to Machine Learning. Popular Email services such as Gmail use automatic Email Spam Filtering which already moves a suspected spam mail into the Spam folder. Well what about FaceID ? Guess what ? It's all Machine Learning, specifically Computer Vision. So, you see, Machine Learning is nearly all around you, making your life easy and more fun!

• DEEP LEARNING : YOUNGEST BUT EXPLOSIVE

Finally, let's talk about Deep Learning. It is a new player in the modern world, but has proven to be one of the most dynamic fields under the big hood of Artificial Intelligence. So what is exactly Deep Learning ? Well lads, once again we will seek our friend IBM here

According to the techies at IBM :

DEEP LEARNING IS A SUBSET OF MACHINE LEARNING, WHICH IS ESSENTIALLY A NEURAL NETWORK WITH THREE OR MORE LAYERS. THESE NEURAL NETWORKS ATTEMPT TO SIMULATE THE BEHAVIOUR OF THE HUMAN BRAIN

In simpler terms, you can say that Deep Learning is a form of Machine Learning that uses a computing model blueprinted on the inspirations of human brain, while calling for lesser human supervision. While the human brain has billions of neurone (cells that carry informations from brain to other body parts and vice versa), in Deep Learning, we deploy a system of Artificial Neural Network which work by impersonating the human mind and forming a connection between various neurone.

Now, I know you might be thinking, if Deep Learning is a subset of Machine Learning what exactly is the difference between between the two of them ? Well the main difference is the methods with which both parties operate.

Deep Learning removes some of the Data Preprocessing steps which usually happen in Machine Learning as unlike the latter, Deep Learning algorithms have the very much ability to work with unorganised data

Deep Learning algorithms work on a basis of 'picking up and working on the works of previous nodes'. There can be many layers of neurone present in a model. The input and output layers are called the visible layers and the middle layers are called the hidden layers. Then the model uses a process of back-propagation to calculate errors and then it self adjusts the coefficients of the mathematical equations to work better

The practical applications of Deep Learning is seen practically everywhere. Well, for starters, ever wondered how one of the biggest language translator works ? Yes, I am talking about Google Translate. Behind the scenes, the employee of the month is its powerful DL algorithm. How about Self Driving Cars ? How to achieve it ? Well one of the things that can be done is to use a Deep Learning network called Convolutional Neural Network or CNN (which is popularly used in Image Classification) and then use it along with other sensors to read and react to the road signs. The most common application is the recommender systems of the entertainment giant Netflix.

IS AI THE FUTURE ?

The answer : Absolutely! Even the world AI is now being associated with the upcoming generations. Don't believe me ? Well let's look at the statistics shall we ? According to the Stock Market giant Nasdaq, the global AI domain is to grow to USD 20 Billion by 2025.

As mentioned before, a Deloitte survey in 2020 claims that 67% of companies are using Machine Learning and 97% companies are planning to use it in the following year.

Many companies have reported that using AI have improved their customer experience, attracting as well as retaining a lot more customers (in business terms, we call it Customer Churn)

Using the computation power of modern computers, AI has substantially improved the decision making which has lead to the companies possessing an edge over their competitors while reduced costs have become a big attractor to companies

Well let's talk in terms of data. Up until 2005, we have cracked upto 130 Exabytes of data. If you are not tech savvy, well, here is another perspective. If the size of the Earth is 1 Gigabyte (10^9 bytes), then 1 Exabyte is the size of Sun (10^{18} bytes)

We all have seen self driven cars in futuristic movies. Well, now it's possible, thanks to Machine Learning and Artificial Intelligence. Tesla, Google, Apple, Mercedes Benz have all been heavily invested in Machine Learning and Computer Vision to make their auto-pilots much more accurate and safer.

Let's play a small game. I want you to imagine the first brand / services that comes in your mind when you think of an Over The Top (OTT) platform, where you can stream the popular shows. Now it's my turn to guess the platform. You are thinking of Netflix right ? (Hopefully, I am not wrong!). That's because, Netflix invests millions and millions of dollars in their recommender systems, which is one of the most accurate AI recommender out there. Let's play another round. This time I want you to think of a brand / service that comes in your mind when it comes to the music industry. My guess, Spotify (Once again, I hope I am right). That's because Spotify has the best AI model which makes customised music playlists in comparison to its competitors

Various modified versions of Reinforcement Learning (Branch of M.L.) are used by modern day e-commerce platforms such as Amazon, Walmart (through Walmart Labs) which allows these companies a strategic edge. There is a famous case study which used A.I to find that people who bought milk also bought bread. Using this, many supermarkets keep these two often together. Amazon uses its own algorithm to generate the perfect 'customers who bought this, also bought that..' which has allowed it to crush it's competition and be the part of the elite MAANG (For those who are not aware, MAANG is like a Holy Trinity for Computer Science Engineers. It comprises of the following firms : Meta (formerly Facebook), Apple, Amazon, Netflix and Google)

A.I has entered even our homes in forms of smart assistants which perform amazing tasks and all you have to do is just say the words. Companies such as Google, Amazon, Apple and a few have brought their OS assistants to physical world. Your wish is their command, all thanks to Natural Language Processing.

A study performed by Stanford University has allowed research to be performed in the detection of Breast Cancer on the basis of various criterions (we will not dive deep into these criterions). This has shown promising advancements of AI for the future of Medicine which allows more accurate decisions and in these cases, even life saving decisions

You must have seen this, as the OS of smartphones is upgrading (I am mainly talking about iOS and Android), the Photos / Gallery app automatically sorts your photos on the grounds of faces. Well thanks to Convolutional Neural Networks, these algorithm do the tiresome tasks of sorting images

Now I am about to tell you some awesome AI projects and research models out there. I also want you to check them out. Believe me, you won't regret it. We have many products such as Dall-E and ChatGPT from OpenAI (one of the most brilliant companies ever to exist on the grounds of AI). Or if you want to play a game, check out Google's Quick Draw. Want to play some music with AI, all you need to do is open MuseNet from OpenAI and start your own AI band. Ever wished that you could get an higher resolution of your blurred image, well, use the cutout.pro's AI Image Enhancer.

And all of these wonders were not even 'thinkable', a couple of decades ago. Every day, the AI expansion is no less than a revolution which has allowed itself to be an integrable parts of our future, if not life. The only limit is your imagination.

A BASIC INTRODUCTION TO OOPS

Just like everything in the technology world, ML is based on coding. Now to code, we use programming languages (languages specifically made to build computer applications and algorithms). Now technically there are many types of programming languages but we mainly need to focus on one of the most important type called 'Object Oriented Programming'.

Now Object Oriented Programming is basically a branch of programming languages which uses the concept of 'object' as their variables (containers that store information such as integers, floating values or decimals, characters, strings etc.).

Along with objects, we also have a concept of classes which are basically containers for the variables and features of the objects. For example, In the world of mammals, animals such as dogs, cats, whales etc. are object and the whole category 'mammals' acts as the class for the same problem

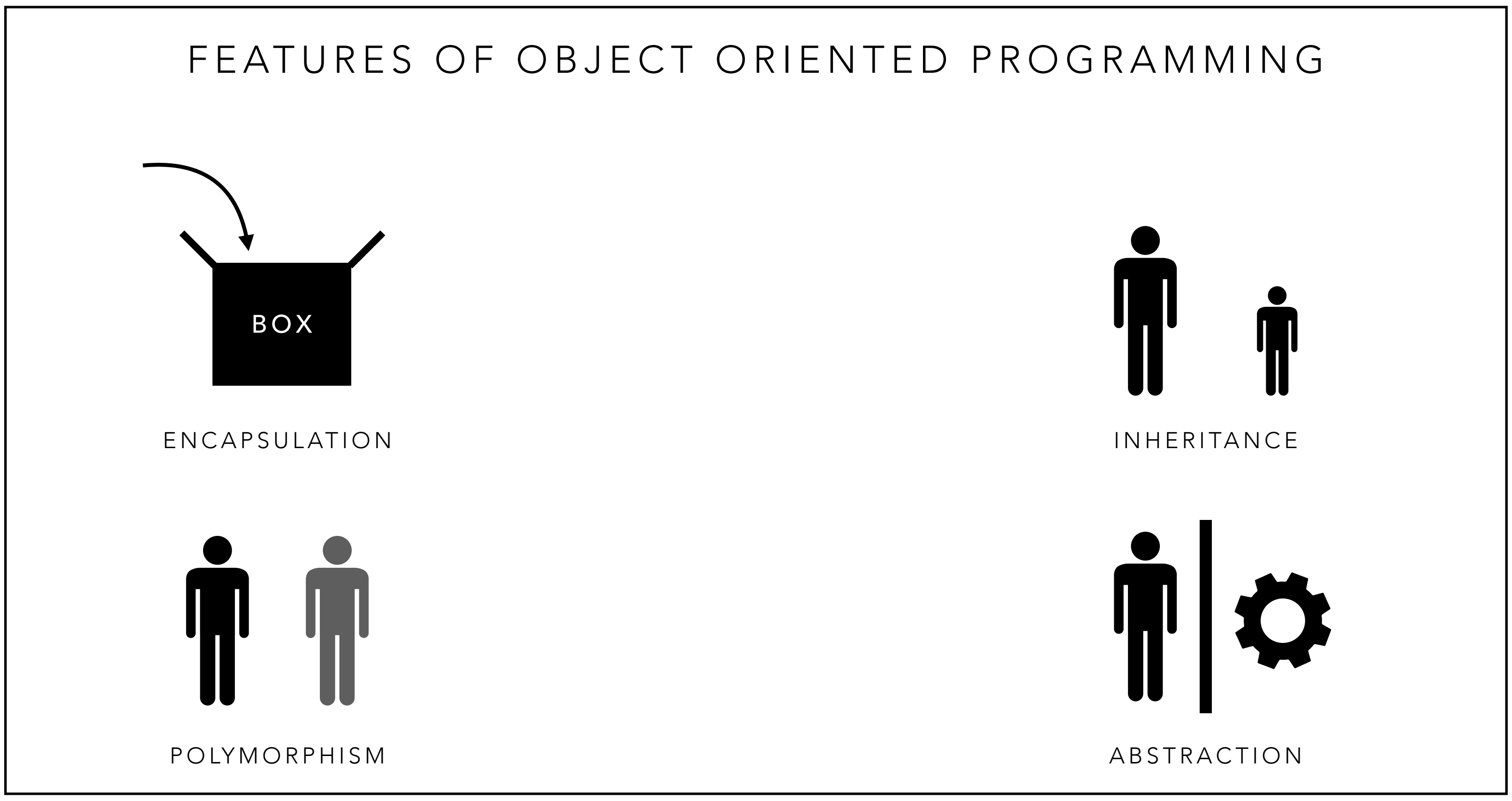
What basically makes OOPs different is that priority is given to the data and also provides a certain level of security in the form of access levels. There are mainly four features :

1. **Encapsulation** : In simple words, encapsulation is basically gift-wrapping all features and variables into a whole single box. In OOP languages, Encapsulation is done automatically and basically the main advantage of that is to hide the structured data for security. However it is much much different than Abstraction

2. **Abstraction** : Abstraction is providing vital features while excluding the irrelevant processes happening in the background. For example, lets say we are buying a packet of potato chips (also known as crisps) from a vending machine. Now we do not wonder what is happening in the mechanism with gears and stuff right? All we know is that we put in money, push some buttons and then take the packet and leave. So, Abstraction is like hiding the gears as we simply do not care with them

3. **Inheritance** : Just like in the real world, Inheritance is benefaction of the derived class (also called the child class) from its corresponding existing class (also called the parent class) in terms of their features. This allows us to reuse classes and have a much more clean and organised code. To understand it better, lets take an example. Let's take a trip down history. The first computer was made by the legendary 'Charles Babbage' in 1882 and it was a simple model which was used for complex calculations. But now the computers have upgraded a lot, we can play games, search the web, etc. but mainly we can still do its original feature that is calculation.

4. **Polymorphism** : Polymorphism is made of two words : 'Poly' which means multiple and 'Morphism' which means structure. If you combine the words then you automatically understand its meaning. Polymorphism in simple words basically means 'Same name, different game'. It is used to perform a task but with multiple response type, according to the different parameters. For example, for the function of adding, we can have two functions with different parameters (one which adds strictly integers and other that adds strictly decimals) but with the same name.



Note : For those who do not understand the example mentioned in Polymorphism, lemme try my best to explain something about coding world. We have different data types such as Integers, Floating Point (for decimals), Strings (sentences), Characters (single alphabets) etc. Now we cannot add two different types of data types as the output variable where you store the sum also has a data type. In non OOP languages, we would need to make two separate functions for addition where you use one when you only add integers and the other when you only add decimals. However, in OOP you only create a single function and have a modified version for the the other data type and the computer automatically figures out which function to use

So, now that we have discussed what OOPs are, lemme tell you why it is important. For most of the languages (except R) we have OOPs as when we build models, we need to import libraries and packages. So to access them you would need to create objects of these classes in the libraries and then use these objects to access the general features of that class and for that reason it is beneficial of having a small knowledge about OOPs

PSEUDO CODE

PROGRAMMING LANGUAGES FOR MACHINE LEARNING

Now that we have learned a little about OOPs, let's discuss some programming languages that you can use to code your Machine Learning models. Now there are many languages that exist for Machine Learning (some might even surprise you!). I will share a few of them which I find to be particularly useful.

Now I have to give a disclaimer here : In order to code Machine Learning, you DO NOT need to master all of the mentioned programming languages. Instead, it would be better to pickup a particular language (preferably a more famous one) and then code your models in it. All languages have their pros and cons (for example R does not support CNN).

That being said lets see the top 4 languages which are commonly associated in the coding world of Machine Learning. Once again, you do not need learn all of the languages but if you do that's great, but you can still be a ML champ with either of these only

A) PYTHON

Python is at the top of the Machine Learning world and for great reasons as well. Developed in 1991, Python is one of the newer languages but don't let that fool you. Despite a new player, it has managed to already dominate the coding world. According to StackOverflow, a reputed website for code discussions, Python is the only language that has been on the constant rise and by 2020, most of the questions that exist are from Python showing its popularity. Lets discuss why Python is the one of the best languages in the Machine Learning. Believe me, it will make you fall in love with it!

