# FOOD DEMAND FORECASTING PROJECT

**A Project Report for Industrial Training and Internship**

**submitted by**

**DIBYAMOY ROY**

***In the partial fulfillment of the award of the degree of***

**B. Tech**

in the Department of

**ELECTRONICS & COMMUNICATION ENGINEERING**

**Of**

**B.P. PODDAR INSTITUTE OF MANAGEMENT AND TECHNOLOGY**

At

**Ardent Computech Pvt. Ltd.**

ARDENT COMPUTECH PVT. LTD.
High-End Technology Training and Project

Ardent Computech Pvt. Ltd.
*Drives you to the Industry*
Module 132, SDF Building, Sector V, Salt Lake, Pin - 700091
www.ardentcollaborations.com

ARDENT
COMPUTECH PVT. LTD.
High-End Technology Training and Project

**Ardent Computech Pvt. Ltd.**
***Drives you to the Industry***
Module 132, SDF Building, Sector V, Salt Lake, Pin - 700091
www.ardentcollaborations.com

ISO 9001:2015 CERTIFIED

UDYAM REGISTRATION
MSME
N·E·A·T
National Educational Alliance for Technology

**CERTIFICATE FROM SUPERVISOR**

This is to certify that ***“Dibyamoy Roy”*** have completed the project titled " **FOOD DEMAND FORECASTING PROJECT**" under my supervision during the period from “04-07-2025” to “15-08-2025” which is in partial fulfillment of requirements for the award of the **B. Tech** degree and submitted to the Department of “**ELECTRONICS & COMMUNICATION ENGINEERING**” of **“B.P. PODDAR INSTITUTE OF MANAGEMENT AND TECHNOLOGY”**.

____________________________

**Signature of the Supervisor**

**Date: / /**

**Name of the Project Supervisor: SOURAV GOSWAMI**

ARDENT COMPUTECH PVT. LTD.
High-End Technology Training and Project

**Ardent Computech Pvt. Ltd.**
***Drives you to the Industry***
Module 132, SDF Building, Sector V, Salt Lake, Pin - 700091
www.ardentcollaborations.com

CERTIFIED ISO 9001:2015 CERTIFIED

UDYAM REGISTRATION MSME

N·E·A·T
National Educational Alliance for Technology

## BONAFIDE CERTIFICATE

Certified that this project work was carried out under my supervision

***"FOOD DEMAND FORECASTING PROJECT"*** is the bonafide work of

***Name of the student: DIBYAMOY ROY***

***Signature:*** Dibyamoy Roy

**SIGNATURE** __________________

Name: **SOURAV GOSWAMI**

**PROJECT MENTOR**

**SIGNATURE**

**Name:**

**EXAMIES**

**Ardent Original Seal**

ARDENT
COMPUTECH PVT. LTD.
High-End Technology Training and Project

**Ardent Computech Pvt. Ltd.**
***Drives you to the Industry***
Module 132, SDF Building, Sector V, Salt Lake, Pin - 700091
www.ardentcollaborations.com

ISO 9001:2015 CERTIFIED

UDYAM REGISTRATION MSME

N·E·A·T
National Educational Alliance for Technology

## ACKNOWLEDGEMENT

The achievement that is associated with the successful completion of any task would be incomplete without mentioning the names of those people whose endless cooperation made it possible. Their constant guidance and encouragement made all our efforts successful.

We take this opportunity to express our deep gratitude towards our project mentor, ***Sourav Goswami*** for giving such valuable suggestions, guidance and encouragement during the development of this project work.

Last but not the least we are grateful to all the faculty members of **Ardent Computech Pvt. Ltd.** for their support.

## ABSTRACT

Artificial Intelligence is finding its way into ever more areas of life. The latest craze is AI chips and related applications on smartphones. However, the technology dates back as early as the 1950s with the **Dartmouth Summer Research Project on Artificial Intelligence** at Dartmouth College, USA.

The beginnings go even further back to the work of **Alan Turing**—which includes the well-known *Turing Test*—as well as **Allen Newell** and **Herbert A. Simon**. With the chess computer **Deep Blue** from IBM, which succeeded in 1996 as the first machine to beat the then-reigning chess world champion **Garry Kasparov** in a match, artificial intelligence gained significant attention from the global public.

In data centres and on mainframes, AI algorithms have been used for many years.

# Table of Contents

# INTRODUCTION

In recent years, incredible progress has been made in computer science and AI. Watson, Siri or Deep Learning show that AI systems are now delivering services that must be considered intelligent and creative. And there are fewer and fewer companies today that can do without artificial intelligence if they want to optimize their business or save money. AI systems are undoubtedly very useful. As the world becomes more complex, we need to leverage our human resources and high-quality computer systems help. This also applies to applications that require intelligence.

The other side of the AI medal is: The possibility that a machine might possess intelligence scares many. Most people believe that intelligence is something unique, which is what distinguishes *Homo sapiens*. But if intelligence can be mechanized, what is unique about humans and what sets it apart from the machine? The quest for an artificial copy of man and the complex of questions involved are not new. The reproduction and imitation of thought already occupied our ancestors.

From the sixteenth century, it was teeming with legends and the reality of artificial creatures. Homunculi, mechanical automata, the golem, the Mälzel chess automaton, or Frankenstein were all imaginative or real attempts in the past centuries to artificially produce intelligences and to imitate what is essential to us. The idea of making inanimate objects into intelligent beings by giving life along time is fascinating the mind of mankind. Ancient Greeks had myths about robotics, and Chinese and Egyptian engineers made automatons.

We can see the traces of the beginning of modern artificial intelligence as an attempt to define the classical philosophers‘ system of human thought as a symbolic system. However, the field of artificial intelligence was not formally established until 1956. In 1956, a conference "Artificial Intelligence" was held for the first time in Hanover, New Hampshire, at Dartmouth College. Cognitive scientist Marvin Minsky at MIT and other scientists participating in the conference were quite optimistic about the future of artificial intelligence.

As Minsky stated in his book *AI: The Tumultuous Search for Artificial Intelligence*:
―In a generation, the problem of artificial intelligence creation will be solved at a significant level.‖

One of the most important visionaries and theoreticians was Alan Turing (1912–1954): in 1936, the British mathematician proved that a universal calculator—now known as the Turing machine—is possible. Turing‘s central insight is that such a machine is capable of solving any problem as long as it can be represented and solved by an algorithm. Transferred to human intelligence, this means that if cognitive processes can be broken down into finite well-defined individual steps, they can be executed on one machine.

A few decades later, the first practical digital computers were actually built. Thus, the "physical vehicle" for artificial intelligence was available. The electromechanical machine of Turing, considered a precursor of modern computers, managed to unlock the code used by the German submarines in the Atlantic. His work at Bletchley Park is considered key to the end of World War II.

His work at Bletchley Park, an isolated country house north of London, was made public in the 1970s, when the role of the brilliant mathematician in the war was revealed. The cryptographers who worked helped shorten World War II by about two years, by deciphering around 3,000 German military messages a day. Turing's team deciphered the *Enigma* code, which the Germans considered unbreakable, and designed and developed *Colossus*, one of the first programmable computers.

## HISTORY OF ARTIFICIAL INTELLEGENCE (AI)

To be informed about the history of artificial intelligence, it is necessary to go back to previous dates in Milat. In the Ancient Greek era, it is proven that various ideas about humanoid robots have been carried out. An example of this is Daedelus, who is said to have ruled the mythology of the wind, to try to create artificial humans. Modern artificial intelligence has begun to be seen in history with the aim of defining philosophers' system of human thought. 1884 is very important for artificial intelligence. Charles Babbage, on this date, has worked on a mechanical machine that will exhibit intelligent behavior. However, as a result of these studies, he decided that he would not be able to produce a machine that would exhibit as intelligent behaviors as a human being, and he took his work suspended.

In 1950, Claude Shannon introduced the idea that computers could play chess. Work on artificial intelligence continued slowly until the early 1960s. The emergence of artificial intelligence officially in history dates back to 1956. In 1956, a conference artificial intelligence session at Dartmouth College was introduced for the first time. Marvin Minsky stated in his book *"Stormed Search for Artificial Intelligence"* that *"the problem of artificial intelligence modelling within a generation will be solved."*

The first artificial intelligence applications were introduced during this period. These applications are based on logic theorems and chess game. The programs developed during this period were distinguished from the geometric forms used in the intelligence tests; which has led to the idea that intelligent computers can be created.

## MILESTONES OF AI HISTORY

In 1950, Alan Turing created a test to determine whether a machine was intelligent. This test shows the intelligence given to computers. The intelligence level of the machines that passed the test at that time was considered adequate. LISP (List Processing Language), developed by John McCarthy in 1957, is a functional programming language developed for artificial intelligence. One of the rather old and powerful programming languages, LISP is a language that allows you to create flexible programs that represent basic operations with list structure.

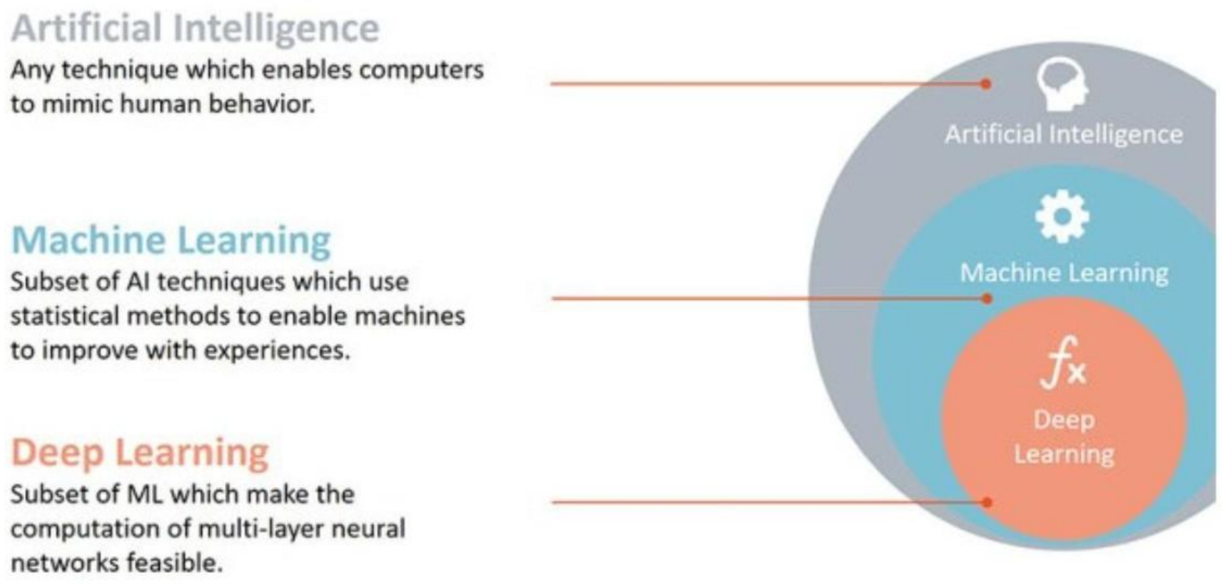
Between 1965 and 1970, it could be called a dark period for artificial intelligence. The developments on artificial intelligence in this period are too few to be tested. The hasty and optimistic attitude due to the unrealistic expectations that have emerged has led to the idea that it will be easy to uncover the machines with intelligence. But this period was named as a dark period on behalf of artificial intelligence because it did not succeed with the idea of creating intelligent machines by simply uploading data.

Between 1970 and 1975, artificial intelligence gained momentum. Thanks to the success achieved in artificial intelligence systems that have been developed and developed on subjects such as disease diagnosis, the basis of today's artificial intelligence has been established. During the period 1975–1980, they developed the idea that they could benefit artificial intelligence through other branches of science such as psychology.

Artificial Intelligence began to be used in large projects with practical applications in the 1980s. The next time the daylight is passed, the artificial intelligence has been adapted to solve real-life problems. Even when the needs of users are already met with traditional methods, the use of artificial intelligence has reached to a much wider range thanks to more economical software and tools.

The concept of Machine Intelligence emerging with various code algorithms and data studies reveals that all the technological devices produced from the first computers to today's smartphones are developed on the basis of people. The artificial intelligence, which was developed very slowly in the old periods but so important steps as the day-to-day, reveals how much progress has been made with the emergence of gifted robots today.

## What is an Artificial Intelligence?



Artificial Intelligence
Any technique which enables computers to mimic human behavior.
Machine Learning
Subset of AI techniques which use statistical methods to enable machines to improve with experiences.
Deep Learning
Subset of ML which make the computation of multi-layer neural networks feasible.
Artificial Intelligence
Machine Learning
Deep Learning

According to the father of Artificial Intelligence, John McCarthy, it is ―The science and engineering of making intelligent machines, especially intelligent computer programs‖.

Artificial intelligence is the general name of the technology for the development of machines, which are start at the beginning and could not be maintained as expected on both sides. In the symbolic artificial intelligence studies, robots cannot give exactly the expected responses and answers to the questions of the people, whereas on the cybernetic artificial intelligence side, the artificial neural networks do not give the expectation and the works on the two sides cannot be successful with literally. Artificial intelligence has led to the emergence of specialized artificial intelligence exercises that will continue with a single purpose, rather than different branches and minds, after failures in Symbolic and Cybernetic artificial intelligence studies developed on different sides.