I. **Simplicity** : Python is one of the easiest language to understand so much so that it is literally one of the first language a coder learns to enter the world of coding. It is now even being taught at various unis and schools. The syntaxes in Python constitutes of English words which makes them memorable and easy to apply on your own. Consider Python is a very high level language (coding language that can be read and understood by humans), this feature gives Python an edge over other languages

II. **Independent** : Unlike many popular languages such as C and C++, Python is platform independent which basically means that Python can run on different platforms without undergoing any changes. For example in computers we have two types of compilers (machines that convert high level language to low level languages which machine understands). These are 32 bit and 64 bit compilers (bit is unit of storage in computers). Now the problem with C/C++ is that for different compilers, the storage required is different. For example an integer is 16 bits for a 32 bit compiler while it is 32 bits for a 64 bit compiler which is a huge problem however, Python does not have this issue which makes it platform independent

III. **Community** : There are many Machine Learning libraries and frameworks which contain multiple classes for various models as well other process that are essential for building great Machine Learning models. Some of these include :

   a) **Tensorflow :** Free Open Source software library from Google

   b) **Keras :** Free Open Source software which acts as interface for Tensorflow

   c) **Pandas :** Software for Data Manipulation and Analysis

   d) **SciPy :** Free Open Source software for scientific computation

   e) **NumPy :** Software used for working with arrays

   f) **Matplotlib :** Python Library used for data visualisation

   g) **Orange3 :** Open Source software for ML and Data Mining

   h) **PyTorch :** Python Library based on Torch library used for NLP and CV provided by Meta

   i) **Theano :** Python Library for mathematical evaluation

   j) **Scikit-Learn :** Free software for Machine Learning

IV. **Flexibility** : Python supports multiple types of programming including Object Oriented Programming which allows a variety of users to use Python to code their problems away such as in Web Development (Django is a popular framework), gaming (PyGame) and many more

B) JAVASCRIPT

Surprised ?! Well, I did tell you in the beginning that some languages may surprise you. We all know that JavaScript or JS for short dominates the world of Web Development, which brings more beauty in terms of animations as well as in terms of cleanliness. Many Machine Learning programmers pursue JS after studying Python due to its often involvement in Machine Learning programs (one of them includes Computer Vision!)

JavaScript is becoming so popular in Machine Learning that Tensorflow also created a JavaScript version of its more famous Python version. With Tensorflow.js, you can use ML directly in the browser as well as integrate the ML models with Node.js, which is another popular environment for JS. This functionality is slowly causing a rise of usage of JS in Machine Learning.

Let's look at some other advantages of JS for the Machine Learning world :

I. **Popular** : JavaScript is the second most popular language according to StackOverflow and is often seen being used in various domains which makes it a popular deal among various developers as you can achieve awesome Machine Learning models along with JavaScript's amazing features. JS has been here for some time due to which we already have many JS developers not only in the Web / App Development but other industries such as gaming and many many more…

II. **Fast and Secure** : Believe it or not, JavaScript is faster than Python which is a no brainer to it's advantage over Python all thanks to JS having an advanced OOP concept known as Multithreading. Along with speed, JS also commands another important qualities. Any guesses ? Well from the title you can tell its security. In JavaScript, the security is in-built, that is, JavaScript cannot access your files. Yes, I know languages like JS which are used in WebDev are easy to be cracked to weaken the sensitivity but platforms like Tensorflow.js have ensured to take care of it. These two features are more than enough to sway many ML developers and even working professionals to explore JS as a secondary language for their careers

III. **Model Fusion with Web and Mobile Applications** : Since JS is used practically in every web and mobile applications, JS has a lead over Python where Python still needs to develop better. However not all JS libraries support this as React Native (version of React strictly for mobile applications), however fear not because your prayers have been answered by the legendary people at Google, who have released a special version of Tensorflow.js for it's integration with React Native

Twitter has used JS to make their 'content recommender system' which is basically a recommender system but the purpose is to rank the content tweets in the order of relevance. Definitely read more about it on the Tensorflow website. It is definitely a great website to get amazing knowledge brought to you by Google themselves

On the Tensorflow.js website, there are many AI projects for the demo, feel free to check them out. Remember, Tensorflow.js is relatively new in comparison to its more famous Python brother, but it does not mean it is not useful at all!

JS is definitely a great language to pursue for the Machine Learning along with Python and a few others. We are just getting into it people. Lets see what other languages are popular for the Machine Learning domain.

C) JULIA

What ? I have heard of Python and JavaScript but what the hell is Julia ? These were the questions asked when I was researching on an article by the amazing people at SpringBoard (online platform on career development, check them out!). But when I read a little more about Julia through different articles I am glad I discovered Julia. Frankly it is the next language that I learnt after Python for ML. Even though it is relevantly new, with it's age same as literally a country (Fun Fact : Just like Julia, South Sudan was created in 2012, making it the youngest country to exist till this date), Julia is the next big thing

I. **High Performance** : Julia is often associated with the word 'Fast', thanks to something what people are calling Just-In-Time compiler or JIC for short. What does JIC do ? Well according to Wikipedia, JIC is a way to execute computer code with compiling during execution of a program during it's runtime rather than before its execution (usually the code is compiler before the execution, but run-time compilation is much much faster)

II. **Easy to Learn** : Contrary to the popular belief among programmers, Julia is absolutely not a difficult language to master. Even though beginners usually prefer Python over Julia due to it's popularity (as I told you, I didn't even knew about Julia until I was in my undergraduate college), Julia is also an awesome alternative as it has some advantages over Python (for example, as mentioned, its rocket speed!)

III. **It's in their Genes** : Julia was specifically made for mathematical and scientific research which is the core of most of the Machine Learning algorithm. Moreover since Machine Learning involves complex equations along with large dataset which demands high computational power, which may take hours and even days. Fortunately for us, Julia equals Fast! With this, we can say it's in their genes for their application in the Machine Learning world

IV. **Developing Packages** : Even though it's practically just a child, Julia has already many packages that contain different classes and libraries for Machine Learning. Even Julia's official website julialang.org, mentions these packages such as MLJ.jl which has some common Machine Learning algorithm. For Deep Learning, we got Flux.jl and Knet.jl and so on.

Not only that, just like we have DataScience libraries such as Pandas, Numpy for Python, we also have packages such as Dataframes.jl to work with datasets. In fact many libraries platform such as Tensorflow (yep, you will hear a lot of it in the upcoming chapters), Sci-Kit learn, Spark and a few utilise Julia's features and blessings for their Machine Learning models

Fun Fact!

The packages in Julia are written in Julia itself! unlike packages in Numpy, which have their inner kernel made of a faster language such as C / C++

Due to its various advantages in Machine Learning, many reputed companies, even government firms incorporate Julia in their work. National Aeronautics and Space Administration or NASA for short, Federal Aviation Administration (FAA), Federal Bank of New York are a few government firms. In the tech world, we got Apple, Disney, Oracle, BlackRock and many many more.

D) R

R is another popular language along with Python for Machine Learning. Developed by the tech famous Bell Laboratories (formerly AT&T Bell Laboratories and now Nokia Bell Libraries). It's was originally developed for statistical computing but has now gained more traction in the Data Science and Machine Learning community. Although R is a little more difficult than Python but it has the upper ground when we compare the computation power. Let's take a look at the other features of R which allows it to be in top languages for Machine Learning

I. **Simple Technique** : Just like Python, R has simple syntaxes. Moreover people who already have an experience with famous languages such as Java or C++, R might even be better to understand in comparison to Python. Since people who do R have some knowledge and experience in statistical world of mathematics

II. **Awesome Packages**: Due to it's originality for data analysis, R already has an extravagant libraries and tools which can help to boost the performances of models by performing tasks such as preprocessing as well as understand the data in order to figure out the best model that can work through data visualisations packages such as Shiny or Markdown

III. **Open Source** : Like Python, R is also an open source programming language, this means that the only thing you need to work in R is an interest in R itself. Unlike Julia, which has it's licensing provided by MIT (Massachusetts Institute of Technology) Licensing, Python and R do not need any license or permit to work with

NOTE

For the sake of simplicity, we will use Python throughout this book for the libraries and other syntaxes under the assumption, many people know about Python. No Worries Python is an easy language to understand and learn and would take almost no time to learn it, thanks to it's syntaxes (commands) being similar to the words of similar meanings in the English Lexicon

THE TRIO OF TOOLS FOR MACHINE LEARNING

In this sub section, we will have a look at some amazing tools and websites which are definitely useful for Machine Learning. Here, we will not discuss the libraries that we often use for Machine Learning as that would be discussed in Chapter II but basically, we can use these libraries for Machine Learning.

A) ANACONDA

Anaconda, as their website claims it, is the 'World's Most Popular Data Science Platform'. And rightfully, so! Anaconda has many products depending on the usage such as personal or for business, each version having it's own advantages and pricing.

As mentioned by the legends at Anaconda themselves, you can use Anaconda for :

- Neural Network
- Machine Learning
- Data Visualisation .....

Anaconda also comes with Jupyter (which we will discuss next) and has in built libraries such as Pandas, Numpy, Matplotlib etc. along with some other tools such as PyCharm, Spyder, Orange3 etc, which are most commonly used for Data Science and other applications of Machine Learning.

These above mentioned features have ensured Anaconda to continuously being a great assistance for Data Science and Machine Learning programmers. One of the amazing things that contribute to popularity of Anaconda is the large library of software it comes with, having over 200 packages and 5000+ additional packages available for download, Anaconda is sure bound to be favourite

B) JUPYTER

Project Jupyter or simply known as Jupyter is an Open Source software for interactive computing over 40 language (as claimed by their website). A fun fact! The name Jupyter comes from the combination of three programming languages which act as its core programming languages. Any guesses ? Well, if you just wanna jump to the answer, here it is : Jupyter is combination of Julia, Python and R. Well another amazing thing about Jupyter is that one of the languages I mentioned previously was involved in the implementation of its notebook. Once again, if you are not in a mood for games, the correct answer is JavaScript

You can download Jupyter separately or as a part of Anaconda installation where you would need to separately import packages such as Pandas, Numpy etc. manually with Python commands in the former while all the most user packages are installed while Jupyter is being installed through the latter method being installed through the latter method

Jupyter is a common sight across Data Scientists and Machine Learning developers and is definitely a useful tool to use for more on the offline side of the work in comparison to it's online competitor : Google Colab

C) GOOGLE COLAB

Google Colab is a product of Google Research team that is based on Jupyter and is one of the best in the market. There are many advantages of using Colab over Jupyter, but I would be mainly discussing the two biggest advantages (in my opinion).

Firstly, you will not need to download any packages such as Pandas, Numpy, Tensorflow, NLTK etc as they are pre-installed in Colab files. Secondly, Google Colab offers free GPU (Graphical Processing Units) which leads to faster execution of large datasets and model training in Deep Learning which are so computationally heavy that even though Deep Learning has theoretically been there for 50+ years but was limited towards the practical side due to being 'too advanced' for the technology at that time (which has improved a lot since then)

Unfortunately till the time of writing this book, Google Colab only supports Python and with its limited GPU for free users (you can buy more in their ColabPro subscription). Well an ironic fact about Google Colab is that it's strength is the reason for its weakness. Due to the functionality of cloud sharing as one of it's strength, it leads to the issue of data security

Keep in mind, Both Jupyter and Colab allows you to create interactive python notebooks with each having it's own advantages and disadvantages and both are relevant in the Data Science and Machine Learning world.

TRIO OF MACHINE LEARNING TOOLS

ANACONDA JUPYTER COLAB

Now I know the question you are about to ask me and don't worry, it is one of the most popular questions in the world of Machine Learning. You might be wondering "When should I use Jupyter and when should I use Colab ?".

Well, hopefully the next sub heading would solve that query!

JUPYTER V/S GOOGLE COLAB

Jupyter, as mentioned in the previous section, saves the files in hard disk, being a running application on a local machines. Although it may sounds like a bad option, but in the world of Data Science, we work with very very large data (I mean we are talking about thousands of thousands of records with hundreds of parameters)

However another person that wishes to open a Jupyter Notebook, needs to have Jupyter installed in their system, along with the used libraries. This is a huge bane for low RAM devices as it consumes a lot of hard disk space. Moreover for applications involving Deep Learning, it is important to utilise GPU cores for faster results. This is where Jupyter is useful in situations where you have GPU cores in your hardware (usually found in Gaming Laptops and PCs)

Jupyter might be the best option for you if you want

- **Large Datasets**
- **Personal Projects / Notebooks**
- **Sensitive Data**
- **Data Security**

The above points make Jupyter an amazing resource and these should be considered while selecting your desired platform of work

However, Jupyter does have the following disadvantages which must also be considered before electing Jupyter as a winning candidate for your project

- **Notebook Synchronisation (constantly need to check if all hosts have same notebook)**
- **Accessibility to GPUs (for faster results)**
- **Team Projects (lack of notebook sharing)**

Google Colab, on the other hand, is cloud based which results in automatic backup and synchronisation in your Google Cloud. Colab files can be opened in multiple devices and has that added advantage of sharing files. Moreover we have an access to free GPUs, which adds an extra layer of speed in the calculations.

You may be inclined towards Google Colab if you are looking for :

- **Free GPUs**
- **Notebook Synchronisation**
- **Team Sharing**
- **Ease of Accessibility**
- **Small Datasets**

On a slight disadvantage, Colab loses the fight against Jupyter for the following cases which you should also consider before accepting Google as the king of all solutions

- **Partially free (after some time you would need to purchase further GPUs and TPUs)**
- **Limited Runtime (between 12 - 24 hours)**
- **Data Sensitivity (Secure but less privacy in comparison to Jupyter)**
- **Large Datasets (a single large file can exhaust all of your GPUs)**

CH 2 - IT’S ALL ABOUT DATA

THE MACHINE NUTSHELL

A Machine Learning model is nothing but an object of various Machine Learning algorithms (which we will discuss later in this chapter) along with some parameters on the basis of which the model trains on the data and then makes predictions on the basis of its learning.

There are mainly three phases involved in a typical Machine Learning model. Please note that the terms Alpha, Beta and Charlie in this book are used by me, it is not a standard notation to call them these (I just found them cool!)

A) PHASE ALPHA : This phase is also known as **Data Preprocessing**, which involves the following tasks such as

- Data Import (importing the data)
- Data Cleaning (to be explained later)
- Data Splitting (to be explained later)
- Feature Scaling (to be explained later)

In a nutshell, we can say that Data Preprocessing is preparing the data for the Machine Learning model to train on and find the useful connections for successful predictions

B) PHASE BETA : This phase is also known as **Data Modelling**, which involves the following tasks such as

- Building the Models using various Algorithms and Parameters
- Training the Models on the Preprocessed Data
- Making Predictions on new Data

In a nutshell, we can say that Data Modelling is the construction and training of the Machine Learning model to make predictions based in the preprocessed data

C) PHASE CHARLIE : This phase is also known as **Data Evaluation**, which involves the following tasks such as

- Using Statistics of Machine Learning to find some results
- Comparing different models and finding the best model

In this chapter, we would be focusing mostly on Data Preprocessing and a bit on Data Evaluation along with some in-depth knowledge on various libraries commonly used in Machine Learning

Keep in mind that when we say the best model, we refer not only to the algorithms but also the parameters on which we "bound" our model to stay. We can find the best combinations of parameters and algorithm through various methods, mentioned in **later Chapter**

NOTE

To understand the phases even better, I highly recommend having a look at the diagram under the Machine Learning part of the subsection 'The Artificial Intelligence Breakdown' in Chapter 1 : A Brief Introduction

DATASETS : HOW DATA IS STORED ?

Datasets are N-Dimensional Matrices where your Data is stored (in common terminology, we store data in form of tables) with every column corresponding to a separate dimension. Attendance Sheets in Schools and Universities, Salaries of employees in a Company, Leaderboard of a championship, etc. All of the above examples are cases of Datasets

Now just like every other item in our computers, we need to store our Datasets with an extension (an extension is a short form which is used to represent the datatype of an item stored in a computer. For example images are often stored in .PNG for Portable Network Graphics or .JPEG for Joint Photographic Experts Group)

We have the following extensions in which Datasets are more commonly stored in computers. You may have used some of them yourselves for your projects or maybe even heard of them. Let's have a look at few of them :

A) CSV (COMMA SEPARATED VALUES)

This is one of the most commonly used extension for Machine Learning Datasets, thanks to its simplicity. As the name suggests, the records (values) are separated by commas (in case of columns) and line breaks (in case of rows). One of the amazing thing about CSV is that it is designed in such a way that you can open the files in a normal text editor (for example Notepad in Windows or TextEdit in MacOs)

CSV files are not only used in Machine Learning, but is a common sight in Web Development, Research Gathering, Databases / Servers etc. One of the key players in its properties which contributes to the popularity of CSVs is that CSVs can be opened by any Spreadsheet applications such as Microsoft Excel or Numbers. You can even open the CSV documents in Database applications such as MySQL, PostgreSQL etc. However we must also look at the fact that CSV has less compatibility and scalability than other formats like JSON

B) XLSX (MICROSOFT EXCEL FILE)

We all have used or at least heard about Microsoft Excel, a spreadsheet software developed by Microsoft and plays a huge role in business and data handling. One of the advantages which is also a disadvantage (will see later) regarding Microsoft Excel is the usage of concept of sheets, which allows us to store different types of data in a single Excel File, thus utilising the space and storage capacity in a brilliant manner.

Besides in Machine Learning, Microsoft Excel files are also used in Time Management, Financial Structuring, Accounting and so on… However, the disadvantage of using Microsoft Excel is the lack of key services applied in Machine Learning world, along with a difficult syntax (along with file name we also mention the sheet name, to be seen later), which does not adds to a good image in terms of useful machine learning tool, but hey, if you are an expert in Microsoft Excel, you can also create Machine Learning models without coding!

C) JSON (JAVASCRIPT OBJECT NOTATION)

Ok now, JSON is simply a data format which in a nutshell, stores data in a format which is readable in many different languages and platforms. Other key advantages of JSON includes it's human readability, light weightiness (requires less CPU resources) and fast processing.. JSON is a dual presenter of data, in the form of Objects (check out more information on what object is in Ch 1) and Arrays (like a list of items, for example : your grocery list)

A fun fact about JSON is that despite it has JavaScript in it's name, JSON, in reality, is independent of any programming language. Another important fact to remember regarding JSON's role in Machine Learning world is that most of the data stored in the world is in the forms of JSON which also allows JSON to be a nice option for Complex Datasets. However, just like the advantages we have disadvantages in JSON as well. JSON files are not suitable for Large Datasets and also suffer from less security and slower processing speeds.

NOTE

There are various other softwares such as MongoDB, SQLite, BigQuery etc. However, this book is written by considering a beginner's perspective due to which, I will only be discussing the most commonly used dataset format which is CSV.

A DEEPER LOOK INTO DATASETS

Now that we have discussed some dataset formats, lets dive deep into how exactly our dataset looks so that we can have an introduction to two new terminologies which are commonly used in the Machine Learning world.

So without further ado, lets see what our datasets look like !

PRICES (IN USD)	**AREA** (IN METERS²)	**BEDROOMS**	**LOCATION** (NEAREST CITY)	**AGE** (IN YEARS)
5,60,000	2000	2	SEATTLE	3
2,80,00,000	5700	4	SAN FRANCISCO	2
6,90,000			AUSTIN	3
1,30,00,000	4800	3	SAN FRANCISCO	4
95,000	1400		MONCTON	10
	2960			

A SAMPLE DATA SET OF PRICES OF HOUSES BASED ON THE AREA, NUMBER OF BEDROOMS, LOCATION AND AGE OF THE HOUSE

Our datasets typically look like something from above. So how do we use it ? Well, as a first step we divide our dataset in the following two categories, pay attention, cause this is important.

A) Features (X) : As the name suggests, this contains data fields that acts as features or properties on the consideration of which, we get the final output. In the above dataset, our features include Area (bigger houses may be more expensive), Bedrooms (more bedrooms mean more expensive), Location (popular cities may have more expensive houses) and finally Age (older houses may be less expensive unless they are too old to have history behind them such as castles or inns)

Features are also known as Independent variables, this is because we often assume a single feature to be independent of any other feature though this might not always be true. The main usage of features is to train the model and find some sort of pattern or relationship between the housing prices and different features

B) Target (y) : Once again as the name suggest, these are the values we often target to find. In the above dataset, our target variable is 'Prices' as we want to find to use the different features to find what should be the price of the house. Target is also known as the Dependent variable as it is dependent on different features for its values. Notice when I said variable and not variables, this is because in most of the situations we have a single column for our target while we can have more than one columns for our features. We also call the Target variable as 'Label'

NOTE

The dataset mentioned here may be different to what you receive in terms of the positioning of target and features, as it is highly possible for your target column be the last column of the dataset unlike the first one mentioned here. This is completely fine and does not change anything. However we do need to use the same notations throughout (X : Features and y : Target) as these are the common notations you will find while exploring someone's code

Now before we start jumping into model construction, we first need to understand a couple of crucial facts about Machine Learning models in general

I) Computers only understand Numeric Values (as it is build on binary numbers 0 and 1)

II) A model becomes more accurate as it gets more and more data

III) A model is more accurate if it's values are more or less in a similar range

On diving deeper into our model, we have mainly three issues which are generally common to a typical Machine Learning dataset. Do not worry! We have the solutions to all these issues which must be tackled in order to achieve maximum accuracy

A) In 'Location' we have non numeric values as we have Strings as values (check out Ch - 1 on more information on different data types). This is an issue because we want our machine model to consider all available data

B) In some columns you might notice, we have some of the values missing, do not worry, this is rather a common feature in large datasets as typically we have a hundreds of thousands of data (for example : In a typical area zone, we are bound to have at least 50K houses). In data collection, it is practically near to impossible to have 100% data

C) The 'Area' column in our dataset has a large values (inc ranges of 1000s) in comparison to rest of the numerical values (in ranges of 1s and 10s). This may cause our machine learning model to become biased towards a particular

In the next section, we will be exploring the solutions to the above mentioned problems. So let's check them out and come one step closer towards building the models on the fully prepared data

THE DATASET SOLUTIONS

Now that we know about potential problems that exist in our Dataset, its time that we resolve them. We will first start by addressing the issue of having string in our data (also called categorical data), after we have solved it, we will then solve the problem of missing values and finally we will solve the problem of unscaled data (unscaled data is data where we have many values in different ranges). After we have prepped our dataset for building the model, we will look at dividing the data for our advantage

Note that it is important we solve our issues in a particular order as the next stage is dependent on the previous one so always remember the order :

MISSING → CATEGORICAL → FEATURE SCALING

Without any more wait, ladies and gentlemen lets solve our first issue !

A) MISSING VALUES

Missing values are common and can happen due to many reasons, such as human error, or the subject (person who is giving the data) does not want to respond or some other factors. Nevertheless, this issue needs to be dealt with

In Machine Learning we have two options to deal with the situations of missing values we can either remove the missing records or replace the missing values. Now the question comes, when should I remove the records and when should I replace the values. Also what would I replace the missing values with is another question that might pop in the head. The answer to the all the questions above is 'It Depends on the Situation'. Lets see the below situation in which we perform the operations

• DELETING THE RECORDS

It is a quick but a desperate solution of handling missing values. We can do this in cases of a small fraction of missing records in a large dataset. This method works well in cases where, for example, we have 1% of the total data missing as removing the records will not affect the model that much.

The problem with this solution is that 'Data is Precious' and not only it is a real pain collecting data and loosing it seems like a waste of time, we might also lose some important information. Moreover the solution is completely useless in cases of large missing values.

• REPLACING THE VALUES

Well, yes, but the proper terminology is 'Imputing Values', but in a nutshell, yes we will replace the values. This is a better solution as it not involves discarding our precious data. We can replace the missing values with a variety of options the most commonly used options are replacing with Mean, Median, Mode and Forward Fill. Lets have a closer look at all these methods and when to use them

A) REPLACEMENT WITH MEAN

Mean or more commonly known as average and is calculated as sum of the numbers divided by the number of numeric values. For example, if we have 5 boxes contain candies in the amounts of 5, 6, 7, 8 and 9 candies spread across the 5 boxes, then the average candies in each box is (5 + 6 +7 + 8 + 9) / 5 = 35 / 5 = 7 candies in an average.This is one of the most used method where we replace the missing values with the mean of the rest of the records This way we can prevent data loss and is very useful on smaller datasets, however, since mean is a mathematical term, it only works on numeric data.

B) REPLACEMENT WITH MEDIAN

Median is the value of the centre element in a sorted list. The formula of median is bit more complex than that of Mean. In case we have even number of elements, the centre element is the average of the (n/2)th and (n/2 + 1)th element. For example in a list of 6 elements : 34, 53, 12, 6, 67 and 45 has the median value 39.5 as on sorting the list first (we get 6, 12, 34, 45, 53, 67) and we have 3rd and 4th element as 34 and 45. The average of 3rd and 4th element is 39.5. In the case of odd elements, the median is the ((n + 1) / 2)th element. For example in the case of 5 elements : 34, 53, 12, 6, 67, we have the median as 34 because on sorting the list we have 6, 12, 34, 53 and 67 with the median being 34 as the 3rd element. This is also one of the most replaced values along with mean and has the similar advantages such as working only with numeric values and prevention of data loss

C) REPLACEMENT WITH MODE

Mode is basically the most frequently occurring value in a list of items. For example in a list of 20 items as 1,1,1,2,2,2,2,3,4,4,4,4,4,4,4,5,5,5,2,2 we have the number and frequencies stored as (1, 3)| (2, 6) | (3, 1) | (4, 7) | (5, 3) and from these records it is clear that the mode is number 4. There is no formula for finding Mode and the best way to find the mode is to create a small table of values and their corresponding frequency and select the record with the highest number in the frequency section. We often use this on Categorical Data (we dive deeper into categorical data in the next section) which has an added advantage of not converting the values into numeric and as usual it prevents data loss.

D) REPLACEMENT WITH FORWARD FILL

In some situations, forwarding the previous value to the missing value is a suitable replacement. This process is known as forward fill. It is mostly used in cases where we know that the previous data would be very similar to the missing record if it was not empty. We usually call these Time Series Analysing situations. One of the examples to relate is the weather. We know that if the weather of the previous date was 25°C (or 77°F if you prefer that) then the weather of the next day is most likely to be near to 25°Celsius so it makes more logical to put the value of 25°C as the missing value as well

Similar to Forward Fill we also have Backward Fill in which instead of the previous value, we replace the missing value with the value succeeding the missing record. However in this book we mostly would need to deal with replacement with mean. Now we have plenty of more solutions for handling missing value but in order to keep the book beginners friendly, I have only discussed most commonly used replacement strategies but feel free to check all the options out. There also some algorithms which ignore missing values (discussed later)

Congratulations! We have official solved 1 out of 3 problems. Next we are going to take a look at the Categorical Values and the problem of String!

B) HANDLING CATEGORICAL VALUES

Now the issue of categorical values is that we have String values in our Numerical Data, due to which our model cannot read the data completely. Well, if the model cannot read String data then we will convert our String Data into Numerical Data, that way the model will consider all data. But how can we do that. Well, lets play a game of pattern finding. Tell me what is the pattern in the following text : A(1) → B(2) → C(3) → D(4) and so on..

Well I really hope you have found the pattern, but if not, no worries, it is a pattern of Alphabets matched to their position in the English Lexicon. Based on this pattern, if I tell you to convert the number 89 into the English word, I am sure you will get the output as 'HI' as according to the pattern, H is 8 and I is 9. This concept is known as encoding where we convert one form of data into a different form in order to make it more usable. In this case we are converting our 'String' data into a more useable 'Numeric' data.

Now that we see what we will do, we are going to see the two variations in which we can perform categorical encoding. We mainly call them OneHotEncoding and LabelEncoding. Now I know what you could be thinking, What is the difference and when to use it ? Well lets see the answers to these question because the difference is a simple but a crucial one!

• ONE HOT ENCODING

In One Hot Encoding, for each unique String we have, we create a separate column and enter the value 1 if the String is associated to that column and 0 otherwise. The below example shows the output of One Hot Encoding on a String column containing Fruits

FRUITS	APPLE	BANANA	ORANGE
APPLE	1	0	0
BANANA	0	1	0
ORANGE	0	0	1
BANANA	0	1	0

Now we perform One Hot Encoding is usually done in order to avoid unnecessary or non existing patterns to emerge out as not only these are useless, but it also leads to the model generating false outputs in many cases (we will look at what we mean by false assumptions later in this section)

The major drawback of One Hot Encoding is the fact that it leads to an increase of dimensionality (in simple terms, the drawback is that it increases the number of columns). Now, in case you are not aware of the concept of dimensionality, every column of dataset corresponds to a dimension. Humans can only visualise upto 3 Column dataset in which we will have 2 input features on two of the three axis of XYZ, while the output label is on the last feature. One thing to remember is that with every new dimension added, the cost increases by a huge amount, which we all know is never a good thing

• LABEL ENCODING

In Label Encoding, we make the changes in the same column itself instead of creating new column. As the name suggest, it is mostly performed on the Label (Target) column but we can also use it in other cases such as ordinality (if we have the strings in some hierarchal order). The below example shows the case of label encoding

POSITION IN ARMY (US)	CODE FOR US ARMY POSITION
PRIVATE	0
SERGEANT	2
CORPORAL	1

In the above example case we have the hierarchal order of rankings in the United States Army, we can clearly see there is some sort of hierarchy. Another great example case is the order of education where we have the increasing order, starting from Elementary School, all the way to Ph.Ds. We also use Label Encoding in case we have a large number of features which is an advantage in comparison to the storage issues of One Hot Encoding

The possible drawback of using Label Encoding is the situation of False Assumptions. Now I know what you are wondering, what is it with the False Assumptions. What is this great 'False Assumption' situation which has caused to give a lot more caution towards using a method. Well, lets have a look at it shall we ?

• FALSE ASSUMPTIONS !