While the concept of artificial intelligence has stimulated artificial intelligence studies, the fact that artificial intelligence products do not have enough knowledge about what is being worked on has brought about various problems. However, the artificial intelligence developers who brought rational solutions to the problems that have arisen, have reached to commercial level of artificial intelligence, and the artificial intelligence industry that emerged in the coming periods has shown that the achievement of successful works is achieved with billion-dollar billets.

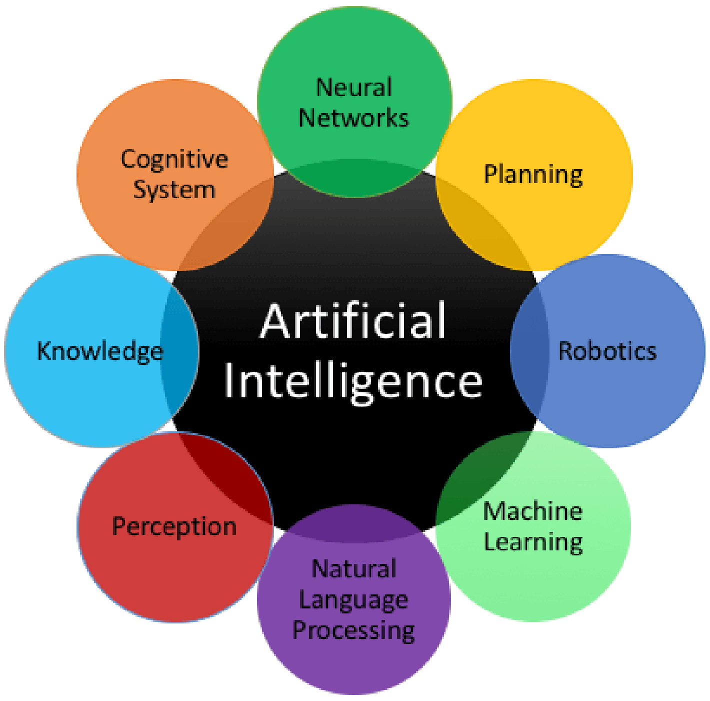
Recent developments in artificial intelligence studies have revealed the importance of language. As anthropology, Human Science studies show, people have begun to hold the language in front of the artificial intelligence studies in recent years because people think with language and put out various functions. Later, a number of artificial intelligences marking languages appeared with the language studies that were behind those who supported Symbolic Artificial Intelligence studies. Today, artificial intelligence studies carried out by Symbolic artificial intelligence developers have benefited from artificial intelligence languages and have made it possible to show even robots that can speak.

## GOALS OF AI

To Create Expert Systems − The systems which exhibit intelligent behaviour, learn, demonstrate, explain, and advice its users.
To Implement Human Intelligence in Machines − Creating systems that understand, think, learn, and behave like humans.

The overall research goal of artificial intelligence is to create technology that allows computers and machines to function in an intelligent manner. The general problem of simulating (or creating) intelligence has been broken down into sub-problems.

These consist of particular traits or capabilities that researchers expect an intelligent system to display. The traits described below have received the most attention. Erik Sandwell emphasizes planning and learning that is relevant and applicable to the given situation.



• **Reasoning, problem solving:** Early researchers developed algorithms that imitated step-by-step reasoning that humans use when they solve puzzles or make logical deductions. By the late 1980s and 1990s, AI research had developed methods for dealing with uncertain or incomplete information, employing concepts from probability and economics. For difficult problems, algorithms can require enormous computational resources—most experience a ―combinatorial explosion‖: the amount of memory or computer time required becomes astronomical for problems of a certain size. The search for more efficient problem-solving algorithms is a high priority.

• **Knowledge representation:** Knowledge representation and knowledge engineering are central to AI research. Many of the problems machines are expected to solve will require extensive knowledge about the world. Among the things that AI needs to represent are: objects, properties, categories and relations between objects; situations, events, states and time; causes and effects; knowledge about knowledge (what we know about what other people know); and many other, less well researched domains. A representation of ―what exists‖ is an ontology: the set of objects, relations, concepts and so on that the machine knows about. The most general are called upper ontologies, which attempt to provide a foundation for all other knowledge.

• **Planning:** Intelligent agents must be able to set goals and achieve them. They need a way to visualize the future – a representation of the state of the world and be able to make predictions about how their actions will change it – and be able to make choices that maximize the utility (or ―value‖) of available choices. In classical planning problems, the agent can assume that it is the only system acting in the world, allowing the agent to be certain of the consequences of its actions. However, if the agent is not the only actor, then it requires that the agent can reason under uncertainty. This calls for an agent that can not only assess its environment and make predictions, but also evaluate its predictions and adapt based on its assessment.

• **Learning:** Machine learning, a fundamental concept of AI research since the field‗s inception, is the study of computer algorithms that improve automatically through experience. Unsupervised learning is the ability to find patterns in a stream of input. Supervised learning includes both classification and numerical regression. Classification is used to determine what category something belongs in, after seeing a number of examples of things from several categories. Regression is the attempt to produce a function that describes the relationship between inputs and outputs and predicts how the outputs should change as the inputs change.

• **General intelligence:** Many researchers think that their work will eventually be incorporated into a machine with artificial general intelligence, combining all the skills mentioned above and even exceeding human ability in most or all these areas. A few believe that anthropomorphic features like artificial consciousness or an artificial brain may be required for such a project.

## Applications of Artificial Intelligence (AI)

AI has been dominant in various fields such as:

**Gaming** − AI plays crucial role in strategic games such as chess, poker, tic-tac-toe, etc., where machine can think of large number of possible positions based on heuristic knowledge.

**Natural Language Processing** − It is possible to interact with the computer that understands natural language spoken by humans.

**Expert Systems** − There are some applications which integrate machine, software, and special information to impart reasoning and advising. They provide explanation and advice to the users.

**Vision Systems** − These systems understand, interpret, and comprehend visual input on the computer.
For example: A spying aeroplane takes photographs, which are used to figure out spatial information or map of the areas. Doctors use clinical expert system to diagnose the patient. Police use computer software that can recognize the face of criminal with the stored portrait made by forensic artist.

**Speech Recognition** − Some intelligent systems are capable of hearing and comprehending the language in terms of sentences and their meanings while a human talks to it. It can handle different accents, slang words, noise in the background, change in human‘s voice due to cold, etc.

**Handwriting Recognition** − The handwriting recognition software reads the text written on paper by a pen or on screen by a stylus. It can recognize the shapes of the letters and convert it into editable text.

**Intelligent Robots** − Robots are able to perform the tasks given by a human. They have sensors to detect physical data from the real world such as light, heat, temperature, movement, sound, bump, and pressure. They have efficient processors, multiple sensors and huge memory, to exhibit intelligence. In addition, they are capable of learning from their mistakes and they can adapt to the new environment.

# MACHINE LEARNING

## Abstract

Machine Learning has always been an integral part of artificial intelligence and its methodology has evolved in concert with the major concerns in the field. In response to the ever-increasing volume of knowledge in modern AI systems, many researchers have turned their attention to machine learning as a way to overcome the knowledge acquisition bottleneck. Together with many other disciplines, machine learning methods have been widely employed in bioinformatics. The difficulties and cost of biological analyses have led to the development of sophisticated machine learning approaches for this application area. In this chapter, we first review the fundamental concepts of machine learning such as feature assessment, unsupervised versus supervised learning and types of classification. Then, we point out the main issues of designing machine learning experiments and their performance evaluation. Finally, we introduce some supervised learning methods.

## Introduction to Machine Learning

Machines have come a long way since the Industrial Revolution. They continue to fill factory floors and manufacturing plants, but now their capabilities extend beyond manual activities to cognitive tasks that, until recently, only humans were capable of performing. Judging song competitions, driving automobiles, and mopping the floor with professional chess players are three examples of the specific complex tasks machines are now capable of simulating. But their remarkable feats trigger fear among some observers. Part of this fear nestles on the neck of survivalist insecurities, where it provokes the deep-seated question of *what if?* What if intelligent machines turn on us in a struggle of the fittest? What if intelligent machines produce offspring with capabilities that humans never intended to impart to machines? What if the legend of the singularity is true?

The other notable fear is the threat to job security, and if you‘re a truck driver or an accountant, there is a valid reason to be worried. According to the British Broadcasting Company‘s (BBC) interactive online resource *Will a robot take my job?*, professions such as bar worker (77%), waiter (90%), chartered accountant (95%), receptionist (96%), and taxi driver (57%) each have a high chance of becoming automated by the year 2035. [1] But research on planned job automation and crystal ball gazing with respect to the future evolution of machines and artificial intelligence (AI) should be read with a pinch of skepticism. AI technology is moving fast, but broad adoption is still an unchartered path fraught with known and unforeseen challenges. Delays and other obstacles are inevitable.

Nor is machine learning a simple case of flicking a switch and asking the machine to predict the outcome of the Super Bowl and serve you a delicious martini. Machine learning is far from what you would call an out-of-the-box solution. Machines operate based on statistical algorithms managed and overseen by skilled individuals—known as data scientists and machine learning engineers. This is one labor market where job opportunities are destined for growth but where, currently, supply is struggling to meet demand. Industry experts lament that one of the biggest obstacles delaying the progress of AI is the inadequate supply of professionals with the necessary expertise and training.

According to Charles Green, the Director of Thought Leadership at Belatrix Software: ―It‘s a huge challenge to find data scientists, people with machine learning experience, or people with the skills to analyze and use the data, as well as those who can create the algorithms required for machine learning. Secondly, while the technology is still emerging, there are many ongoing developments. It‘s clear that AI is a long way from how we might imagine it.‖

To build and program intelligent machines, you must first understand classical statistics. Algorithms derived from classical statistics contribute the metaphorical blood cells and oxygen that power machine learning. Layer upon layer of linear regression, k-nearest neighbors, and random forests surge through the machine and drive their cognitive abilities. Classical statistics is at the heart of machine learning and many of these algorithms are based on the same statistical equations you studied in high school. Indeed, statistical algorithms were conducted on paper well before machines ever took on the title of artificial intelligence.

## WHAT IS MACHINE LEARNING?

In 1959, IBM published a paper in the *IBM Journal of Research and Development* with an, at the time, obscure and curious title. Authored by IBM‘s Arthur Samuel, the paper invested the use of machine learning in the game of checkers ―to verify the fact that a computer can be programmed so that it will learn to play a better game of checkers than can be played by the person who wrote the program.‖ [3] Although it was not the first publication to use the term ―machine learning‖ per se, Arthur Samuel is widely considered as the first person to coin and define machine learning in the form we now know today. Samuel‘s landmark journal submission, *Some Studies in Machine Learning Using the Game of Checkers*, is also an early indication of *homo sapiens'* determination to impart our own system of learning to man-made machines.

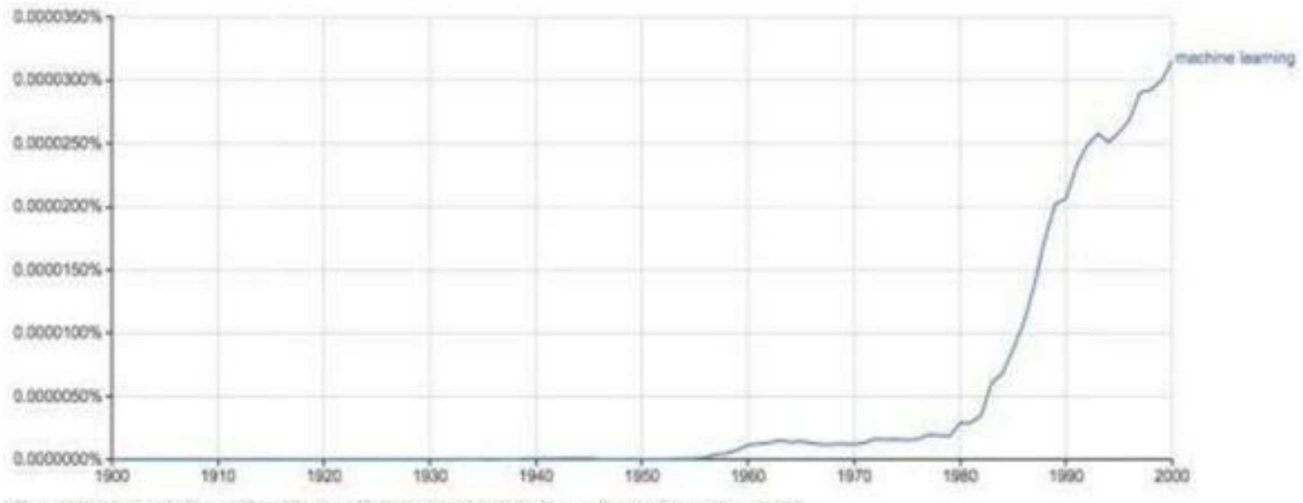


**Figure 1: Historical mentions of "machine learning" in published books.** *Source: Google Ngram Viewer, 2017*

Arthur Samuel introduces machine learning in his paper as a subfield of computer science that gives computers the ability to learn without being explicitly programmed. [4] Almost six decades later, this definition remains widely accepted. Although not directly mentioned in Arthur Samuel‘s definition, a key feature of machine learning is the concept of self-learning. This refers to the application of statistical modelling to detect patterns and improve performance based on data and empirical information; all without direct programming commands. This is what Arthur Samuel described as the ability to learn without being explicitly programmed. But he doesn‘t infer that machines formulate decisions with no upfront programming. On the contrary, machine learning is heavily dependent on computer programming. Instead, Samuel observed that machines don‘t require a direct input command to perform a set task but rather input data.