In the below sample dataset we have various car companies and the number of models they sold of the colour 'White'. We know that there is no hierarchy structure existing between these companies so as per the rules, One Hot Encoding seems like a better option than LabelEncoding. But lets say someone working with this goes 'Hey, rules are made to be broken!' And proceeds to apply Label Encoding. In that case we have

BRAND	BRAND CODE	ITEMS SALES
ASTON MARTIN	0	1000
MERCEDES	2	3000
FORD	1	2000
TESLA	4	5000
SUBARU	3	4000

Notice something, a pattern perhaps ? Yes! We have found a pattern that as the brand code increases, the number of cars sold increases. Now since ether brand coding is done in an alphabetical order by default, This is a problem. Why ? Because we know that there is no such pattern that exist, but the model in fact, created a pattern. So apparently, the person would claim that if you name your company by a latter member of the alphabet series, it would see more success, even if the cars are not good. In cases of large amount of features, it becomes near to impossible to identify this problem and thus I present to you, the False Assumption

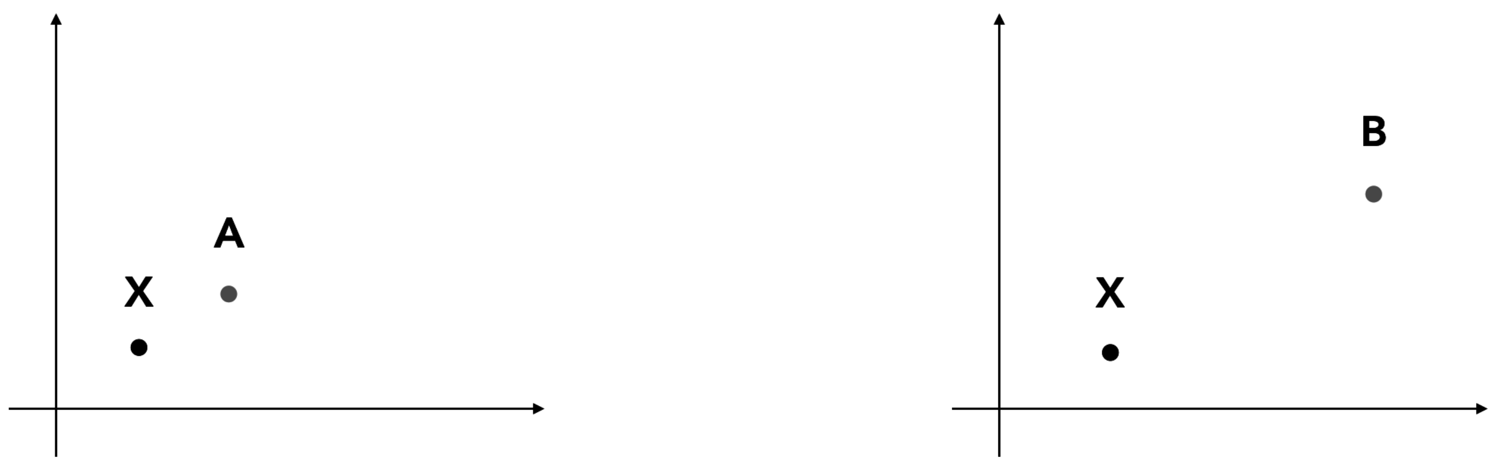
Great Momentum, we have solved 2 / 3 problems. Finally lets have look at the issue of Feature Scaling and then it is only a matter of time before we begin to learn our first machine learning model algorithms. So lets jump in right away !

C) FEATURE SCALING

In textbook definition, Feature Scaling is a process performed in order to normalise the features in a dataset into a finite range. In other words, we perform Feature Scaling in order to bring different sets of ranges of values in a single common range so that our model performs better. Now the question must come, why do we do this ? Well, the real world datasets would contain different ranges from values, for example in a dataset of company's age and average salary, we have different ranges of values (the age ranges in the values of 10s while the salary ranges from values of 1000s and even 10,000s and 100,000s)

Now you may wonder, how is this a problem. Well, let me tell you about some of the Machine learning models. Since using the dataset, each record corresponds to a set of coordinates in an nth dimensional vector space (for example a point in 3D vector space is represented by three coordinates : x, y and z). Now for the sake of visualisation in this book, lets have a look at the below 2 dimensional plots

WHICH OF THE TWO POINTS A AND B
ARE SIMILAR TO X ?



Take a guess to the above answer, you are probably right. Well, I am sure your intuition or lets say, a "gut feeling" may tell you that it is A and you are right! Now that we have found which point is similar, can you guess why ? If you look carefully, you might have notice the fact that the distance between A and X is much much less than that between X and B. I am sure you had used this property to compare the similarities of A and B with respect to X.

Most of the Machine Learning models work just the way you made the assumptions, by comparing distances. There are various types of distance metrics which can be used but we mostly talk about Euclidean Distance

Now the issue is since we have a dataset with large ranges of values as well and since we use the concept of Euclidean Distance, it is highly possible that the model does not give much attention to the feature which has smaller ranges of value (for example Age). This is wrong as we know in most of the cases, Age plays an important role.

Now if we can get the values to be in a similar range, we will definitely have better results as since the values in a common range, the distances between records of a particular feature would not undergo an unfair advantage and true importance would be given to the features that contribute more towards the result

OK, so we have seen that Feature Scaling is important, but you may ask, how do we do it then ? Well below are the most common techniques for Feature Scaling

• NORMALISATION

In normalisation, we try to bring the values in the range of 0 to 1 with the below formula of the converted value based on the previous values

NORMALISATION

$$X' = \frac{X - X_{MIN}}{X_{MAX} - X_{MIN}}$$

It is found better to use Normalisation in processes which involves normalised data. However when you are not sure what to prefer, always choose Standardisation over Normalisation. In other books or codes or research paper, you might not find Normalisation as a text, do not worry as in many cases we also call it as MinMax Scaler as it involves the minimum and maximum values in the records. However do keep in mind that not only Normalisation retains the shape of original distribution, also that Normalisation is sensitive to outliers, which makes sense because an outlier can cause a whole change of values if it acts as the maximum or minimum value which is highly possible as outliers are generally the points away from the group.

Now that we have discussed Normalisation, lets discuss Standardisation which is also known by the name of Standard Scaler

• STANDARDISATION

In standardisation, we change the values such that we have the average around 0 and the standard deviation is around 1. It is useful for optimisation of gradient descent based algorithm (we will discuss Gradient Descent in Chapter NO) such as Regression and Neural Network algorithms. We have the formula of Standardisation as follows

$$X' = \frac{X - \mu}{\sigma}$$

Here **μ** (called "mu", 12th letter of Greek Alphabet) is the Mean (also known as Expectation in the world of maths) and σ (called "sigma", 18th letter of Greek Alphabet) is Standard Deviation (how much values deviate from the mean) Below is the formula of Standard Deviation

$$\sigma = \sqrt{\frac{\Sigma(X - \mu)^2}{N}}$$

We apply feature scaling in two steps, first we find the mean and standard deviation of the Training Set features scale them. This method is called fit_transform() method as we are fitting as well as transforming our values. In the second step, we simply use the values of standard deviation and mean to transform the Test Set features, instead of finding the mean and standard deviation of the Test Set features again. We do this in order to prevent Data Leakage. Do not worry, we are just about to see what do we mean by Training Set, Test Set and Data Leakage. Just remember this, Feature Scaling must be done after splitting our dataset. Also remember, we do not apply on Feature Scaling on dummy variables as these are only representing the categorical values and are already in the ranges of 1s and thus Feature Scaling will do nothing but complicate things.

With that, Congratulations! we have solved all issues we faced while prepping our dataset. Now lets jump in the next section where we see how we can utilise our dataset for Training as well as Testing our model

SPLITTING OUR DATASET

Splitting our Dataset ? Why and How would we do it ? Well, to answer that question, lets play a scenario here. Imagine a scenario where you are a sniper for the army (so cool!) and you have been trained to shoot targets in a fixed set of environment (say jungle). For months and months, you train to shoot targets in that specific environment and are now become an expert in it. Its time for the final test. You are given to shoot targets in jungle. Now obviously you have practiced for months and thus you ace the exam. But now when you go in a war, you see that you have to shoot targets, but instead of jungle, you are in desert. Sure, you know how to shoot but you would not be as precise as in the jungle environment, more specific the jungle where you trained and aced the exam. This is a disaster!

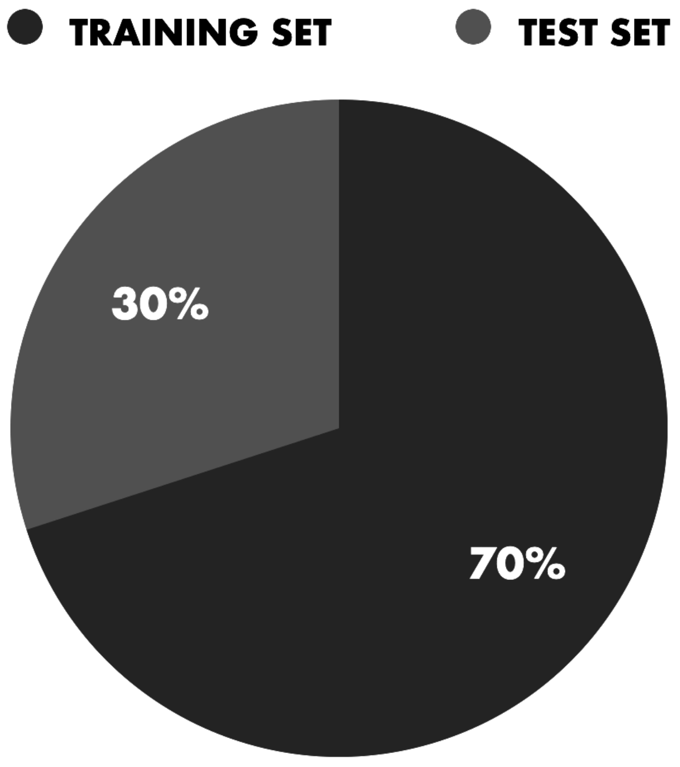
In a more similar way but less cool, if we use 100% of our data to train our model, how do we know it is accurate ? I mean, if we use the same data to test it, surely the model would give an accuracy of 100%!. However is it really true that the model will predict any value 100% ? The answer is obviously not! So what should we do ?

Well the solution is that we split our dataset into two sets. The first dataset will be known as Training set. It contains most of the data and serves its purpose for the model to train on and make the equations and find relations between features and label. The second dataset is called the Test set and it is used for ? Yep ! You guess it! It is used to test our model. Now if you remember in the previous sub heading, we said that we perform Feature Scaling on Training and Testing set separately in order to prevent Data Leakage. What is Data Leakage you may ask ? Well, before we address that, lets address the obvious fact. I am sure it goes without saying that Training and Testing Set must be independent of each other, otherwise we have the whole sniper problem again.

Data Leakage is a phenomenon where data outside the Training set is used for training the model, thus resulting in a violation of Data Independence. It is important to solve the issue of data leakage as if not detected, Data Leakage would cause false results generate, resulting in hiding of flaws in model's performance in the real world. Now that we have seen what Data Leakage is, lets see how is it relevant to Feature Scaling being performed after splitting of data.

Remember the fit_transform method, we find the mean and standard deviation of the values and scale them accordingly, well, lets see what would happen if we perform Feature Scaling before splitting the dataset, the values of Test Dataset would influence the value of Mean and Standard Deviation and since the rest of values are scaled according to Mean and Standard Deviation, we have that the Training Set values being influenced by the Test Set values, thus causing Data Leakage

So that is why we need to split our Dataset, but wait, I never told you how to split it right ? I apologise for that. Well there is no exact percentage in which you must divide the dataset, it totally depends on how much you want your model to be exposed to varieties of dataset. If you are unsure, go for the common split : 70% of Data to Training Set and 30% of Data to Test Set. The general idea is to keep Training Data more than Test Data.



Before we look at the libraries we use for Machine Learning, we need to cover the statistics we often use in the Machine Learning world for evaluation of models, basically checking how good is the model and where does the flaws occur

STATISTICS FOR JUDGING AND COMPARING MODELS

As we all know by now, Machine Learning models work on mathematics, and statistics is one of the most crucial part of maths in the world of Machine Learning. Statistics is a branch of mathematics which helps us to analyse our data. Statistics often includes pattern finding, data collection and most importantly data visualisation. A proper visualisation of data helps in understanding relationships between labels and features better as well some potential patterns before we feed it to the model to play with.

First lets look at the terms True and False along with their integration with the terms Positive and Negative. We call a judgement Positive when the prediction predicts a positive match for that category. Similarly, a judgement is Negative if the prediction predicts a negative match for the same category. For example, if a model works on predicting if an animal in the image belongs to the cat species or a dog species on the basis of parameters like eyes, fur, ear shape and so on. Lets say the model predicts that the image belonging to the species of Cat, therefore for the categories of Cat, we have a Positive Judgement and for the categories of Dog, we have a negative judgement.

Now that we have seen what we mean by positive and negative judgements, lets see what do we mean by True Positives and True Negatives. True Positives is a situation where the model predicts a Positive Judgement for a category and in reality the label belongs to the positive judgement. For example, if an image from Test Data belongs to the category of Dogs and the trained model predicts it to be a Dog, we have a situation of True Positives. Similarly True Negatives is a situation where the model predicts a Negative judgement and in reality the judgement is really Negative. For example, the model predicts the image not of Dog category and in reality, the image is not of the Dog Category

So, True Positives and True Negatives seems like a simple concept, but do we have something like False Positive and False Negatives, and the answer is yes, and many people often get confused what are False Positives and False Negatives. Fret not, dear reader, for I have found a simple way to remember them, but before that lets check out False Positives and False Negatives.

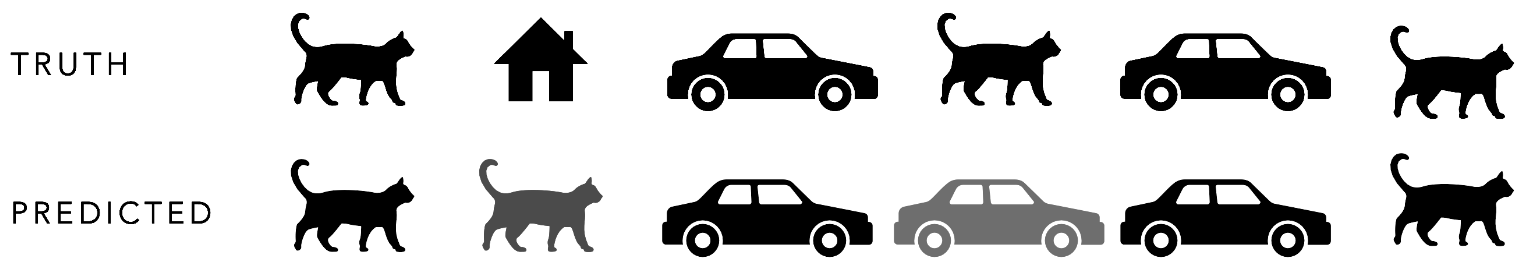
False Positives occur in a situation where the model predicts a Positive Judgement for a category but in reality the image does not belong to that particular category. For example, the model predicts that the image belongs to the category of Dogs but in reality the image belonged to the category of Cats. False Positives are also known as **Type - I error.** Similarly we have False Negatives, where the model predicts the Negative Judgement for a category but in reality the image belongs to the same category. For example, the model predicts that the image is not of a Dog but in reality it was the image of Dog. False Negatives are also known as **Type - II Error** and generally are more dangerous for our model than Type - I error. The best way to understand is, lets say we arrested a suspect for a particular crime and lets say we fed the data into a lie detector. According to Type I error, we would think that the person is lying while in reality he was telling the truth, this would cause some disturbances sure but eventually the right decision would be found and it would be well at end. However, according to Type II error, we would have the person as innocent but in reality he was the criminal, so we would set him free and who knows, by the time we find the mistake, he would already have done something even more terrible.

Now to represent True Positive, True Negative, False Positive, False Negatives, we use a table like structure we often call Confusion Matrix. It is nothing complex really, as the name suggest it is a matrix where for Rows we take the True and False of Predicted values and the Columns are for True and False of Actual Values. The diagonal of Confusion Matrix represents the True Positives and True Negatives while the Non Diagonal Values represent False Positives and False Negatives

Now in case it is not evident by the terms used here, remember that Confusion Matrix is only for Classification Problems as classification problems have unique discrete categories. Another advantage of Confusion Matrix is that we can easily use it to measure other measurements such as Accuracy, Recall, F1 – Score (discussed later in the same heading)

		ACTUAL VALUES	
		POSITIVE	NEGATIVE
PREDICTED VALUES	POSITIVE	TRUE POSITIVE	FALSE POSITIVE
	NEGATIVE	FALSE NEGATIVE	TRUE NEGATIVE

A point to remember, this is the general notion on how you can create your confusion matrix, in some cases or books you might notice that the values may be same but the pattern might be different, do not worry about that, it is the same thing just represented in a different way. The above figure shows the Confusion Matrix, in still cannot understand what are these True and False terms, do not worry, this matrix and its terms are confusing at first but eventually you get it or you look at the below figure and hopefully it will clear all confusions. In the below scenario, the values are in terms of prediction and truth whether the object is a cat or not



TRUE POSITIVE :	**2**	**FALSE POSITIVE :**	**1**
TRUE NEGATIVE :	**2**	**FALSE NEGATIVE :**	**1**

Now that we have understood what we mean by Confusion Matrix, lets look at how to use the Confusion Matrix to get scoring values like Accuracy, Precision, Recall and so on…

A) CONFUSION MATRIX

As the name suggest we have a matrix well, in common terminology, it is your everyday table with rows and columns where we usually we have the True Values as the Columns and Predicted Values on Rows, but as mentioned before, it is the general way to represent the matrix, you can also have Actual Values on Rows and Predicted Values on Columns provided the values of Accuracy, Precision, Recall etc are correct in both cases.

A note to remember is that Confusion Matrix is made only for Classification Type Problems, we will see Classification Problems in Chapter No. Now lets have a detailed look at one of the most common measured metric : Accuracy

B) ACCURACY

Accuracy in Machine Learning is same as its usage in real life scenarios. Basically, out of maximum points, how many points are correct. For example in an examination of 150 marks, scoring 75 marks will result in an accuracy of (75 / 150) x 100 = 50%. Now that we already have been introduced to terminologies such as True and False Positives and Negatives, lets have a look at the formula of accuracy

$$\text{ACCURACY}$$

$$\frac{\text{TRUE POSITIVES} + \text{TRUE NEGATIVES}}{\text{TOTAL PREDICTIONS}}$$

In the Cat Identification example, discussed earlier, we have True Positives as 2 and True Negatives as 2 for a prediction of 6 items, thus in that case

$$\frac{2 + 2}{6} = \frac{4}{6} = 66.67\%$$

Now you may have noticed, in Confusion Matrix Section, I introduced Accuracy as one of the "most common" measured metric. Now you might be thinking, why not Accuracy is the "only" metric to evaluate how good is our model ? Well, that is because we have a major loophole in the accuracy system.

Lets consider the below situation where we have built a Gun Detection Model in order to improve security in an area. We trained a model and have the below confusion matrix (Note : The structure is different than the standard form, however the results are same)

		PREDICTED VALUES	
		GUN	NOT GUN
ACTUAL VALUES	GUN	100	30
	NOT GUN	60	10

Now, according to the formula of accuracy, **Accuracy** = (100 + 10) / (100 + 10 + 30 + 60) which corresponds to 110 / 200 which in percentage is **55%**. Keep this in mind. Now, lets say your best friend comes to you and says, "Why not classify every image as a gun ? That way we will always detect a present gun and if the object is not a gun, we will let the person go“. Obviously, this idea is not useful in the real world as it would result in unnecessary confusion and arguments but hey, since your best friend suggested a new approach, it is your duty to check this approach as well. Now, since the model always predicts the items as a gun, that means there are going to be 0 predictions for 'Not Gun', and their predictions would be in fact added into 'Gun' Prediction, thus creating the following confusion matrix

		PREDICTED VALUES	
		GUN	NOT GUN
ACTUAL VALUES	GUN	130	0
	NOT GUN	70	0

Now, in this case, **Accuracy** = (130 + 0) / (130 + 0 + 0 + 70) which is 130 / 200, which eventually results in Accuracy as **65%.** Notice anything ? If not, no worries, have a look below. This phenomena is known as **Accuracy Paradox** in the Machine Learning World

ACCURACY FLAW

ACCURACY WITH MODEL		ACCURACY WITHOUT MODEL
55%	**V / S**	**65%**

NOT USING THE MODEL GIVES US A BETTER ACCURACY, HOWEVER THE LOGIC OF THE BETTER SCENARIO IS NOT PRACTICAL AS MENTIONED BEFORE

Now, you may ask, what are some other metrics which we can use ? Well lets have a look at other common metrics which are often discussed

C) PRECISION

Precision in its common usage may sound similar to Accuracy, however we have a different formula with Precision. Precision is basically the ratio of True Positive to Total Positive Outcomes made. I know, it might be slightly difficult to remember in comparison accuracy, but do not worry, let me share a common trick to remember it : Precision start with the letter 'P', the 15th member of the English Alphabet. Now, the word Positive also starts with the letter 'P'. I know, its a conspiracy created by elite people who control the world (just kidding), but seriously, P = Precision = Positive. The below image is perhaps more presentable in terms of remembering and understanding the formula

PRECISION

$$\frac{\text{TRUE POSITIVES}}{\text{TRUE POSITIVES + FALSE POSITIVES}}$$

In our Gun Detector Example (the situation where we are using the model), Precision comes out as 100 / (100 + 60) which basically tells us that Precision is 0.625, in other words, when the model predicts a gun, the probability that it is correct is 62.5%. Precision is often accompanied with another metric, in other words, Precision has a best chum, which we call Recall. We use a balance of Precision and Recall as another way to evaluate the quality of our model in the most common metric, F1 - Score. Well, lets have a look at Recall

D) RECALL

Recall, as mentioned before, is usually heard along with Precision, but in its self, Recall is the ratio of True Positive with summation of True Positive and False Negative. Wow! Quite a mouthful isn't it. Okay, so we know what it is, how are we going to remember it ? Guess what ? I gotcha back! Before I share a trick to remember the formulas of Precision and Recall lemme just present you the formula of Recall in a more presentable manner

RECALL

$$\frac{\text{TRUE POSITIVES}}{\text{TRUE POSITIVES + FALSE NEGATIVES}}$$

In our Gun Detector Example, Recall is 100 / (100 + 30) which comes out to Recall as 0.769. Now the trick to remember the formula of Precision and Recall as promised :

- Precision :
    - Start with P : Related to Ratios of Positives
    - Concerned with **Positive Prediction** Values
- Recall :
    - Concerned with **Positive Actual** Values (Remember False Negatives are values which are ***actually true*** but are predicted as false)

E) F1 – SCORE

Unfortunately, F1 Score has nothing to do with the awesome motorsport but it is equally interesting. Remember I mentioned, we use a **'balance'** of Precision and Recall to evaluate quality of Machine Learning Model. Well, F1 Score is one such balance and frankly F1 Score is the easiest formula to remember after accuracy, Why ? Well, that's because you can say the formula in a single simple sentence. Ready ? Here it goes :

F1 – SCORE IS THE HARMONIC MEAN OF PRECISION AND RECALL

If you are not familiar with the term 'Harmonic Mean', no worries, you can simply remember the below formula or if you want to go the extra mile, Harmonic Mean is like the Arithmetic Mean but instead of finding the mean of a and b as (a + b) / 2, we find the mean of 1/a and 1/b, thus if c is the Harmonic Mean of a and b, then equation is 1/c = 1/2(1/a + 1/b). Anyways, here is the formula

F1 – SCORE

$$\frac{2 \times \text{RECALL} \times \text{PRECISION}}{\text{RECALL} + \text{PRECISION}}$$

F1 Score is one of the widely used metric to evaluate the model with the simple concept of Better F1 Score corresponds to Better Quality of Model. There are many many more metrics for evaluation of models, but these metrics are the most common terminologies used.

Now that we are more familiar with popular metrics used for evaluation of our models, lets have a look at some of the common libraries that we use in Machine Learning. Please note that I will be discussing the most commonly used libraries specific for Machine Learning, but if you encounter other libraries and are more comfortable using them, please feel free to use them, maybe some of them work better than others

COMMON LIBRARIES OF MACHINE LEARNING

If you are unfamiliar with the concept of Libraries, you can imagine them as their real world application. Libraries are places where we find many books right ? Or in other words, Libraries is a collection of books. Now libraries can vary in sizes, from a small collection of 10 books to large libraries which contain subsections like Fiction, Medicine and so on… Similarly, in programming, Libraries are collection of different pieces of codes (also sometimes referred as modules). Lets have a look at the most commonly used libraries in Machine Learning. (Note : It is **important** you have a **good grasp of basics of Python** before you use these code snippets)

A) PANDAS

Pandas is a Python Library which is commonly used in performing Data Preprocessing such as OneHotEncoding, Handling Missing Values and so on… Pandas is also used to import our data present in various formats, such as .CSV, .XLSX and others (if you are unfamiliar with these extensions please refer to subsection : 'Datasets : How Data is Stored ?' In Chapter 2). Let's have a look at the below code pieces for various situations that we would typically use for Machine Learning (Note : These are **NOT** sorted in any form of importance)

- IMPORTING PANDAS IN PYTHON

```
import pandas as pd
```

- READING CSV FILES IN PANDAS

```
dataframe_name = pd.read_csv("Enter Name")
```

- DIVIDING BETWEEN FEATURES AND LABEL (ASSUMING LAST COLUMN AS LABEL WHILE OTHERS AS FEATURES)

```
X = dataframe_name.iloc[:, :-1]
y = dataframe_name.iloc[:, -1]
```

In Pandas, the **iloc** method is used to select section of data frame by providing the set of indices range of the form [Indices of Rows , Indices of Columns]. Since we are selecting all records, therefore for Rows, we will put the range as : (Remember, ‘ : ’ indicates selection of all indices, in this case, for rows), while for Columns since we are selecting all columns other than the last one for Features, we will put the range as ‘ :-1 ‘ (Remember, Python is 0 - indexed language and permits usage of negative indices). Similarly, we will select all rows and the last column.

NOTE

There are other ways to extract Features and Labels. For example you can select the Features by providing the Column Names instead of Range or you can make a copy of data frame and drop the irrelevant columns for both cases. However, it is recommended with usage of indices as it makes your code more robust to other situations as well as in cases where you share the code with your friends or colleagues, which is always appreciated

- DROPPING COLUMNS

```
new_df = old_df.drop(“Column Name”, axis=“columns”)
```

- CONVERTING TO CSV

```
new_df = dataframe_name.to_csv(“CSV Name.csv”)
```

NOTE

It is suggested that we mention the extension of **.csv** as well, this is because when the file is exported, it will be stored as example.csv which makes it easier for us to later to identify the file format of the file, which is helpful in practical world

- GENERATING DUMMY VARIABLES FOR ONE HOT ENCODING

```
pd.get_dummies(“DataFrame Name”)
```

- LIST OF COLUMNS PRESENT (RETURNS COLUMN NAMES)

```
dataframe_name.columns
```

- STATISTICS OF DATA FRAME

```
dataframe_name.describe()
```

- FINDING COUNT OF EMPTY VALUES IN EACH COLUMN

```
dataframe_name.isna().sum()
```

- REPLACING EMPTY RECORDS

```
df[“Column Name”].fillna(“New Value”,inplace=True)
```

NOTE

Here **inplace** is an optional parameter, which is used to replace the old data frame with the newly updated data frame, if that is, this parameter is set to True data type. However, if you set it to False, then you would need to store the result in a new variable, so it is suggested to set inplace as True, in order to make your code cleaner and more manageable

- FINDING COUNT OF EMPTY VALUES IN EACH COLUMN

```
dataframe_name.isna().sum()
```

- LOOKING AT TOP N RESULTS (BY DEFAULT N = 5)

```
dataframe_name.head(n)
```

- LOOKING AT BOTTOM N RESULTS (BY DEFAULT N = 5)

```
dataframe_name.tail(n)
```

NOTE

We use the **head** and **tail** method to look at the top n and bottom n records respectively along with their respective values in records. We generally use the head function more than the tail function, and you fill find that every good code contains either of these. These are usually executed to get an idea of what are the features and labels on the basis of their values as well as the datatypes of each column, which is particularly useful to check if there is any requirement of Data Preprocessing such as OneHotEncoding or Feature Scaling

- DATATYPE PRESENT IN DATASET

```
dataframe_name.dtypes
```

- GROUPING OF RECORDS

```
dataframe_name.groupby([“Column Name”])
```

Pandas also has many more function such as **date_time()** which are used in cases where we want the involvement of date, time or both, for example, in TimeSeries Analysis (we predict sales in next time duration on the basis of previous sales). However, I have not discussed them as those codes are situation specific while I am sharing the most common pieces of codes.

Now that we have looked at Pandas, lets have a look at another libraries which are often used with Pandas. This library is known as Numpy

B) NUMPY

Numpy is used as a replacement of List for construction and manipulation of arrays and matrices. Now I know you might be wondering, If we already have Lists, what is the need of Numpy Array ? Well, that's a great question. Lets have a look at the answer before we look at the codes

• NUMPY ARRAYS ARE FASTER

Numpy arrays are faster due to their inner kernel built in C and thus binds C with much slower Python. Due to this, Numpy takes less execution time in comparison to heavier Python Lists. To see this in practice, I would recommend you to run the below piece of code for both list and numpy array of a large size (say 1 million items)

```
import time

start = time.time()

/* Piece of Code */

end = time.time()

print(end - start)
```

• NUMPY ARRAYS CONSUME LESS SPACE

Numpy Arrays consume less memory space in comparison to Python Lists, which is always a good thing in a code and thus eventually results in faster runtime due to faster load time and execution time (lower the memory space, faster it is to load the numpy arrays in memory). To compare, you can execute the below piece of code. Note that to compare these two, ensure that you have entered the same number for size

- ### TO CHECK SIZE FOR NUMPY ARRAY

```
import numpy as np
a = np.arange(“Enter the Size”)
print(a.size)
```

- ### TO CHECK SIZE FOR PYTHON LIST

```
import sys
b = []
for i in range(“Enter the Size”):
    b.append(i)
print(sys.getsizeof(b))
```

NOTE

We can also convert Pandas DataFrame to Numpy Arrays, this is often practiced in many codes in order to increase efficiency and speed. To do that, you would need to add the extension **.values** to the end of the code, for example in the below situation :

```
X = dataframe_name.iloc[:, :-1].values
```

Below are the codes of some of the most common applications of Numpy. Please note that similar to Pandas, these are not sorted in any order of importance

- **IMPORTING NUMPY IN PYTHON**

```
import numpy as np
```

- **CONVERTING PYTHON LIST TO NUMPY ARRAY**

```
np.array([...])
```

- **CREATING AN ARRAY OF SIZE N**

```
np.arange(N)
```

- **CHECKING SHAPE OF ARRAY (RETURNS NO OF ITEMS / DIMENSION)**

```
numpy_array.shape
```

- **ACCESSING ELEMENT OF ITH ROW AND JTH COLUMN**

```
numpy_array[Row No][Col No]
```

- **SORTING AN ARRAY**

```
np.sort(array_name)
```

- **NUMBER OF DIMENSIONS**

```
numpy_array.ndim
```

- **CONVERTING DATA TYPE OF ITEMS IN ARRAY**

```
numpy_array.astype("DataType")
```

- **RESIZING THE ARRAY**

```
np.resize(numpy_array, (new shape))
```

Many times we may need to resize the array into a different shape. For example, np.arange(8) would return the array [0, 1, 2, … 7] and the shape would be (8,). Now on reshaping it to (2, 4) would result in the array [[0, 1, 2, 3] , [4, 5, 6, 7]]. As you can see, we now have a matrix of 2 Rows and 4 Columns. It is important that the n, number of items in new shape is same as that of the original array, otherwise, it will take the first n items out of the original array

- **CHECKING DATATYPE OF ARRAY**

```
numpy_array.dtype
```

NOTE

In case we have multiple data types in a numpy array, the **dtype** function returns datatype of the array as <U*int* where U denotes Unicode (in situations of String) and int is usually the length of longest string. In situations where we do not have String, for example, bool + int, bool + float, and others, the dtype returns the higher member of typecasting where the order of type casting is : Float > Int > Bool as we can convert lower members to upper.