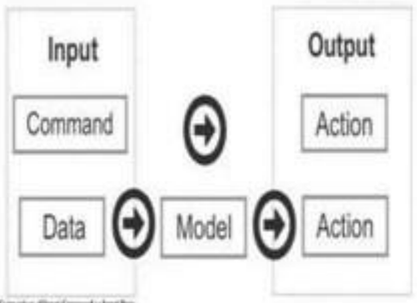


Figure 2: Comparison of Input Command vs Input Data

## TRAINING & TEST DATA

In household energy consumption forecasting, data is typically divided into training data and test data. The training data comprises the initial historical energy usage records used to develop and train the forecasting model. For instance, this could include past hourly or daily electricity consumption, temperature, and appliance usage. The model learns patterns such as peak usage times, seasonal variations, or energy spikes based on this training data.

Once the model is trained and its performance is tuned to an acceptable level, it is evaluated on the test data—new, unseen household consumption records. This ensures the model can generalize and accurately predict future energy usage beyond the data it was trained on. If the model performs well on both the training and test datasets, it can then be deployed to predict and forecast upcoming electricity consumption, assisting users in energy management and cost control.

The distinction between machine learning and traditional rule-based programming becomes clearer in this context. Unlike traditional methods that require explicit instructions for every scenario, machine learning models adapt by learning patterns in energy usage data automatically—making them more flexible and intelligent for real-world forecasting.

Another key concept to grasp is how household energy consumption forecasting fits into the broader scope of data science and computer science. This includes understanding the relationships between machine learning, statistical modeling, and related disciplines such as data mining and artificial intelligence. These fields intersect in various ways when building intelligent systems that can predict future trends from historical energy data.

For example, while machine learning focuses on building predictive models, data mining explores hidden patterns in the dataset, and computer science provides the computational framework for implementing these techniques. Within this larger domain, data science bridges the gap by applying machine learning algorithms to extract actionable insights—such as optimizing energy use and minimizing electricity bills in smart homes.

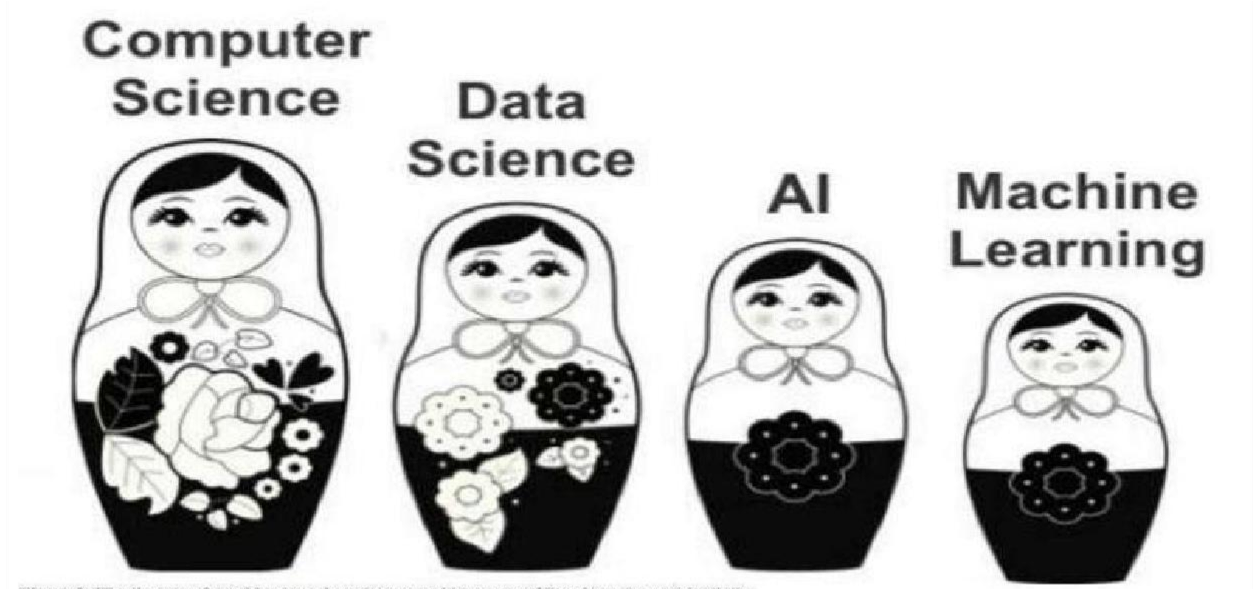


Figure 3: The lineage of machine learning represented by a row of Russian matryoshka dolls

## ML CATEGORIES

Machine learning incorporates several hundred statistical-based algorithms and choosing the right algorithm or combination of algorithms for the job is a constant challenge for anyone working in this field. But before we examine specific algorithms, it is important to understand the three overarching categories of machine learning. These three categories are supervised, unsupervised, and reinforcement.

**Supervised Learning**
As the first branch of machine learning, supervised learning concentrates on learning patterns through connecting the relationship between variables and known outcomes and working with labeled datasets. Supervised learning works by feeding the machine sample data with various features (represented as ―X‖) and the correct value output of the data (represented as ―y‖). The fact that the output and feature values are known qualifies the dataset as ―labeled.‖ The algorithm

then deciphers patterns that exist in the data and creates a model that can reproduce the same underlying rules with new data.

For instance, to predict the market rate for the purchase of a used car, a supervised algorithm can formulate predictions by analyzing the relationship between car attributes (including the year of make, car brand, mileage, etc.) and the selling price of other cars sold based on historical data. Given that the supervised algorithm knows the final price of other cars sold, it can then work backward to determine the relationship between the characteristics of the car and its value.

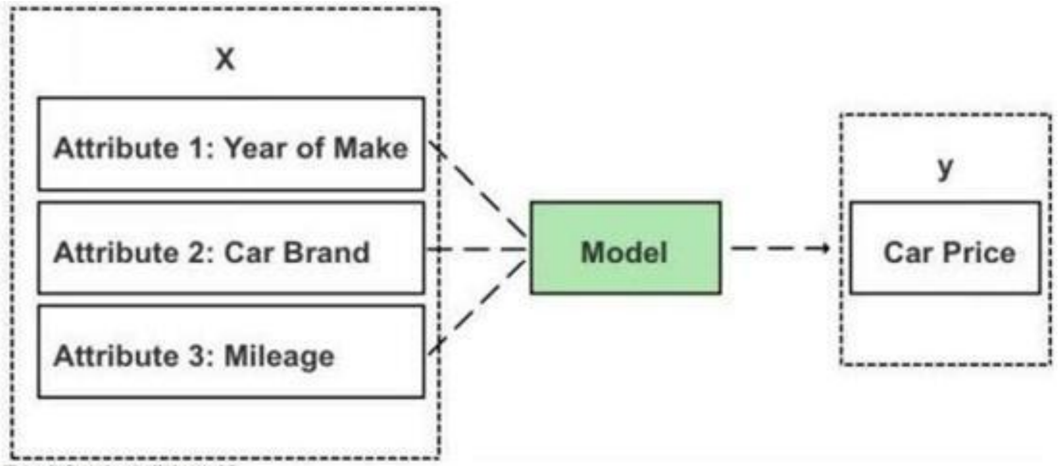


Figure 1: Car value prediction model

After the machine deciphers the rules and patterns of the data, it creates what is known as a model: an algorithmic equation for producing an outcome with new data based on the rules derived from the training data. Once the model is prepared, it can be applied to new data and tested for accuracy. After the model has passed both the training and test data stages, it is ready to be applied and used in the real world. In Chapter 13, we will create a model for predicting house values where **y** is the actual house price and **X** are the variables that impact **y**, such as land size, location, and the number of rooms. Through supervised learning, we will create a rule to predict **y** (house value) based on the given values of various variables (**X**).

Examples of supervised learning algorithms include regression analysis, decision trees, k-nearest neighbours, neural networks, and support vector machines.

## Unsupervised Learning

In the case of unsupervised learning, not all variables and data patterns are classified. Instead, the machine must uncover hidden patterns and create labels through the use of unsupervised learning algorithms. The k-means clustering algorithm is a popular example of unsupervised learning. This algorithm groups data points that are found to possess similar features.

If you group data points based on the purchasing behavior of SME (Small and Medium-sized Enterprises) and large enterprise customers, for example, you are likely to see two clusters emerge. This is because SMEs and large enterprises tend to have disparate buying habits. When it comes to purchasing cloud infrastructure, for instance, basic cloud hosting resources and a Content Delivery Network (CDN) may prove sufficient for most SME customers.

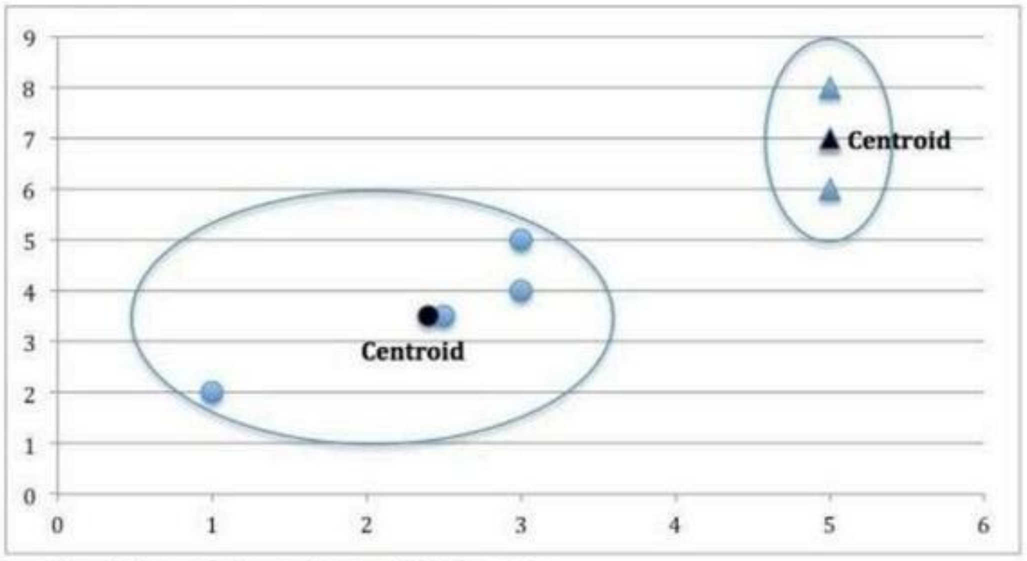


Figure 1: Example of k-means clustering, a popular unsupervised learning technique

The advantage of unsupervised learning is it enables you to discover patterns in the data that you were unaware existed—such as the presence of two major customer types. Clustering techniques such as k-means clustering can also provide the springboard for conducting further analysis after discrete groups have been discovered.

In industry, unsupervised learning is particularly powerful in fraud detection—where the most dangerous attacks are often those yet to be classified. One real-world example is DataVisor, who essentially built their business model based on unsupervised learning.

**Reinforcement Learning**
Reinforcement learning is the third and most advanced algorithm category in machine learning. Unlike supervised and unsupervised learning, reinforcement learning continuously improves its model by leveraging feedback from previous iterations. This is different to supervised and unsupervised learning, which both reach an indefinite endpoint after a model is formulated from the training and test data segments.

Reinforcement learning can be complicated and is probably best explained through an analogy to a video game. As a player progresses through the virtual space of a game, they learn the value of various actions under different conditions and become more familiar with the field of play. Those learned values then inform and influence a player's subsequent behavior and their performance immediately improves based on their learning and past experience.

Reinforcement learning is very similar, where algorithms are set to train the model through continuous learning. A standard reinforcement learning model has measurable performance criteria where outputs are not tagged—instead, they are graded. In the case of self-driving vehicles, avoiding a crash will allocate a positive score and in the case of chess, avoiding defeat will likewise receive a positive score.

A specific algorithmic example of reinforcement learning is Q-learning. In Q-learning, you start with a set environment of states, represented by the symbol ‗S'. In the game Pac-Man, states could be the challenges, obstacles or pathways that exist in the game. There may exist a wall to the left, a ghost to the right, and a power pill above—each representing different states.