- CREATING AN ARRAY OF 0s OF SHAPE (N1, N2, . . .)

```
np.zeros(N1, N2, ...)
```

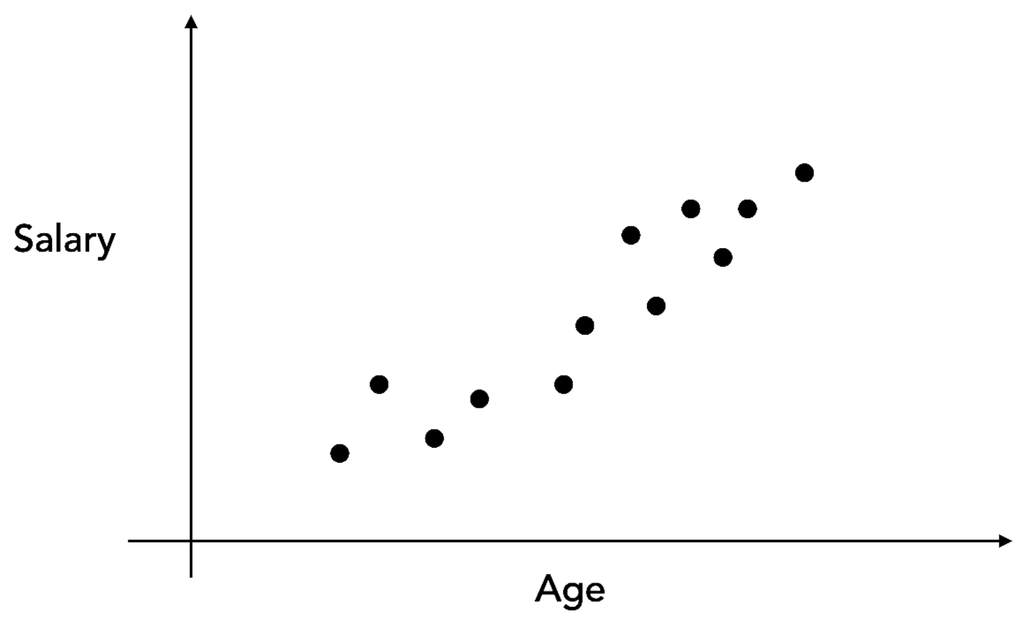
- CREATING AN ARRAY OF 1s OF SHAPE (N1, N2, . . .)

```
np.ones(N1, N2, ...)
```

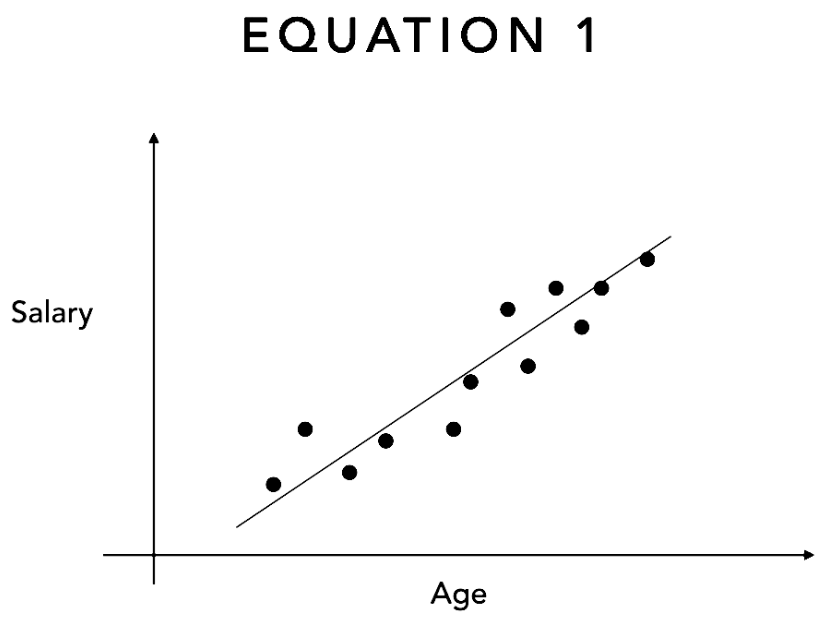
In various Machine Learning related applications such as Computer Vision, NLP and others, we often need to create an array of zeros or ones of size N. For example, np.zeros(5) and np.ones((2,2)) will result in [0, 0, 0, 0, 0] and [[1, 1] , [1, 1]] respectively. By default the datatypes of the outputs is Float but if you want them in different data type, then you would also need to include the parameter **dtype** as dtype = "Data Type".

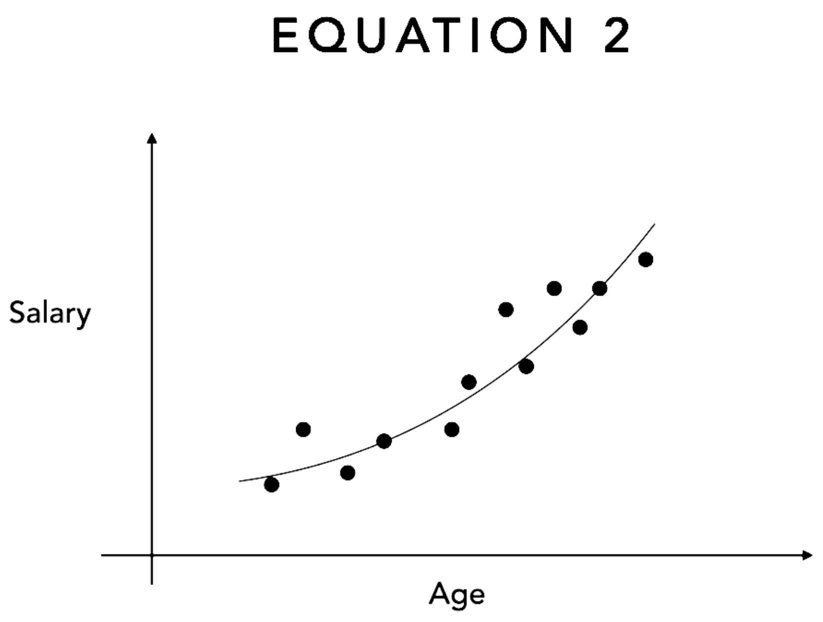
C) MATPLOTLIB

Matplotlib is used for data visualisations, specifically, we use the pyplot module to plot our data on graphs. Now you might ask, why do we do it ? Well, with plotting, we can understand the data better and provide a small intuition of which model might be better. For example consider the below situation where we have a small dataset of age of employees and their corresponding salaries



Now the above pattern is something like a curve. Now you lets say we have two equations and below are their corresponding graphs when plotted with the dataset. Which one do you think, captures the pattern better ?





Although, it may not be visible that much, but a gut feeling must tell you that the curved equation might be slightly better than the linear equation in terms of capturing the pattern, and you are right! Now I am sure you are wondering, how does plotting our dataset help us in this case, a perfectly logical question. Lets see, in the example above, I gave you two options, that is two models to choose from. However, as we will see in the next chapter, we have many many more options of models available. Now to find the best model, you would have to try all models and select the best model (highest accuracy or F1 Score). Moreover, you would also need to find the best possible combination of parameters. This can take hours, even days to find the best model. However, if you plot your data, you can get a small idea to eliminate some of the models. This definitely helps our case as it saves us time and resources as well.

We can plot different types of charts for the same data, depending on the situation and expected output, you would have to choose the best chart type which satisfies the problem / task's requirements. However, I will be only discussing some of the common plots we use and their codes

i. GENERAL CASES

These pieces of codes are used in almost every case, so it is absolutely fitting, that we start with them. These also includes some of the codes to make changes in how the plot is looking.

- **IMPORTING PYPLOT FROM MATPLOTLIB**

```
import matplotlib.pyplot as plt
```

- **NAMING X AND Y AXES**

```
plt.xlabel(‘X - Axis’)
plt.ylabel(‘Y - Axis’)
```

- **GIVING A TITLE TO CHART**

```
plt.title(‘Title of Chart’)
```

- **USING A LEGEND (USEFUL FOR DIFFERENT COLOURED PLOTS)**

```
plt.legend()
```

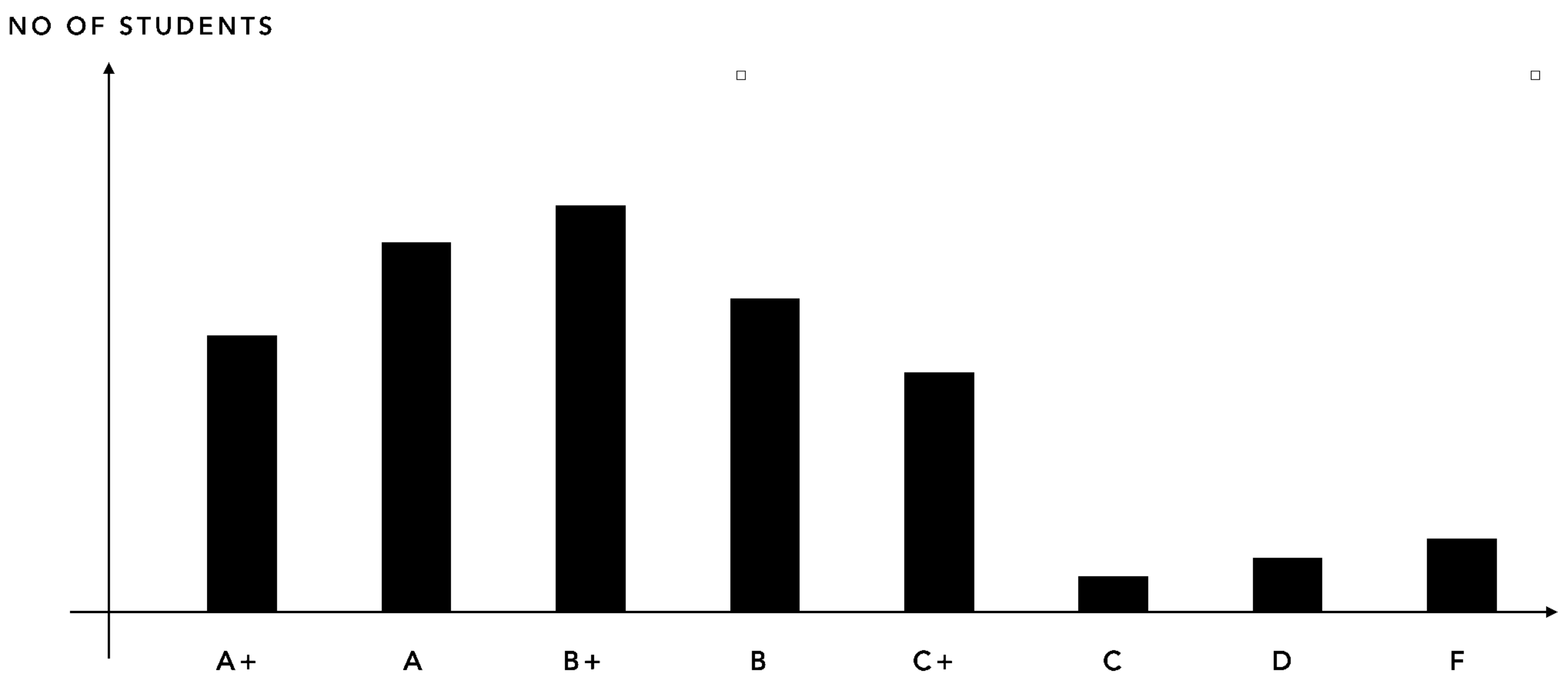
- **SETTING CUSTOM X AXIS TICKS**

```
plt.xticks(label_array, pos_array)
```

Label Array contains the custom ticks in sequential order and Position Array contains the position in which the corresponding tick must be placed. You would better understand the usage and corresponding output of this function in the example cases. Similarly to xticks we can also set the yticks

ii. BAR CHART

Bar Charts, as the name suggests, is a chart / graph which contains bars. These are often used in situations where we want to compare our data on the basis of different categories. For example, to analyse how many students got which grade in their final exam can be represented by bar chart like below



Below are the codes for BarCharts :

- **IMPORTING PYPLOT FROM MATPLOTLIB**

```
import matplotlib.pyplot as plt
```

- **IMPORTING PYPLOT FROM MATPLOTLIB**

```
import matplotlib.pyplot as plt
```

iii. HISTOGRAM

iv. PIE CHART

v. SCATTER PLOT

Complete

CH 3 - REGRESSION

TYPES OF MACHINE LEARNING

Congratulations! Now that we have done Data Preprocessing, we will now look at the different types of Machine Learning algorithms. Note that we also have different fields included with Machine Learning such as NLP or Computer Vision, but in this book, I am only talking about the three main types and algorithms of Machine Learning

A) SUPERVISED LEARNING

As the name suggests, these types of algorithms requires external supervision. By External Supervision, we mean that, we need to provide both the features and their corresponding Labels, or in other words, in these type of algorithms, the output is known and the model creates equations to map the features to labels. For example, if I want to create a model that predicts an image to be that of a cat or a dog. I would need to provide the model with the image and declare that, "Hey! The image I just gave you, its a cat! Now compare it with this new image and tell me does the new image looks like a cat, based on the image I gave you earlier and when you tell me what you think, I will tell you if you are correct or not".

Now that you know what do we mean by supervised learning, I would like you to guess some of the applications of Supervised Learning, in daily life. Here are some of the applications of Supervised Learning :

- Weather Forecasting
- Stock Price Analysis
- Email Spam Detection
- Facial Recognition
- Biometrics Security (Speech, FingerPrints and so on…)

See! From simple Spam Detection of your Gmails, to Multimillion dollar advanced security system. Anything can be achieved with Supervised learning, provided your application covers the basic requirement of Features and the corresponding Label.

There are two types of algorithms in Supervised Machine Learning :

- **REGRESSION :** Regression algorithms are used for Continuous Predictions. Continuous Data is data within a particular range. For example the number of real numbers between 0 and 1 is a set of continuous data. Some real world problems that include regression solutions is Prediction of Housing Prices, Stock Price Prediction (also known as Time Series Analysis)

- **CLASSIFICATION :** Classification Algorithms are used for Discrete Predictions. Discrete Data, unlike continuous data can only take limited values. For examples, a person can only have one type of eye colour (either blue, green, brown and so on…, we never have situation where the eye colour is say, 30% blue, 20% brown and 50% black). Some of the the real world problems that include classification solutions is classification of images into different bundles on the basis of the person being a male or a female.

SUPERVISED LEARNING

PROVIDE FEATURES AND LABELS

REGRESSION

- CONTINUOUS LABEL
- EX : STOCK PREDICTION

CLASSIFICATION

- DISCRETE LABEL
- EX : COLOR CLASSIFIER

B) UNSUPERVISED LEARNING

Contrary to Supervised Learning, Models based on unsupervised learning algorithms do not need labels and are in fact, learning on their own. The model learns by the usage of unlabelled data and thus does not have a fixed output to compare to. Now you might be wondering, when do we use unsupervised learning algorithms ? A good question indeed! Unsupervised Learning, is used to find patterns and other relationships to be exploit later. For example, lets say you work for a fashion company with hundreds and thousands of clients with hundreds of parameters such as earning, age, ethnicity and so on… Now imagine that your company has a special sale for the customers that spend a lot. Perfect! But there is one issue, we do not know in how many categories we can divide our customers into. Thanks to Unsupervised Learning, we can divide our customers into say 5 categories and then we can later provide them some meaning. Or lets take an even more fun example, imagine that you are a supermarket owner and you want to find which item is sold more with your bread, whether its peanut butter, jelly, butter and so on…How will you do it? Well, here is where Unsupervised Learning plays an important role. With help of Unsupervised Learning, you can find the most paired item with bread.

Some of the other applications of Unsupervised Learning include :

• Recommender Systems (for Movies)

• Similarity Detection

• Customer Segmentation

and many many more…. One of the most common usage of Unsupervised Learning is Dimensionality Reduction. Remember, from Chapter 2, every new feature leads to a new dimension. Sometimes when you have a lot of features, it increases the computational strain on the model, which result in a sharp decrease in accuracy. To solve this, we use **Dimensionality Reduction** techniques which reduces the number of columns, making the dataset more controllable while ensuring that the general idea is same but we will look more into that algorithms in **Chapter No**

From above, we can say that we have three types of unsupervised learning algorithms:

- **CLUSTERING :** As the name suggests, Clustering Algorithm forms clusters (collection of objects on the basis of some criterions such as similarities). Basically, with clustering algorithms, the model groups data on the mentioned constraints. For example, we can form clusters of different types of fruits in terms of various factors such as colour, taste, region and so on…

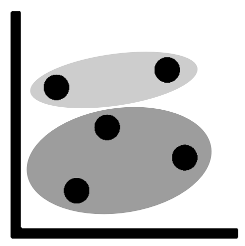
- **ASSOCIATION RULES :** Association Rules algorithms are used for finding relationships between variables on the basis of some defined rules. For example, a supermarket can use association rules to find which pair of items are commonly bought together and thus depending on strategy, these can be put together like bread and milk, peanut butter and jelly and so on…

- **DIMENSIONALITY REDUCTION :** Dimensionality Reduction, as mentioned before is used in situations where there are too many features in a given dataset. For example, for a dataset about cheese, there can be 10+ features such as milk type, region area and so on.. with help of Dimensionality Reduction, we can reduce these into say 3 or 4 features, thus reducing computational complexity as well as perhaps, increase the accuracy

UNSUPERVISED LEARNING

PROVIDE ONLY FEATURES

CLUSTERING	ASSOCIATION RULES	DIMENSIONALITY REDUCTION
• GROUPS DATA INTO CLUSTERS	• RELATIONSHIPS UNDER RULES	• RELATIONSHIPS UNDER RULES
• EX : CUSTOMER SEGMENTATION	• EX : RECOMMENDER SYSTEM	• EX : COUNTRIES TO REGIONS



C) REINFORCEMENT LEARNING

Reinforcement Learning is something different than supervised and unsupervised algorithms. In this situation we do not provide the model with data. Wait what? I know, its not a typo. In Reinforcement Learning, we provide the model, or in this case, we call it an agent, with an environment and allow it to explore the environment on its own. Now since we are not feeding any data, there is no concept of input and output. Instead, we have two states : Starting State and Ending State and what the model does is tries all approached from Starting State to the Ending State and work on the constraints of Rewards and Punishment

For example, in your school, if you scored a good grade in examinations, your parents might take you to dinner to your favourite restaurant or give you money as a reward. This provides as a motivation for you to perform good in other examinations as well and thus in this case, we call it a Positive Reinforcement. However, lets say you scored a terrible grade, your parents might lecture you and criticise you for your poor performance. This provides as a lesson of what not to do next time in order to avoid this scenario and thus we have a situation of Negative Reinforcement.

Some of the applications of Reinforcement Learning are :

- Robotics
- Healthcare
- Manufacturing

and so on… The concept of Reinforcement Learning has been present for a while, but its computational applications have begun recently, proving it to be a new but life changing field

Now that we have seen the types of learnings, lets jump into the first type of learning in supervised category, that is Regression, which literally requires nothing but simple mathematics to create the most basic model

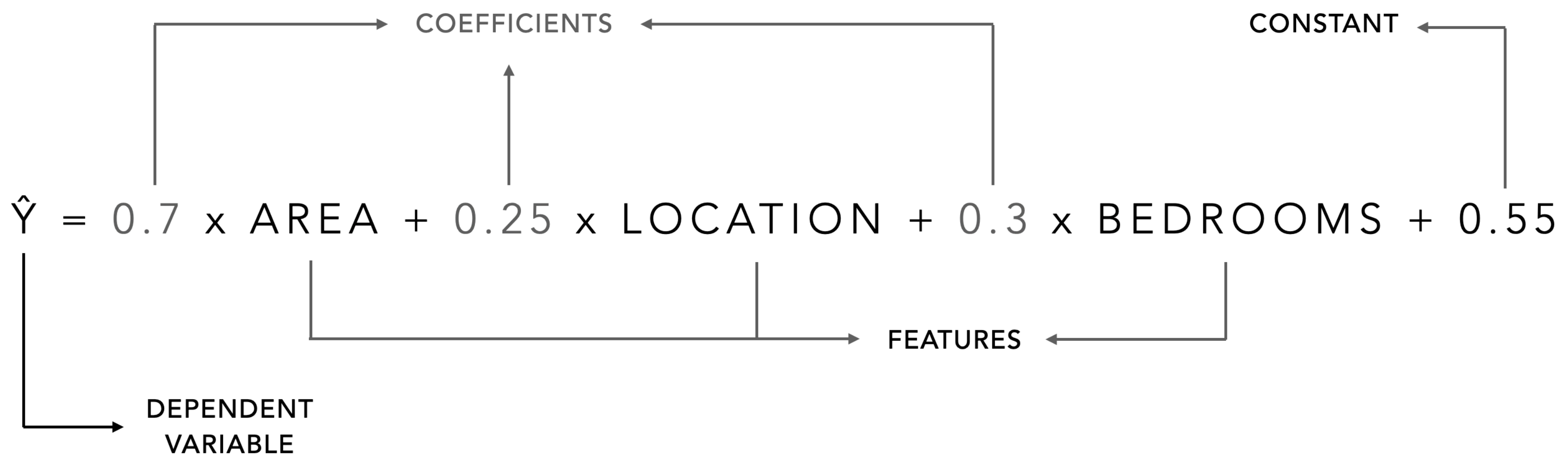
LINEAR REGRESSION

Simple Linear Regression or commonly called Linear Regression is the most basic Regression Model that can be made as the equations behind the Linear Regression is simply a linear equation as below

$$Y = M_1X_1 + M_2X_2 + M_3X_3 + \ . \ . \ . \ C$$

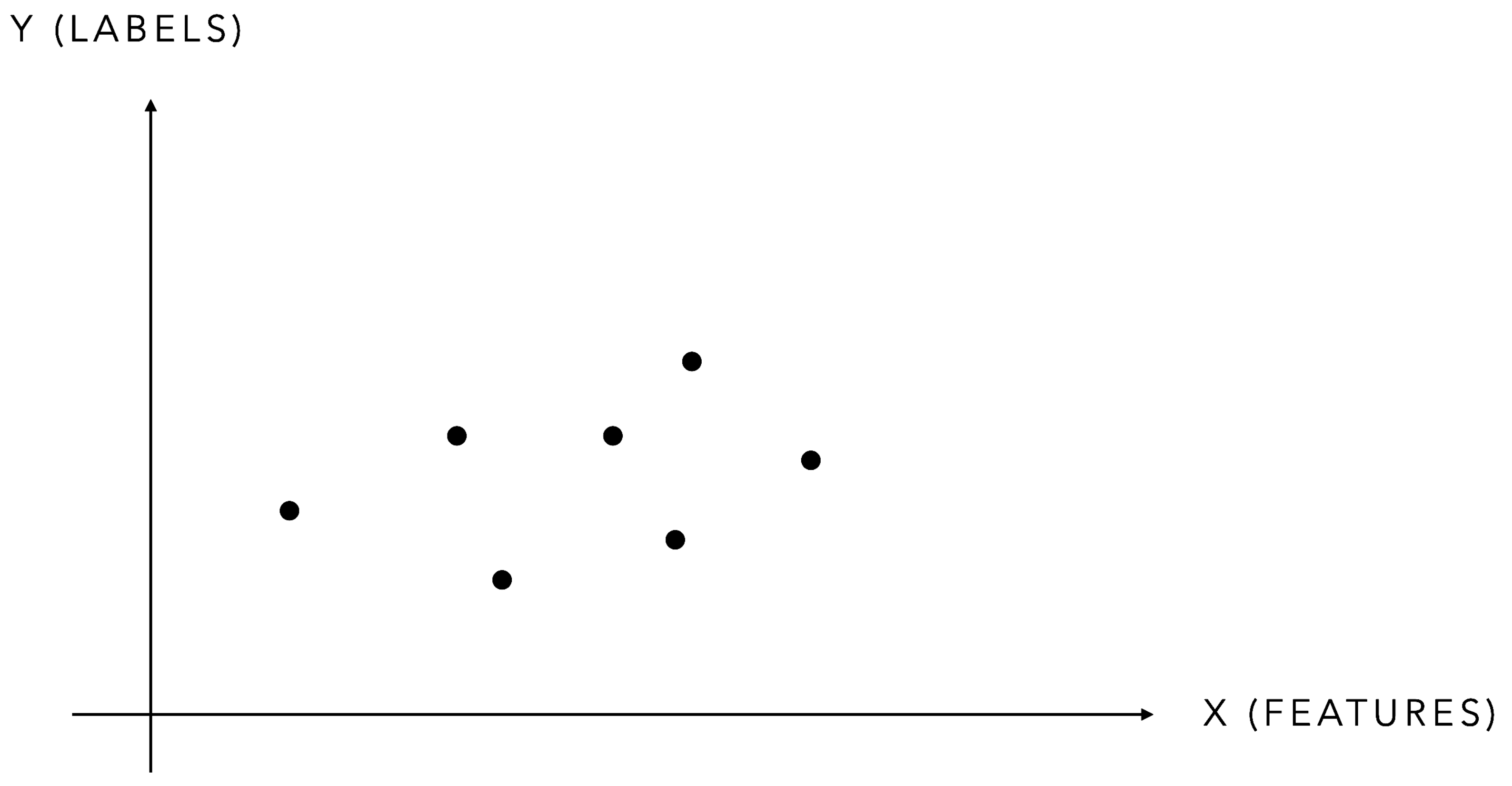
- Y = OUTPUT
- X_i = I
- M_i = I^{TH} COEFFICIENT
- C : INTERCEPT / CONSTANT

For example, lets say in a situation of predicting prices of houses, we only consider 3 parameters, i.e. Size of House, Number of Bedrooms and Location. Lets say after we feed this into the model, we get the below equation (we will look at how we will get these coefficients later)

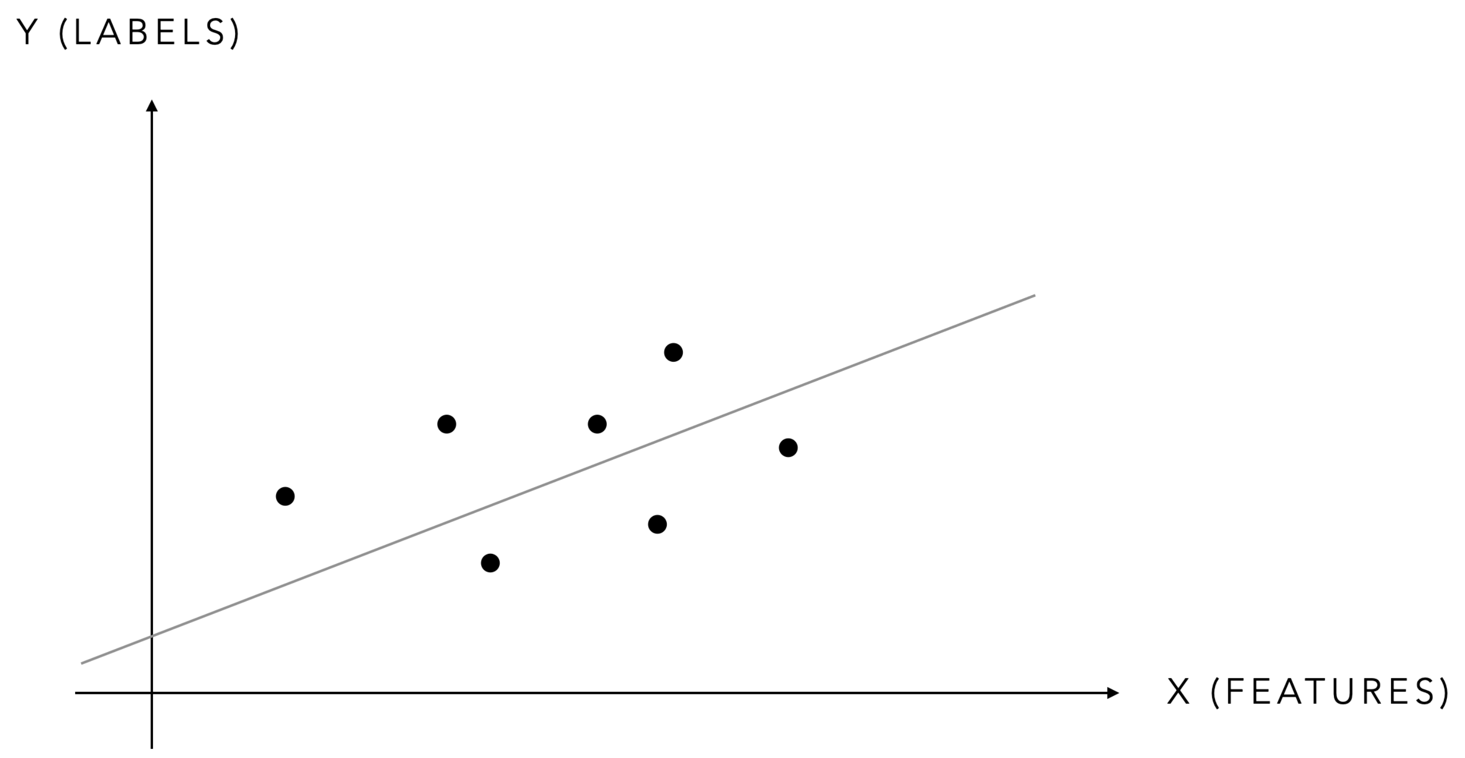


Now say we have a new house built in New York of a size of 10000 sq. ft with 3 bedrooms, on putting these values inside the above equations we will get a value, for example, USD 50000 (To any New Yorkers, first of all, Hello :), secondly, I have absolutely no clue about the pricing in New York, so do excuse me if the amount I mention, well, looks ridiculous), but you get the idea.