The set of possible actions to respond to these states is referred to as ―A.‖ In the case of Pac-Man, actions are limited to left, right, up, and down movements, as well as multiple combinations thereof. The third important symbol is ―Q.‖ Q is the starting value and has an initial value of ―0.‖ As Pac-Man explores the space inside the game, two main things will happen:

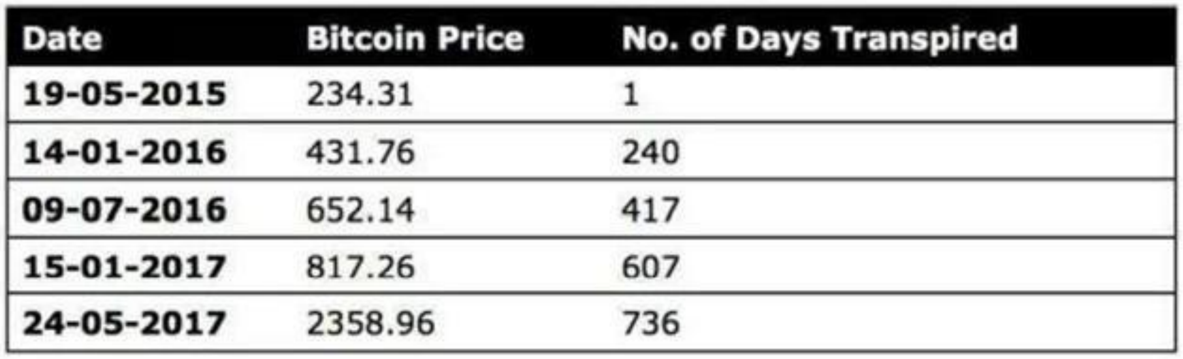
1. Q drops as negative things occur after a given state/action
2. Q increases as positive things occur after a given state/action

In Q-learning, the machine will learn to match the action for a given state that generates or maintains the highest level of Q. It will learn initially through the process of random movements (actions) under different conditions (states). The machine will record its results (rewards and penalties) and how they impact its Q level and store those values to inform and optimize its future actions.

While this sounds simple enough, implementation is a much more difficult task and beyond the scope of an absolute beginner's introduction to machine learning. Reinforcement learning algorithms aren't covered in this book, however, I will leave you with a link to a more comprehensive explanation of reinforcement learning and Q-learning following the Pac-Man scenario.

**The Machine Learning Toolbox**

A handy way to learn a new subject area is to map and visualize the essential materials and tools inside a toolbox. If you were packing a toolbox to build websites, for example, you would first pack a selection of programming languages. This would include frontend languages such as HTML, CSS, and JavaScript, one or two backend programming languages based on personal preferences, and of course, a text editor. You might throw in a website builder such as WordPress and then have another compartment filled with web hosting, DNS, and maybe a few domain names that you‘ve recently purchased. This is not an extensive inventory, but from this general list, you can start to gain a better appreciation of what tools you need to master in order to become a successful website developer.

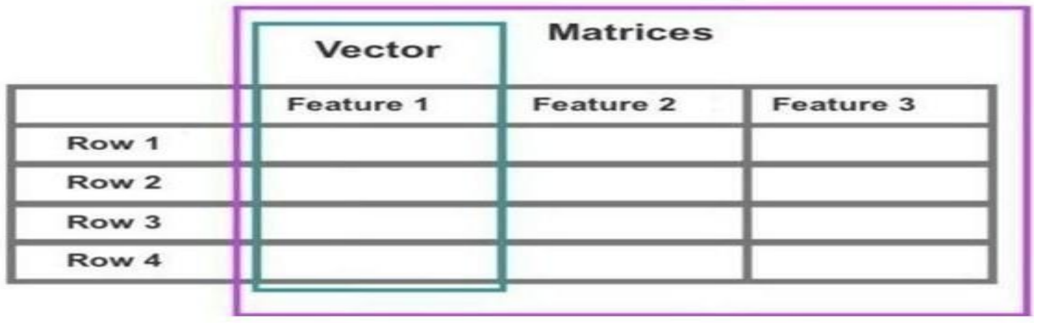


| Date | Bitcoin Price | No. of Days Transpired |
|---|---|---|
| 19-05-2015 | 234.31 | 1 |
| 14-01-2016 | 431.76 | 240 |
| 09-07-2016 | 652.14 | 417 |
| 15-01-2017 | 817.26 | 607 |
| 24-05-2017 | 2358.96 | 736 |

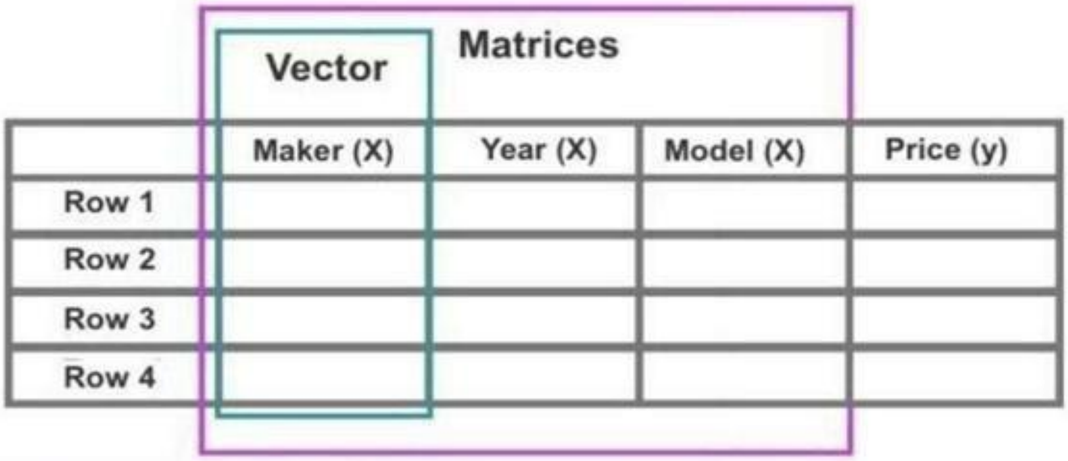
Let‘s now unpack the toolbox for machine learning.

**Compartment 1: Data**
In the first compartment is your data. Data constitutes the input variables needed to form a prediction. Data comes in many forms, including structured and non-structured data. As a beginner, it is recommended that you start with structured data. This means that the data is defined and labeled (with schema) in a table, as shown here:

A tabular (table-based) dataset contains data organized in rows and columns. In each column is a feature. A feature is also known as a variable, a dimension or an attribute—but they all mean the same thing. Each individual row represents a single observation of a given feature/variable. Rows are sometimes referred to as a case or value, but in this book, we will use the term ―row.‖



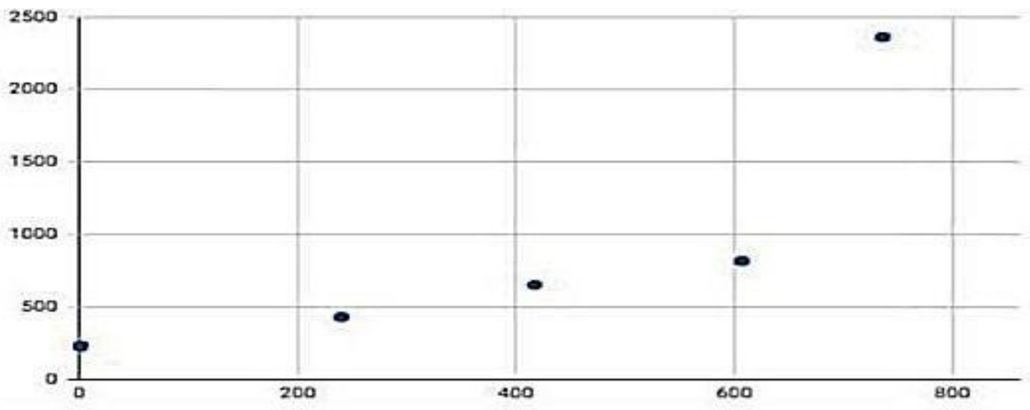
Each column is known as a vector. Vectors store your X and y values and multiple vectors (columns) are commonly referred to as matrices. In the case of supervised learning, y will already exist in your dataset and be used to identify patterns in relation to independent variables (X). The y values are commonly expressed in the final column as shown in figure 2.



| | Maker (X) | Year (X) | Model (X) | Price (y) |
|---|---|---|---|---|
| Row 1 | | | | |
| Row 2 | | | | |
| Row 3 | | | | |
| Row 4 | | | | |

Figure 2: The y value is often but not always expressed in the far right column

Next, within the first compartment of the toolbox is a range of scatterplots, including 2-D, 3-D, and 4-D plots. A 2-D scatterplot consists of a vertical axis (known as the y-axis) and a horizontal axis (known as the x-axis) and provides the graphical canvas to plot a series of dots, known as data points. Each data point on the scatterplot represents one observation from the dataset, with X values plotted on the x-axis and y values plotted on the y-axis.



| | Independent Variable (X) | Dependent Variable (y) |
|---|---|---|
| Row 1 | 1 | 243.31 |
| Row 2 | 240 | 431.76 |
| Row 3 | 417 | 653.14 |
| Row 4 | 607 | 817.26 |
| Row 5 | 736 | 2358.96 |

Figure 3: Example of a 2-D scatterplot. X represents days passed since the recording of Bitcoin prices and y represents recorded Bitcoin price.

**Compartment 2: Infrastructure**
The second compartment of the toolbox contains your infrastructure, which consists of platforms and tools to process data. As a beginner to machine learning, you are likely to be using a web application (such as Jupyter Notebook) and a programming language like Python. There are then a series of machine learning libraries, including NumPy, Pandas, and Scikit-learn that are compatible with Python. Machine learning libraries are a collection of pre-compiled programming routines frequently used in machine learning.

You will also need a machine from which to work, in the form of a computer or a virtual server. In addition, you may need specialized libraries for data visualization such as Seaborn and Matplotlib, or a standalone software program like Tableau, which supports a range of visualization techniques including charts, graphs, maps, and other visual options.

With your infrastructure sprayed out across the table (hypothetically of course), you are now ready to get to work building your first machine learning model. The first step is to crank up your computer. Laptops and desktop computers are both suitable for working with smaller datasets. You will then need to install a programming environment, such as Jupyter Notebook, and a programming language, which for most beginners is Python.

Python is the most widely used programming language for machine learning because:
a) It is easy to learn and operate,
b) It is compatible with a range of machine learning libraries, and
c) It can be used for related tasks, including data collection (web scraping) and data piping (Hadoop and Spark).

Other go-to languages for machine learning include C and C++. If you‘re proficient with C and C++ then it makes sense to stick with what you already know. C and C++ are the default programming languages for advanced machine learning because they can run directly on a GPU (Graphical Processing Unit). Python needs to be converted first before it can run on a GPU, but we will get to this and what a GPU is later in the chapter.

Next, Python users will typically install the following libraries: NumPy, Pandas, and Scikit-learn.

- **NumPy** is a free and open-source library that allows you to efficiently load and work with large datasets, including managing matrices.
- **Scikit-learn** provides access to a range of popular algorithms, including linear regression, Bayes‘ classifier, and support vector machines.
- **Pandas** enables your data to be represented on a virtual spreadsheet that you can control through code. It shares many of the same features as Microsoft Excel in that it allows you to edit data and perform calculations. In fact, the name Pandas derives from the term ―panel data,‖ which refers to its ability to create a series of panels, similar to ―sheets‖ in Excel. Pandas is also ideal for importing and extracting data from CSV files.

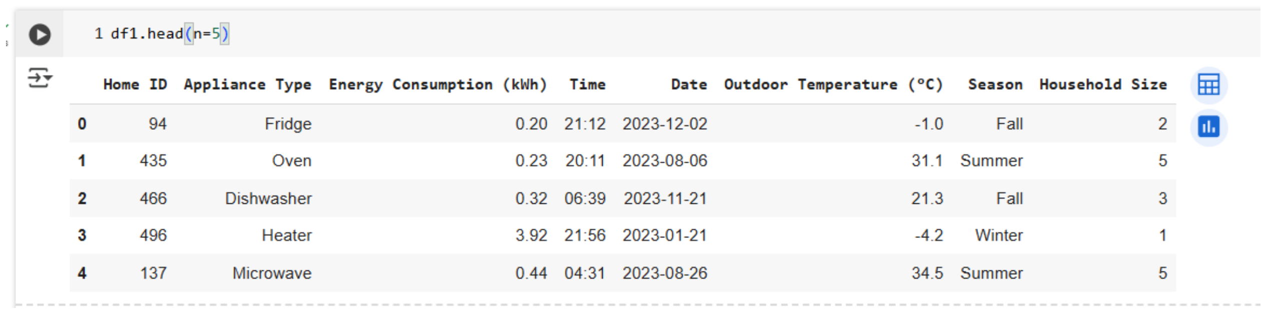
**In summary**, users can draw on these three libraries to:

1. Load and work with a dataset via NumPy.

2. Clean up and perform calculations on data, and extract data from CSV files with Pandas.
3. Implement algorithms with Scikit-learn.

For students seeking alternative programming options (beyond Python, C, and C++), other relevant programming languages for machine learning include **R**, **MATLAB**, and **Octave**.