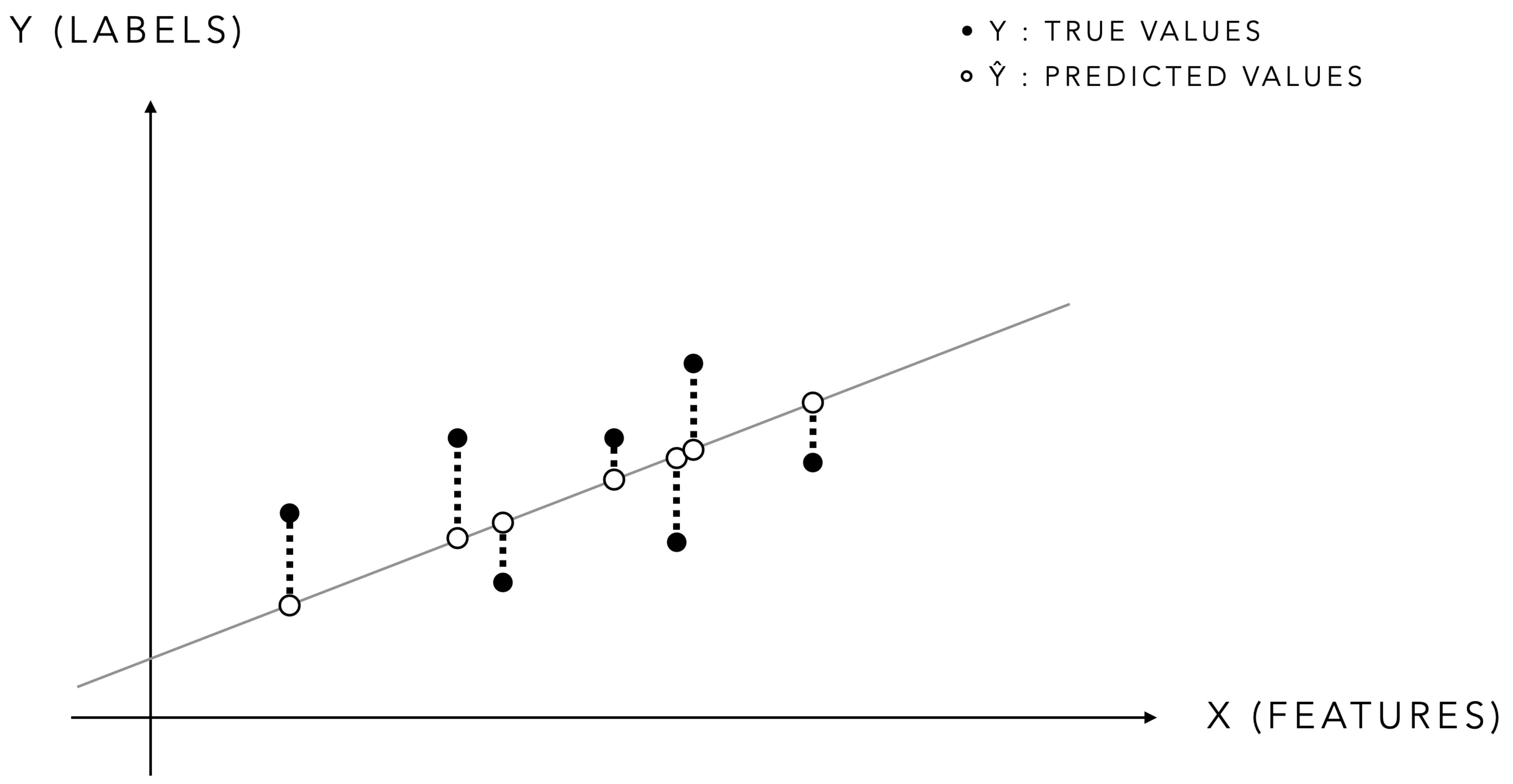
Seems simple right ? Maybe too good to be true ? Well, that's the beauty of Machine Learning, from the outside world, it looks like magic, but really, it is logic! Ok. I know you might be thinking, you know how it works, but I still do not know how we got the coefficients ? Well, lets have a look at that! Now, imagine this, lets say we get the below graph on plotting our data.



Now lets take an equation and plot it on the above graph. Initially, we will assume our coefficients to be some random value. Lets say we got the below graph on plotting our equation along with the data points



Now, for the same values of X, lets see what are the values given by our linear equation. We will call them ŷ (read as 'y hat')



Perfect! Now as you can see, in some data points, we have huge differences while in some cases we have small differences. These differences are officially called as error as, well, we didn't get what we want really!, below is the formula for error

$$\text{ERROR } (\epsilon) = Y - \hat{Y}$$

Now that we know what error is, how can we use it ? Well, lets see, we want our predictions to be as close as possible, right ? That means, we need the difference between the predicted value and true value to be as minimum as possible, seems right! Now, if we get the coefficients of equation such that the error is minimum, we have found the best Linear Regression model right ?! Hmm, but we still have to address one issue, see that we have error as True Value – Predicted Value, now in some cases, it can be that we have Positive Error i.e, $Y > \hat{Y}$ while in some other cases we have Negative Error i.e, $Y < \hat{Y}$. You see where I am going with this ? Well, what if both types of error sum up to become 0. Does it mean we have found the best model ?

No! Its because whether if its a positive magnitude error or a negative magnitude error, an error is an error. So how to solve this ? Well there are two common ways we can do this. We can either

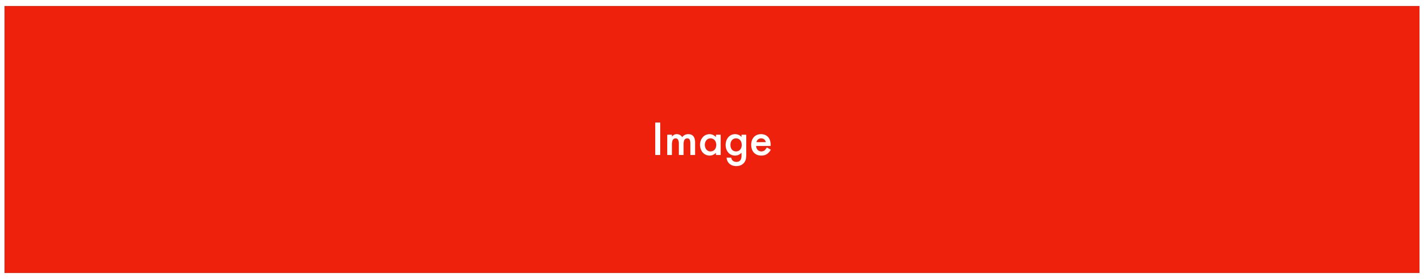
i) Absolute Error : Take the absolute values of error and sum it
(Remember $|x| = |-x| = x$)

OR

ii) Squared Error : Take the sum of square values of error
(Remember $x^2 = (-x)^2 = x^2$)

Generally, people take the sum of the squared error and for that reason, this method of finding the best coefficients for linear regression is known as Ordinary Least Squares. Now if you would like to do this on your own, all you would need to do is run a for loop for each coefficient in Python and for each feature give a range of values. Initially keep error as +ve infinity (you can do this by np.inf from Numpy) and then as you run your loop find the sum of squared error and if it is smaller than the current error, replace it and store the coefficients in another array. Fortunately, the developers at sci-kit learn have made their codes in such a way that the calculations are done behind the scenes and you can later see what these coefficients are!

Now the question is ? Can we use Linear Regression in any situation ? Well, No. Before we apply Linear Regression to a dataset we need it to go through a linear regression "checklist" which would tell us whether it would be smart to use LinearRegression in our dataset. We need this checklist as although the dataset may look to be similar but in reality can have different. A classic example of this is Anscombe's Quartet (image below).



Now there are 5 assumptions that we take in case we want to use Linear Regression, remember, these are for all types of Linear Regression (Linear and Polynomial) :

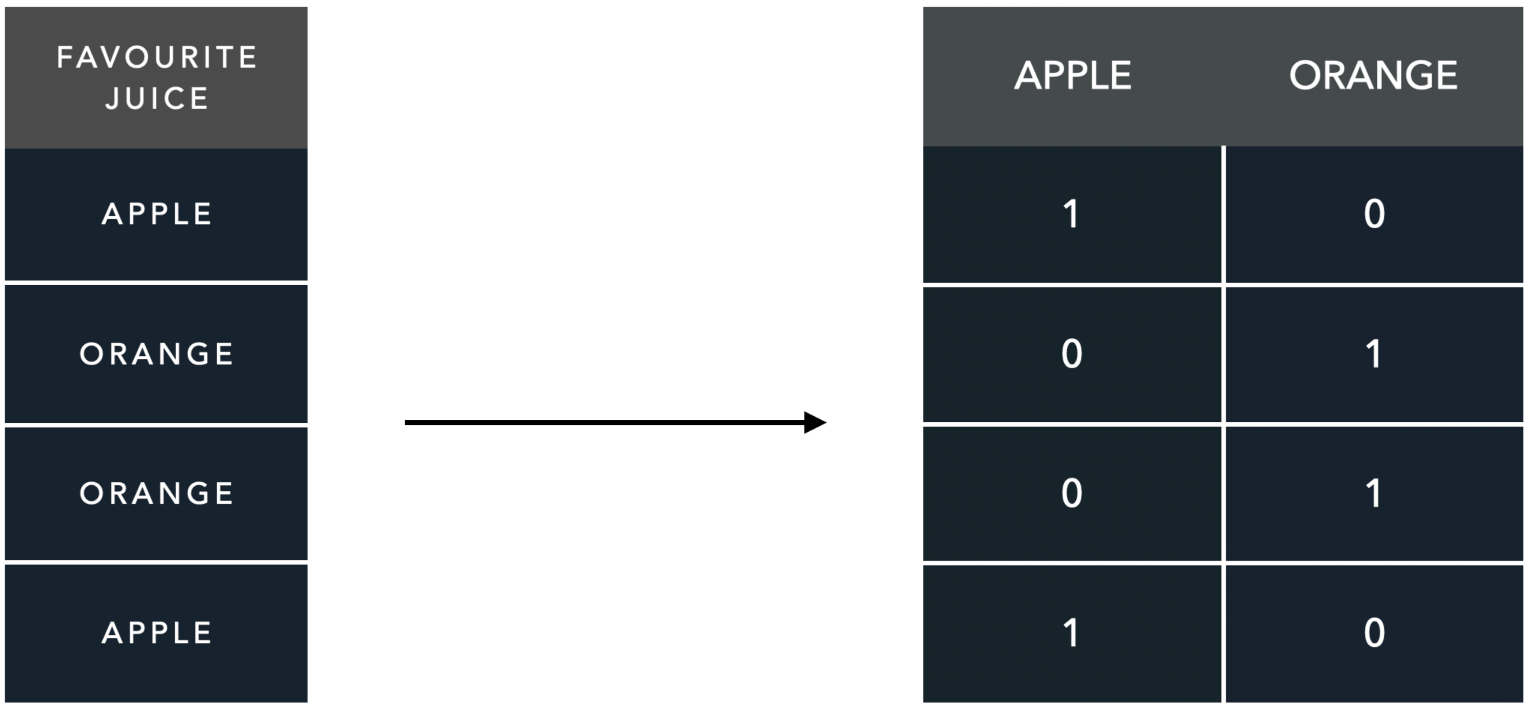
- Linearity : We need to check if the plot is linear (i.e. a linear relationship exists between the feature and its corresponding label) in nature, if not, then we might need to use some other algorithm
- Equal Variance (also known as Homoscedasicity) : Linear Regression is good for situations where the variance is generally very similar (Remember, variance is the measure of how something varies with the general mean and by how much quantity)
- MultiVariate Normality
- Independence : Hello
- Limited MultiCollinearity : One of the most critical assumption, we assume that our features are uncorrelated This rule is often violated in cases where we need dummy variables (One Hot Encoding, to be precise) and causes Dummy Variable Trap (to be discussed later)

Along with the above 5 assumptions we sometimes also check for any outliers as they can heavily influence the trend of the general equation. We can deal with Outliers by the simplest method of removing them, but sometimes that's not advisable as outliers contain some unique information.

Now that we have seen our top 5 assumptions, lets address the topic of Dummy Variable Trap and how we can avoid it.

DUMMY VARIABLE TRAP

Dummy Variables are created as a result of One Hot Encoding or Label Encoding in order to represent Categorical Variables in the equation making by models. Now as seen in OneHotEncoding (see in Chapter 2), each unique category corresponds to a new column. You might think, that it is now ready to use, however, you would find the despite a gut feeling that LinearRegression should perform well, the accuracy would be very low. So is did you use a bad model ? No! Its simply because your dataset has a flaw of one of the assumptions we take while considering Linear Regression. Now you may ask : How so ? Well, let me explain, lets say in a sample feature of favourite juice, we get the following conversion on OneHotEncoding



FAVOURITE JUICE
APPLE
ORANGE
ORANGE
APPLE

APPLE	ORANGE
1	0
0	1
0	1
1	0

Seems normal right ? So where is the problem, well, lets put it this way. Imagine that I tossed a coin, and tell you that the output is **Not Heads** and then ask you, what is the output of the coin toss. You would say that it is obviously Tails and if I ask you to explain why should it be Tails, your probable explanation would be that a coin only has two sides, Heads and Tails and since I told you that the output is Not Heads, it must be Tails!

My Point ? We only need one variable to represent the output of a coin toss, say, if the value of output variable is 1 then the output is Heads and if the variable is 0 then it is Tails. Now our Favourite Juice scenario is similar to that of the Coin Toss, we only need one variable to represent what is the favourite juice of the subject.

Now to look at this in a more Machine Learning way, I mentioned in the previous sub section that Dummy Variable Trap is the phenomenon of violation of Limited Collinearity (or in other words violation of the fact that features are uncorrelated). You might ask, how is this. Well, lets say that we have the below linear equation

$$Y = B_1 \underset{\text{APPLE}}{X_1} + B_2 \underset{\text{ORANGE}}{X_2} + B_3$$

Now that we have also related the fact of Coin Toss logic being similar to that of the Favourite Juice situation, we also the below equation

$$\underset{\text{APPLE}}{X_1} = 1 - \underset{\text{ORANGE}}{X_2}$$

From the above equation, we can say that this is a direct violation of the assumption of limited multicollinearity as X_1 can be predicted by X_2 and vice versa, making it a difficult task to understand the coefficients and constants of variables, thus we are able to rewrite the equation with only variable, resulting in below equation

$$Y = (B_2 - B_1) X_2 + B_1 + B_3$$

Now you may ask, how can we avoid this trap. Well, its easy really. All you need to do in order to **Avoid Dummy Variable Trap is to Drop One of the Columns formed as a result of OneHotEncoding.** Simple right ? And just like that we have avoided a huge boulder in our way of accuracy

NOTE

You would need to drop columns for every set of OneHotEncoding. For example, in the original dataset if we have two sets of Categorical Columns (ex Gender and City) then in total you would need to drop two columns, one of the Gender Categorical and the other of the City Categorical Column (it does not matter which column we drop - as observed from the coin tossing logic!)

POLYNOMIAL LINEAR MODELS

Lets have a look at Polynomial Linear Models, which I can understand can