- **R** is a free and open-source programming language optimized for mathematical operations, and conducive to building matrices and statistical functions, which are built directly into the language libraries of R. Although R is commonly used for data analytics and data mining, R supports machine learning operations as well.
- **MATLAB** is a commercial and proprietary programming language. It is strong in regards to solving algebraic equations and is also a quick programming language to learn. MATLAB is widely used in electrical engineering, chemical engineering, civil engineering, and aeronautical engineering. However, computer scientists and computer engineers tend not to rely on MATLAB as heavily and especially in recent times.
- **Octave** is essentially a free version of MATLAB developed in response to MATLAB by the open-source community.



```
df1.head(n=5)
```

| | Home ID | Appliance Type | Energy Consumption (kWh) | Time | Date | Outdoor Temperature (°C) | Season | Household Size |
|---|---|---|---|---|---|---|---|---|
| 0 | 94 | Fridge | 0.20 | 21:12 | 2023-12-02 | -1.0 | Fall | 2 |
| 1 | 435 | Oven | 0.23 | 20:11 | 2023-08-06 | 31.1 | Summer | 5 |
| 2 | 466 | Dishwasher | 0.32 | 06:39 | 2023-11-21 | 21.3 | Fall | 3 |
| 3 | 496 | Heater | 3.92 | 21:56 | 2023-01-21 | -4.2 | Winter | 1 |
| 4 | 137 | Microwave | 0.44 | 04:31 | 2023-08-26 | 34.5 | Summer | 5 |

**Compartment 3: Algorithms**

Now that the machine learning environment is set up and you've chosen your programming language and libraries, you can next import your data directly from a CSV file. You can find hundreds of interesting datasets in CSV format from kaggle.com. After registering as a member of their platform, you can download a dataset of your choice. Best of all, Kaggle datasets are free and there is no cost to register as a user.

The dataset will download directly to your computer as a CSV file, which means you can use Microsoft Excel to open and even perform basic algorithms such as linear regression on your dataset. Next is the third and final compartment that stores the algorithms.

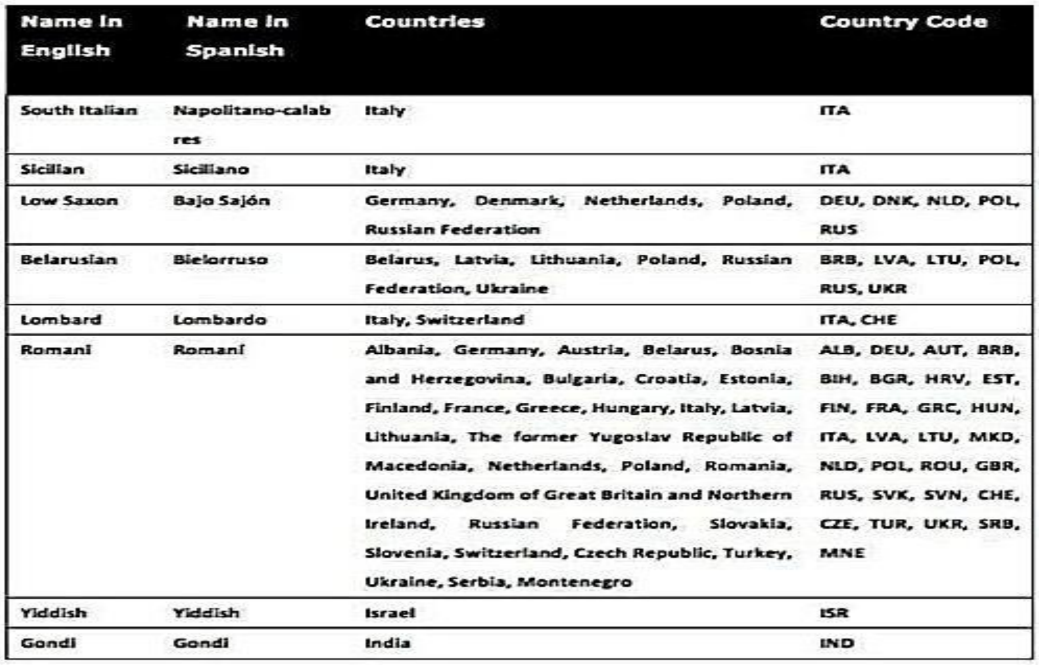
Beginners will typically start off by using simple supervised learning algorithms such as linear regression, logistic regression, decision trees, and k-nearest neighbors. Beginners are also likely to apply unsupervised learning in the form of k-means clustering and descending dimension algorithms.

**Data Scrubbing**
Much like many categories of fruit, datasets nearly always require some form of upfront cleaning and human manipulation before they are ready to digest. For machine learning and data science more broadly, there are a vast number of techniques to scrub data.

**Scrubbing** is the technical process of refining your dataset to make it more workable. This can involve modifying and sometimes removing incomplete, incorrectly formatted, irrelevant or duplicated data. It can also entail converting text-based data to numerical values and the redesigning of features. For data practitioners, data scrubbing usually demands the greatest application of time and effort.

**Feature Selection**
To generate the best results from your data, it is important to first identify the variables most relevant to your hypothesis. In practice, this means being selective about the variables you select to design your model. Rather than creating a four-dimensional scatterplot with four features in the model, an opportunity may present to select two highly relevant features and build a two-dimensional plot that is easier to interpret.

Moreover, preserving features that do not correlate strongly with the outcome value can, in fact, manipulate and derail the model‗s accuracy. Consider the following table excerpt downloaded from kaggle.com documenting dying languages.

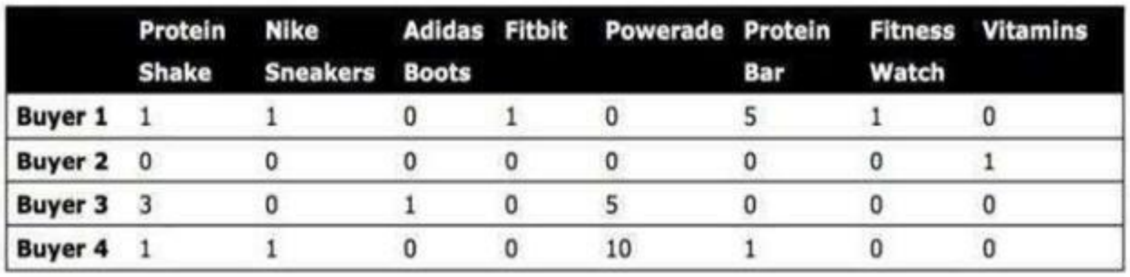


| Name in English | Name in Spanish | Countries | Country Code |
|---|---|---|---|
| South Italian | Napolitano-calab res | Italy | ITA |
| Sicilian | Siciliano | Italy | ITA |
| Low Saxon | Bajo Sajón | Germany, Denmark, Netherlands, Poland, Russian Federation | DEU, DNK, NLD, POL, RUS |
| Belarusian | Bielorruso | Belarus, Latvia, Lithuania, Poland, Russian Federation, Ukraine | BRB, LVA, LTU, POL, RUS, UKR |
| Lombard | Lombardo | Italy, Switzerland | ITA, CHE |
| Romani | Romaní | Albania, Germany, Austria, Belarus, Bosnia and Herzegovina, Bulgaria, Croatia, Estonia, Finland, France, Greece, Hungary, Italy, Latvia, Lithuania, The former Yugoslav Republic of Macedonia, Netherlands, Poland, Romania, United Kingdom of Great Britain and Northern Ireland, Russian Federation, Slovakia, Slovenia, Switzerland, Czech Republic, Turkey, Ukraine, Serbia, Montenegro | ALB, DEU, AUT, BRB, BIH, BGR, HRV, EST, FIN, FRA, GRC, HUN, ITA, LVA, LTU, MKD, NLD, POL, ROU, GBR, RUS, SVK, SVN, CHE, CZE, TUR, UKR, SRB, MNE |
| Yiddish | Yiddish | Israel | ISR |
| Gondi | Gondi | India | IND |

Let‗s say our goal is to identify variables that lead to a language becoming endangered. Based on this goal, it‗s unlikely that a language‗s ―Name in Spanish‖ will lead to any relevant insight. We can therefore go ahead and delete this vector (column) from the dataset. This will help to prevent overcomplication and potential inaccuracies, and will also improve the overall processing speed of the model.

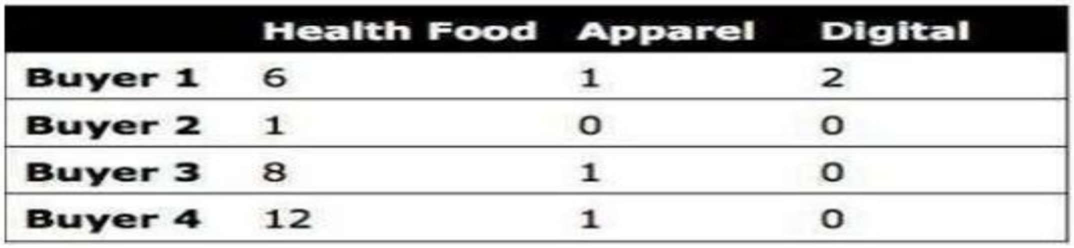
Secondly, the dataset holds duplicate information in the form of separate vectors for —Countries‖ and —Country Code.‖ Including both of these vectors doesn‘t provide any additional insight; hence, we can choose to delete one and retain the other.\

Another method to reduce the number of features is to roll multiple features into one. In the next table, we have a list of products sold on an e-commerce platform. The dataset comprises four buyers and eight products. This is not a large sample size of buyers and products—due in part to the spatial limitations of the book format. A real-life e-commerce platform would have many more columns to work with, but let‘s go ahead with this example.



| | Protein Shake | Nike Sneakers | Adidas Boots | Fitbit | Powerade | Protein Bar | Fitness Watch | Vitamins |
|---|---|---|---|---|---|---|---|---|
| Buyer 1 | 1 | 1 | 0 | 1 | 0 | 5 | 1 | 0 |
| Buyer 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Buyer 3 | 3 | 0 | 1 | 0 | 5 | 0 | 0 | 0 |
| Buyer 4 | 1 | 1 | 0 | 0 | 10 | 1 | 0 | 0 |

In order to analyze the data in a more efficient way, we can reduce the number of columns by merging similar features into fewer columns. For instance, we can remove individual product names and replace the eight product items with a lower number of categories or subtypes. As all product items fall under the single category of —fitness,‖ we will sort by product subtype and compress the columns from eight to three. The three newly created product subtype columns are —Health Food,‖ —Apparel,‖ and —Digital.‖



| | Health Food | Apparel | Digital |
|---|---|---|---|
| Buyer 1 | 6 | 1 | 2 |
| Buyer 2 | 1 | 0 | 0 |
| Buyer 3 | 8 | 1 | 0 |
| Buyer 4 | 12 | 1 | 0 |

This enables us to transform the dataset in a way that preserves and captures information using fewer variables. The downside to this transformation is that we have less information about relationships between specific products. Rather than recommending products to users according to other individual products, recommendations will instead be based on relationships between product subtypes.This enables us to transform the dataset in a way that preserves and captures information using fewer variables.

**Missing Data**
Dealing with missing data is never a desired situation. Imagine unpacking a jigsaw puzzle that you discover has five percent of its pieces missing. Missing values in a dataset can be equally frustrating and will ultimately interfere with your analysis and final predictions.

There are, however, strategies to minimize the negative impact of missing data:

- **Mode Imputation:** Works best with categorical and binary variables.
- **Median Imputation:** Best for integers and continuous variables.
- **Row Deletion:** Only as a last resort, since it reduces the dataset size.

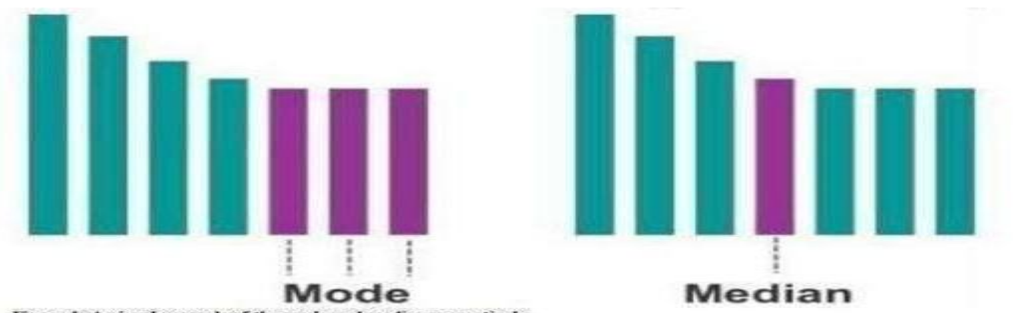
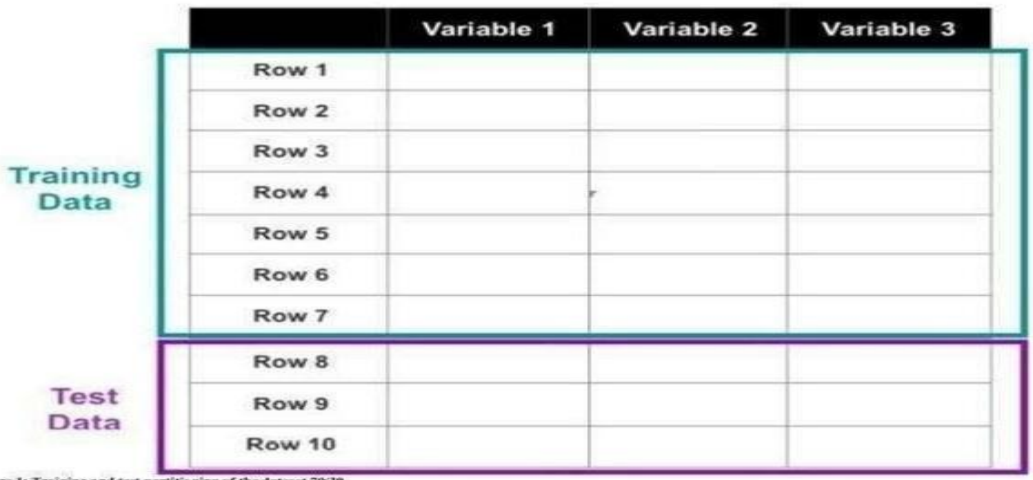


Figure 1: A visual example of the mode and median respectively

**Setting Up Your Data**
Once you have cleaned your dataset, the next job is to split the data into two segments for testing and training. It is very important not to test your model with the same data that you used for training.

The ratio of the two splits should be approximately 70/30 or 80/20. This means that your training data should account for 70 percent to 80 percent of the rows in your dataset, and the other 20 percent to 30 percent of rows is your test data. It is vital to split your data by rows and not columns.



| | | Variable 1 | Variable 2 | Variable 3 |
|---|---|---|---|---|
| Training Data | Row 1 | | | |
| | Row 2 | | | |
| | Row 3 | | | |
| | Row 4 | | | |
| | Row 5 | | | |
| | Row 6 | | | |
| | Row 7 | | | |
| Test Data | Row 8 | | | |
| | Row 9 | | | |
| | Row 10 | | | |

Figure 1: Training and test partitioning of the dataset 70/30

Before you split your data, it is important that you randomize all rows in the dataset. This helps to avoid bias in your model, as your original dataset might be arranged sequentially depending on the time it was collected or some other factor.

Unless you randomize your data, you may accidentally omit important variance from the training data that will cause unwanted surprises when you apply the trained model to your test data. Fortunately, Scikit-learn provides a built-in function to shuffle and randomize your data with just one line of code.

After randomizing your data, you can begin to design your model and apply that to the training data. The remaining 30 percent or so of data is put to the side and reserved for testing the accuracy of the model.

In the case of supervised learning, the model is developed by feeding the machine the training data and the expected output (y). The machine is able to analyze and discern relationships between the features (X) found in the training data to calculate the final output (y).

The next step is to measure how well the model actually performs. A common approach to analyzing prediction accuracy is a measure called **mean absolute error**, which examines each prediction in the model and provides an average error score.

In Scikit-learn, mean absolute error is found using the `model.predict()` function on X (features). Scikit-learn will compare the predictions of the model to the correct outcome and measure its accuracy.

You will know if your model is accurate when the error rate between the training and test dataset is low. This means that the model has learned the dataset‘s underlying patterns and trends.

Once the model can adequately predict the values of the test data, it is ready for use in the wild. If the model fails to accurately predict values from the test data, you will need to check whether the training and test data were properly randomized. Alternatively, you may need to change the model's **hyperparameters**—the algorithm‘s internal settings that influence how it learns and what it prioritizes.

**Cross Validation**
Although the training/test data split can be effective in developing models from existing data, a question mark remains as to whether the model will work on new data. If your existing dataset is too small to construct an accurate model, or if the training/test partition of data is not appropriate, this can lead to poor estimations of performance in the wild. Fortunately, there is an effective workaround for this issue. Rather than splitting the data into two segments (one for training and one for testing), we can implement what is known as cross validation. Cross validation maximizes the availability of training data by splitting data into various combinations and testing each specific combination. Cross validation can be performed through two primary methods. The first method is exhaustive cross validation, which involves finding and testing all possible combinations to divide the original sample into a training set and a test set. The alternative and more common method is non-exhaustive cross validation, known as k-fold validation. The k-fold validation technique involves splitting data into k assigned buckets and reserving one of those buckets to test the training model at each round. To perform k-fold validation, data are first randomly assigned to k number of equal sized buckets. One bucket is then reserved as the test bucket and is used to measure and evaluate the performance of the remaining (k-1) buckets.

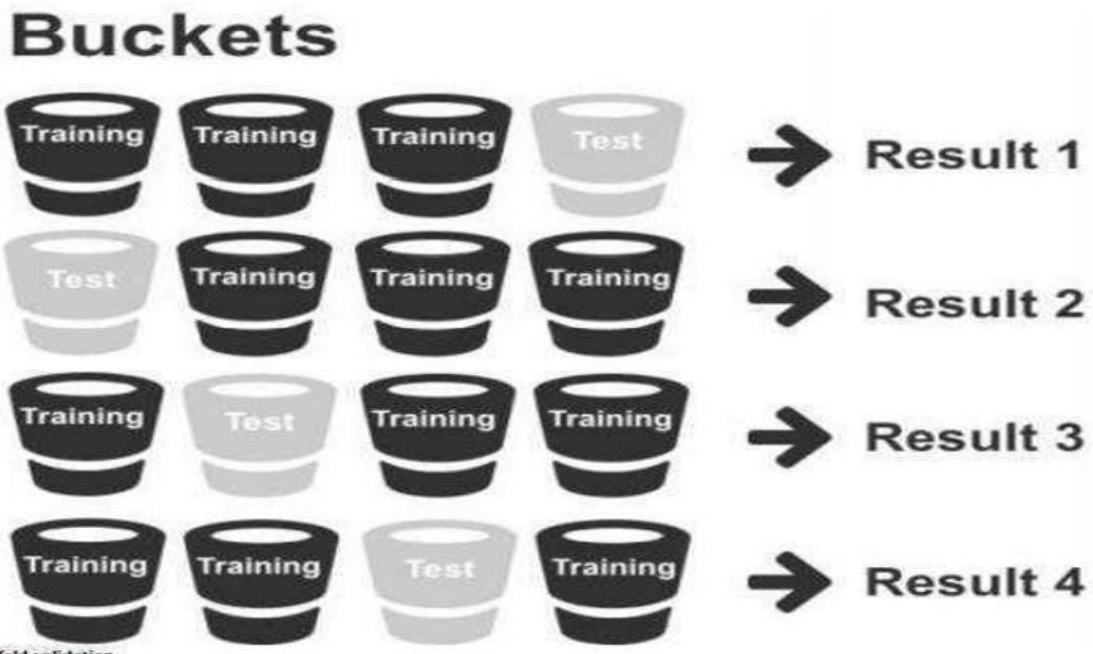


Figure 2: k-fold validation

**Machine Learning Algorithms**

**Supervised Learning**
In supervised learning, the target is to infer a function or mapping from training data that is labelled. The training data consist of input vector X and output vector Y of labels or tags. A label or tag from vector Y is the explanation of its respective input example from input vector X. Together they form a training example. In other words, training data comprises training examples. If the labelling does not exist for input vector X, then X is unlabelled data. Why such learning is called supervised learning? The output vector Y consists of labels for each training example present in the training data. These labels for output vector are provided by the

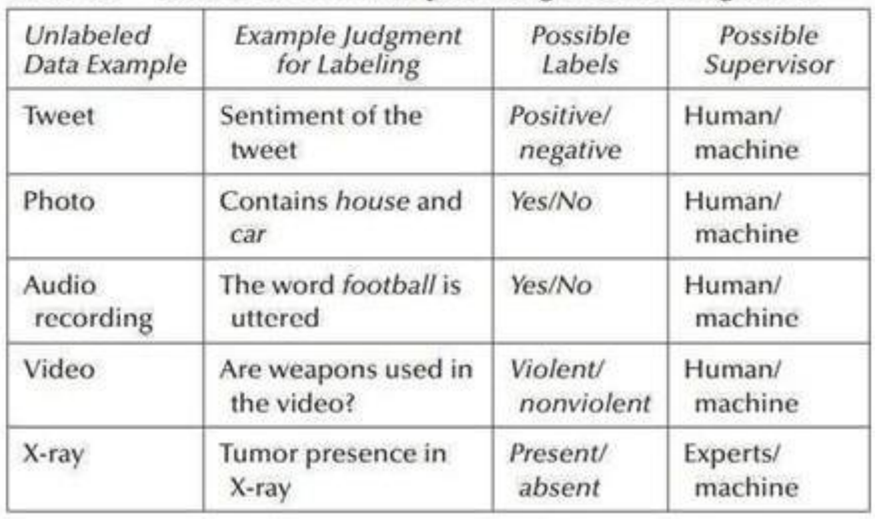
supervisor. Often, these supervisors are humans, but machines can also be used for such labelling. Human judgments are more expensive than machines, but the higher error rates in data labelled by machines suggest superior priority of human judgment. The manually labelled data is a precious and reliable resource for supervised learning. However, in some cases, machines can be used for reliable labelling.

**Example**
Table 1.1 demonstrates five unlabeled data examples that can be labeled based on different criteria. The second column of the table titled, ―Example judgement for labeling‖ expresses possible criterion for each data example. The third column describes possible labels after the application of judgment. The fourth column informs which actor can take the role of the supervisor. In all first four cases described in Table 1.1, machines can be used, but their low accuracy rates make their usage questionable. Sentiment analysis, image recognition, and speech detection technologies have made progress in past three decades but there is still a lot of room for improvement before we can equate them with humans' performance. In the fifth case of tumor detection, even normal humans cannot label the X-ray data, and expensive experts' services are required for such labeling.

Two groups or categories of algorithms come under the umbrella of supervised learning. They are:

1. **Regression**
2. **Classification**

**Table 1.1 Unlabeled Data Examples along with Labeling Issues**



| *Unlabeled Data Example* | *Example Judgment for Labeling* | *Possible Labels* | *Possible Supervisor* |
|---|---|---|---|
| Tweet | Sentiment of the tweet | *Positive/ negative* | Human/ machine |
| Photo | Contains *house* and *car* | *Yes/No* | Human/ machine |
| Audio recording | The word *football* is uttered | *Yes/No* | Human/ machine |
| Video | Are weapons used in the video? | *Violent/ nonviolent* | Human/ machine |
| X-ray | Tumor presence in X-ray | *Present/ absent* | Experts/ machine |

**Unsupervised Learning**
In unsupervised learning, we lack supervisors or training data. In other words, all what we have is unlabeled data. The idea is to find a hidden structure in this data. There can be a number of reasons for the data not having a label. It can be due to unavailability of funds to pay for manual

labeling or the inherent nature of the data itself. With numerous data collection devices, now data is collected at an unprecedented rate. The variety, velocity, and the volume are the dimensions in which Big Data is seen and judged. To get something from this data without the supervisor is important. This is the challenge for today‘s machine learning practitioner.

**Semi-Supervised Learning**
In this type of learning, the given data are a mixture of classified and unclassified data. This combination of labeled and unlabeled data is used to generate an appropriate model for the classification of data. In most of the situations, labeled data is scarce and unlabeled data is in abundance (as discussed previously in unsupervised learning description). The target of semi-supervised classification is to learn a model that will predict classes of future test data better than that from the model generated by using the labeled data alone. The way we learn is similar to the process of semi-supervised learning. A child is supplied with:

1. **Unlabeled data provided by the environment.** The surroundings of a child are full of unlabeled data in the beginning.
2. **Labeled data from the supervisor.** For example, a father teaches his children about the names (labels) of objects by pointing toward them and uttering their names.

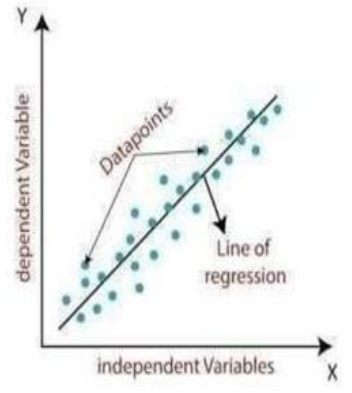
**Reinforcement Learning**
The reinforcement learning method aims at using observations gathered from the interaction with the environment to take actions that would maximize the reward or minimize the risk. In order to produce intelligent programs (also called agents), reinforcement learning goes through the following steps:

1. Input state is observed by the agent.
2. Decision making function is used to make the agent perform an action.
3. After the action is performed, the agent receives reward or reinforcement from the environment.
4. The state-action pair information about the reward is stored. Using the stored information, policy for particular state in terms of action can be fine-tuned, thus helping in optimal decision making for our agent.

**Linear Regression**
Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (x) variables, hence called as linear regression.

Here,
Y = Dependent Variable (Target Variable)
X = Independent Variable (Predictor Variable)
$a_0$ = Intercept of the line (Gives an additional degree of freedom)
$a_1$ = Linear regression coefficient (scale factor to each input value)
ε = Random error

The values for x and y variables are training datasets for Linear Regression model representation.



**Lasso Regression**
Lasso Regression (Least Absolute Shrinkage and Selection Operator) is a regularized version of linear regression that improves the model's performance by adding a penalty term. It is used when there is multicollinearity or when we want to reduce the number of features in a model. Lasso helps in feature selection by shrinking the coefficients of less important features to zero.

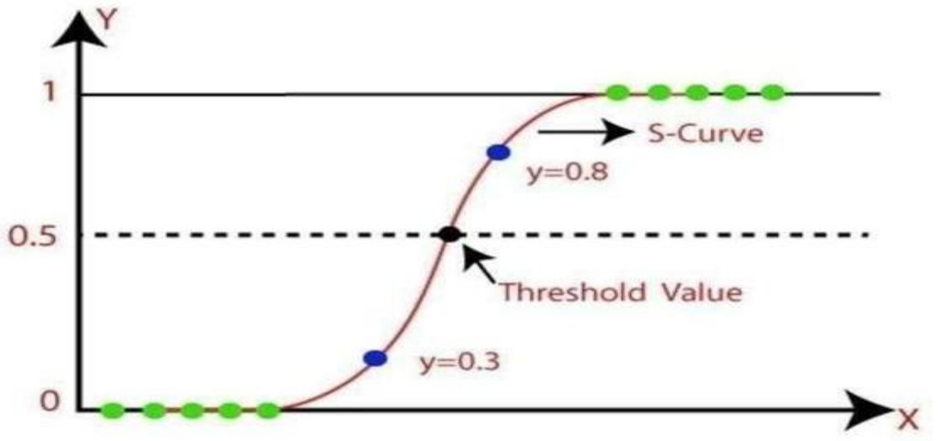
Here,
Y = Dependent Variable (Target Variable)
X = Independent Variable (Predictor Variable)
$a_0$ = Intercept of the line
$a_1$, $a_2$, ..., an = Coefficients for each predictor variable
λ = Regularization parameter (controls the strength of penalty)
ε = Random error

Lasso Regression minimizes the following cost function:
**Cost = RSS + λ * Σ|$a_i$|**
Where,

- RSS = Residual Sum of Squares
- $\Sigma|a_i|$ = Sum of absolute values of coefficients

The values for X and Y variables are training datasets for Lasso Regression model representation. The key feature of Lasso is its ability to perform **feature selection** by driving some coefficients to **zero**, thereby eliminating unnecessary predictors and reducing model complexity.

**Logistic Regression**
Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes. Mathematically, a logistic regression model predicts P(Y=1) as a function of X.
Logistic regression predicts the output of a categorical dependent variable. Therefore, the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.



**Logistic Sigmoid Function (Sigmoid Function)**

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.
- Equation :-

$$y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

## Types of Logistic Regression

On the basis of the categories, Logistic Regression can be classified into three types:

- **Binomial**: In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.
- **Multinomial**: In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep".
- **Ordinal**: In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

## Ridge Regression

Ridge regression is a type of linear regression that includes a regularization technique to reduce model complexity and prevent overfitting. It is especially useful when the dataset has multicollinearity or when the number of predictors is greater than the number of observations.

Ridge regression adds a penalty term to the cost function of linear regression, which is the sum of the squares of the coefficients. This penalty term helps shrink the coefficients, making the model more robust.

Here,
Y = Dependent Variable (Target Variable)
X = Independent Variable (Predictor Variable)
$a_0$ = Intercept of the line (Gives an additional degree of freedom)
$a_1, a_2, ..., a_n$ = Ridge regression coefficients (weights for each input variable)
$\lambda$ = Regularization parameter (controls the amount of shrinkage)
$\varepsilon$ = Random error

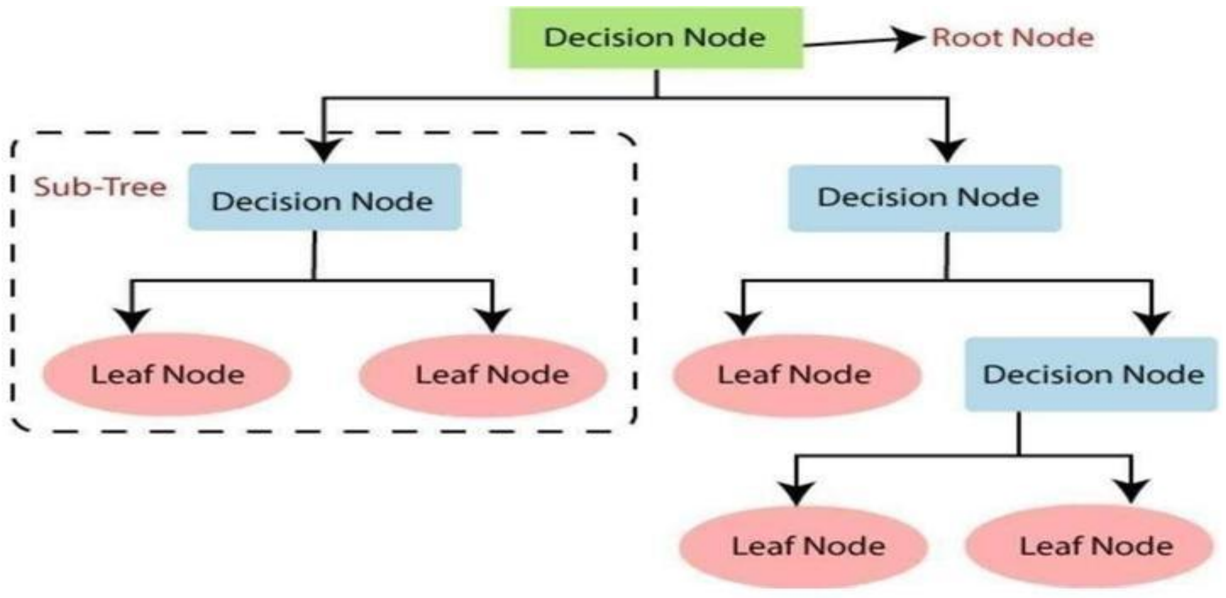
The cost function in ridge regression is:
**Loss = (Sum of squared residuals) + λ * (Sum of squared coefficients)**

The values for x and y variables are training datasets for Ridge Regression model representation.

## Decision Tree

- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

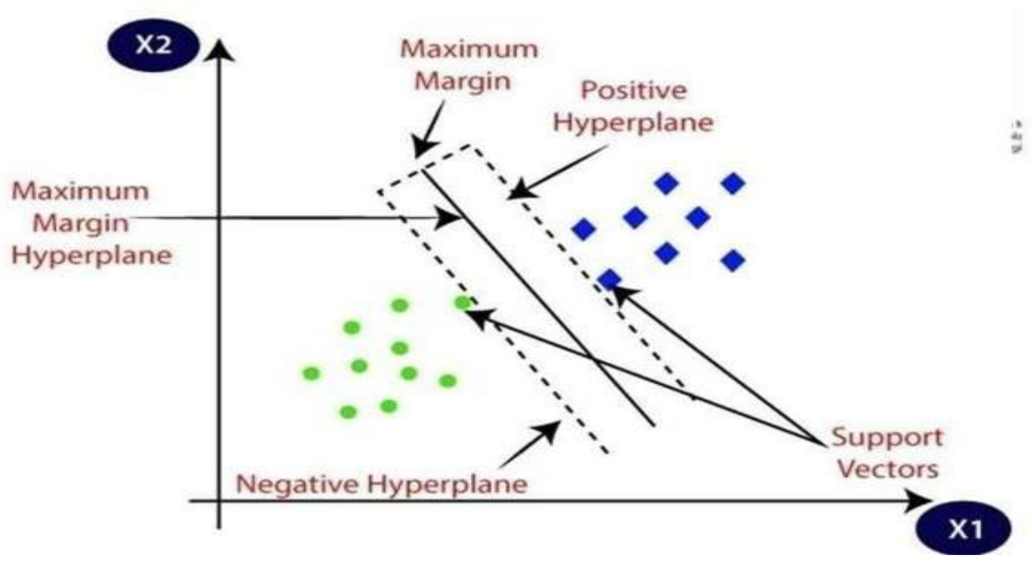
- In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.
- A decision tree simply asks a question, and based on the answer (Yes/No), it further splits the tree into subtrees.
- Below diagram explains the general structure of decision tree:



**Support Vector Machine (SVM)**
Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.
The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.
SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

**SVM chooses the extreme points/vectors that help in creating the hyperplane.** These extreme cases are called as **support vectors**, and hence the algorithm is termed as **Support Vector Machine**.

Consider the below diagram in which there are two different categories that are classified using a **decision boundary** or **hyperplane**:



**Types of SVM**
SVM can be of two types:

- **Linear SVM**: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is called as Linear SVM classifier.
- **Non-linear SVM**: Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

**Naïve Bayes**

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
- It is mainly used in text classification that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.
- Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

**Why it is called Naïve Bayes?**
The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, which can be described as:

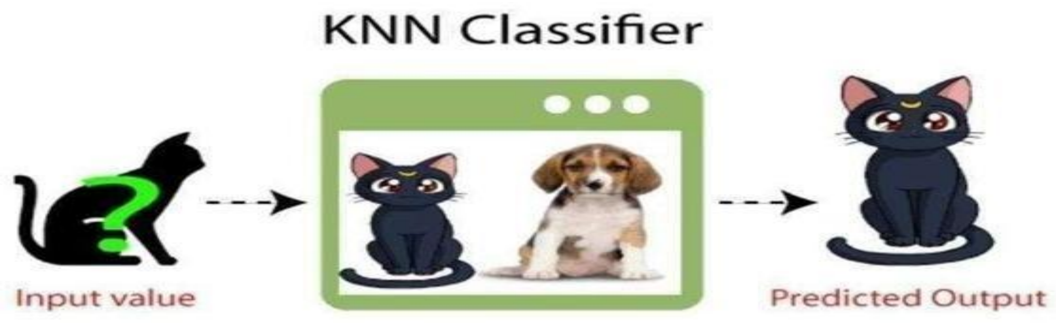
- **Naïve**: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the basis of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- **Bayes**: It is called Bayes because it depends on the principle of Bayes' Theorem.

**Bayes' Theorem Formula**
P(A|B) = [P(B|A) * P(A)] / P(B)
Where,
P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.
P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.
P(A) is Prior Probability: Probability of hypothesis before observing the evidence.
P(B) is Marginal Probability: Probability of Evidence.

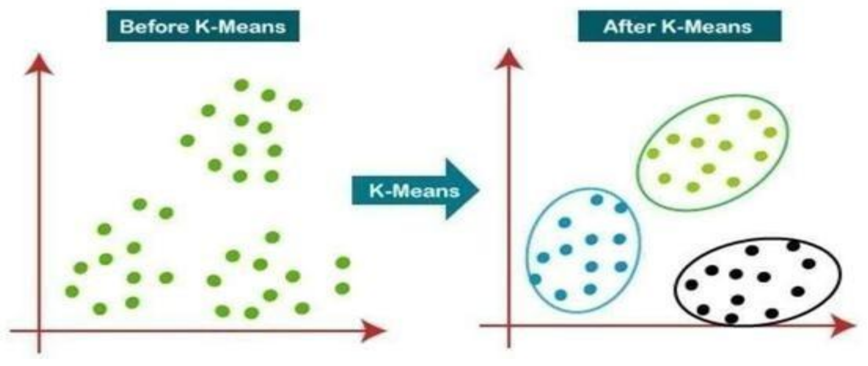
$$\mathrm{P(A|B)}= \frac{P(B|A)P(A)}{P(B)}$$

**KNN Algorithm (K-Nearest Neighbour)**

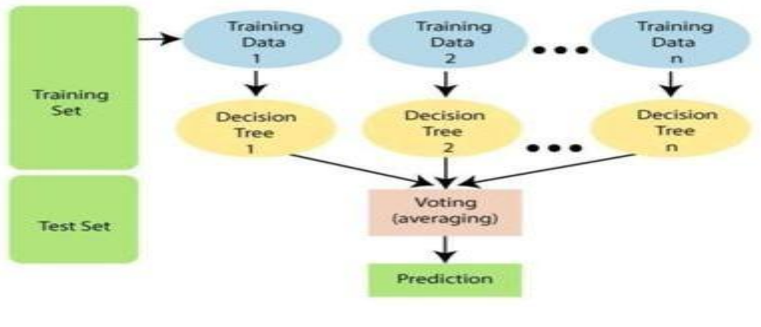
- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well-suited category by using K-NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.
- It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.



**K-means Algorithm**
K-means is a centroid-based algorithm, or a distance-based algorithm, where we calculate the distances to assign a point to a cluster. In K-Means, each cluster is associated with a centroid.



**Random Forest**
Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.



## ALGORITHMS

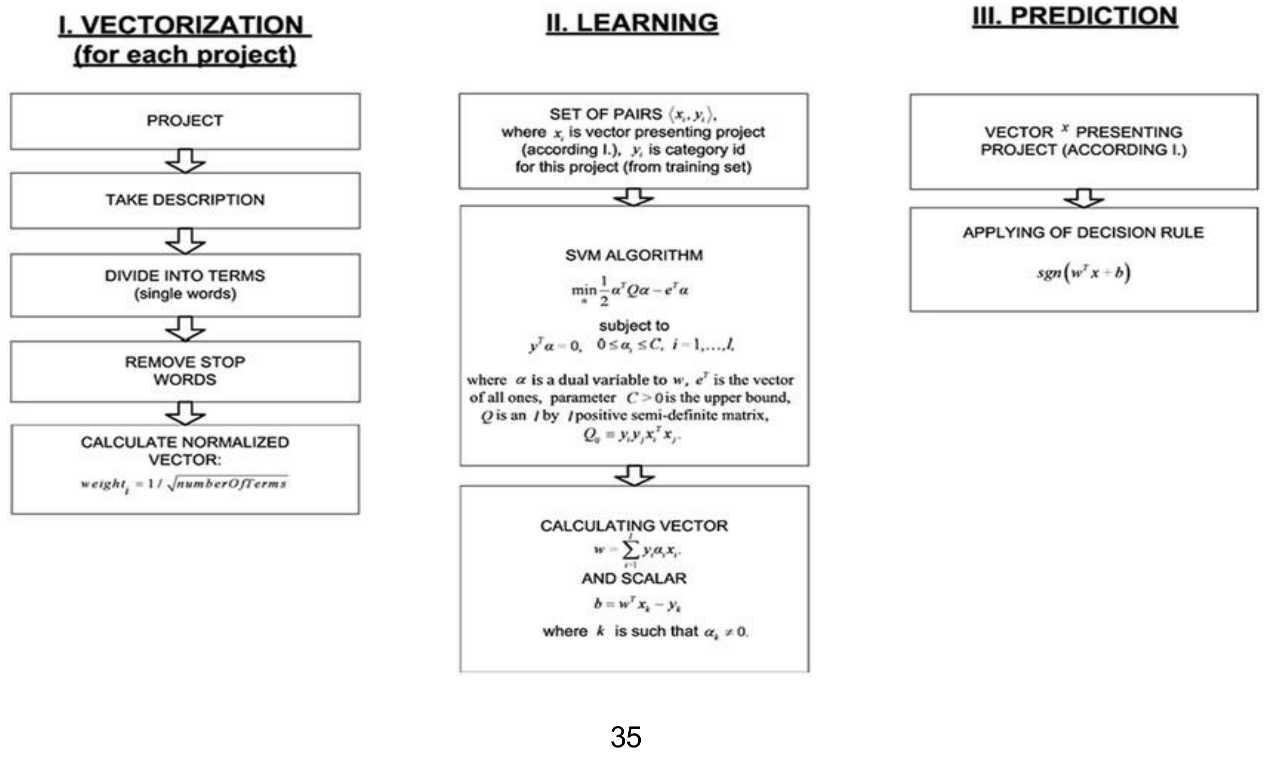
**Linear Regression Algorithm**
Steps to implement Linear Regression model:

1. Initialize the parameters.
2. Predict the value of a dependent variable by given an independent variable.
3. Calculate the error in prediction for all data points.
4. Calculate partial derivative w.r.t $a_0$ and $a_1$.
5. Calculate the cost for each number and add them.

**Logistic Regression Algorithm**

1. Data Pre-processing step
2. Fitting Logistic Regression to the Training set
3. Predicting the test result
4. Test accuracy of the result (Creation of Confusion matrix)
5. Visualizing the test set result.

**Decision Tree Algorithm**
Step-1: Begin the tree with the root node, says S, which contains the complete dataset.
Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).
Step-3: Divide the S into subsets that contain possible values for the best attributes.
Step-4: Generate the decision tree node, which contains the best attribute.
Step-5: Recursively make new decision trees using the subsets of the dataset created in step-3. Continue this process until a stage is reached where you cannot further classify the nodes and call the final node a leaf node.


I. VECTORIZATION (for each project)
PROJECT
TAKE DESCRIPTION
DIVIDE INTO TERMS (single words)
REMOVE STOP WORDS
CALCULATE NORMALIZED VECTOR: $weight_i = 1/\sqrt{numberOfTerms}$
II. LEARNING
SET OF PAIRS $\langle x_i, y_i \rangle$, where $x_i$ is vector presenting project (according I.), $y_i$ is category id for this project (from training set)
SVM ALGORITHM
$\min_{\alpha} \frac{1}{2}\alpha^T Q\alpha - e^T\alpha$
subject to
$y^T\alpha = 0, \quad 0 \le \alpha_i \le C, \ i = 1,\ldots,l,$
where $\alpha$ is a dual variable to $w$, $e^T$ is the vector of all ones, parameter $C > 0$ is the upper bound, $Q$ is an $l$ by $l$ positive semi-definite matrix, $Q_{ij} = y_i y_j x_i^T x_j$.
CALCULATING VECTOR $w = \sum_{i=1}^{l} y_i \alpha_i x_i$.
AND SCALAR $b = w^T x_k - y_k$
where $k$ is such that $\alpha_k \ne 0$.
III. PREDICTION
VECTOR $x$ PRESENTING PROJECT (ACCORDING I.)
APPLYING OF DECISION RULE $sgn(w^T x + b)$

**Lasso Regression Algorithm**
Lasso (Least Absolute Shrinkage and Selection Operator) Regression is a type of linear regression that uses shrinkage. Shrinkage is where data values are shrunk towards a central point, like the mean. The Lasso technique uses L1 regularization, which adds a penalty equal to the absolute value of the magnitude of coefficients.

Steps to implement Lasso Regression model:

1. Data Pre-processing step
2. Apply L1 Regularization (adds penalty equivalent to the absolute value of coefficients)
3. Minimize the cost function including the penalty term
4. Automatically shrink some coefficients to zero (thus performing feature selection)
5. Model prediction and evaluation using test data

Lasso regression is useful when we have a large number of features, as it helps in reducing overfitting by eliminating less important ones.

**Ridge Regression Algorithm**
Ridge Regression is a technique used when data suffers from multicollinearity (independent variables are highly correlated). It is a regularization method that introduces a small amount of bias to reduce variance in the model and improve prediction accuracy. It uses L2 regularization.

Steps to implement Ridge Regression model:

1. Data Pre-processing step
2. Apply L2 Regularization (adds penalty equivalent to the square of the magnitude of coefficients)
3. Modify the cost function to include the regularization term
4. Fit the model on training data
5. Use the trained model to predict and evaluate results

Ridge regression helps when the linear regression model is too complex and overfits the data by constraining the size of the coefficients.

**Naïve Bayes**

1. Data Pre-processing step
2. Fitting Naive Bayes to the Training set
3. Predicting the test result
4. Test accuracy of the result (Creation of Confusion matrix)
5. Visualizing the test set result.

**KNN Algorithm**
Step-1: Select the number K of the neighbors
Step-2: Calculate the Euclidean distance of K number of neighbors
Step-3: Take the K nearest neighbors as per the calculated Euclidean distance

Step-4: Among these K neighbors, count the number of the data points in each category
Step-5: Assign the new data points to that category for which the number of the neighbor is maximum
Step-6: Our model is ready.

**K-means Clustering**
Step-1: Select the number K to decide the number of clusters
Step-2: Select random K points or centroids (It can be other from the input dataset)
Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters
Step-4: Calculate the variance and place a new centroid of each cluster
Step-5: Repeat the third step, which means reassign each data point to the new closest centroid of each cluster
Step-6: If any reassignment occurs, then go to step-4 else go to FINISH
Step-7: The model is ready

**Random Forest**
Step-1: Select random K data points from the training set
Step-2: Build the decision trees associated with the selected data points (Subsets)
Step-3: Choose the number N for decision trees that you want to build
Step-4: Repeat Step 1 & 2
Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes

**XGBoost**
Step-1 Start with an initial prediction (often the mean for regression tasks).
Step-2: Compute the residuals (errors) between the predicted values and the actual target values.
Step-3 Fit a new decision tree to these residuals to learn the patterns of the errors.
Step-4: Add the predictions from the new tree to the existing model, scaled by a learning rate to control the contribution.
Step-5: Continue adding trees iteratively, with each tree aiming to reduce the remaining error.
Step-6 Combine the predictions from all trees for the final result.

- In classification, probabilities are converted into class labels.
- In regression, the average of predictions is used as the final output.

# IMPLEMENTATION OF PROJECT

## Technologies Used:

| **Component** | **Technology** |
|---|---|
| Web App Framework | Streamlit |
| Machine Learning Model | XG Boost |
| Data Preprocessing | Pandas, NumPy |
| Feature Engineering | Label Encoding (center_id, meal_id) |
| Visualizations | Matplotlib, Seaborn |
| Model Storage | Pickle |
| User Input Handling | Streamlit Widgets (st.number_input, st.button) |

The Food Demand Forecasting dataset contains historical records of meal orders from multiple fulfillment centers. Each entry includes:

- Center ID – Unique identifier for the distribution center.
- Meal ID – Unique identifier for each meal item.
- Checkout Price – The final selling price paid by the customer.
- Base Price – The original price of the meal before discounts.
- Emailer for Promotion – Indicator of whether the meal was promoted via email marketing.
- Homepage Featured – Indicator of whether the meal was featured on the platform's homepage.
- Week – The week number of the order, representing the time period.
- Number of Orders – The target variable indicating the quantity of meals ordered.

This dataset enables the analysis of historical sales trends, pricing effects, and promotional impacts, providing a foundation for building machine learning models that can accurately forecast future food demand.

## ML Model Training:

In [ ]:

```
# Importing libraries
import numpy as np
import pandas as pd
import pickle
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import mean_squared_error, r2_score
from math import sqrt
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:

```
# Loading all the datasets
traindf = pd.read_csv('/content/train.csv')
centerdf = pd.read_csv('/content/fulfilment_center_info.csv')
mealdf = pd.read_csv('/content/meal_info.csv')
testdf = pd.read_csv('/content/food_Demand_test.csv')
```

In [ ]:

```
traindf.head() # Displaying the first few rows of the tain dataset
```

Out[ ]:

| | id | week | center_id | meal_id | checkout_price | base_price | emailer_for_promotion | homepage_featured | num_orders |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1379560 | 1 | 55 | 1885 | 136.83 | 152.29 | 0 | 0 | 177 |
| 1 | 1466964 | 1 | 55 | 1993 | 136.83 | 135.83 | 0 | 0 | 270 |
| 2 | 1346989 | 1 | 55 | 2539 | 134.86 | 135.86 | 0 | 0 | 189 |
| 3 | 1338232 | 1 | 55 | 2139 | 339.50 | 437.53 | 0 | 0 | 54 |
| 4 | 1448490 | 1 | 55 | 2631 | 243.50 | 242.50 | 0 | 0 | 40 |

In [ ]:

```
centerdf.head() # Displaying the first few rows of the center dataset
```

Out[ ]:

| | center_id | city_code | region_code | center_type | op_area |
|---|---|---|---|---|---|
| 0 | 11 | 679 | 56 | TYPE_A | 3.7 |
| 1 | 13 | 590 | 56 | TYPE_B | 6.7 |
| 2 | 124 | 590 | 56 | TYPE_C | 4.0 |
| 3 | 66 | 648 | 34 | TYPE_A | 4.1 |
| 4 | 94 | 632 | 34 | TYPE_C | 3.6 |

In [ ]:

```
mealdf.head() # Displaying the first few rows of the meal dataset
```

Out[ ]:

| | meal_id | category | cuisine |
|---|---|---|---|
| 0 | 1885 | Beverages | Thai |
| 1 | 1993 | Beverages | Thai |
| 2 | 2539 | Beverages | Thai |
| 3 | 1248 | Beverages | Indian |
| 4 | 2631 | Beverages | Indian |

In [ ]:

```
print("Statistical Summary of the Train dataset:")
traindf.describe()
```

```
Statistical Summary of the Train dataset:
```

Out[ ]:

| | id | week | center_id | meal_id | checkout_price | base_price | emailer_for_promotion | homepage_featured | num_orders |
|---|---|---|---|---|---|---|---|---|---|
| count | 4.565480e+05 | 456548.000000 | 456548.000000 | 456548.000000 | 456548.000000 | 456548.000000 | 456548.000000 | 456548.00000 | 456548.000000 |
| mean | 1.250096e+06 | 74.768771 | 82.105796 | 2024.337458 | 332.238933 | 354.156627 | 0.081152 | 0.10920 | 261.872760 |
| std | 1.443548e+05 | 41.524956 | 45.975046 | 547.420920 | 152.939723 | 160.715914 | 0.273069 | 0.31189 | 395.922798 |
| min | 1.000000e+06 | 1.000000 | 10.000000 | 1062.000000 | 2.970000 | 55.350000 | 0.000000 | 0.00000 | 13.000000 |
| 25% | 1.124999e+06 | 39.000000 | 43.000000 | 1558.000000 | 228.950000 | 243.500000 | 0.000000 | 0.00000 | 54.000000 |
| 50% | 1.250184e+06 | 76.000000 | 76.000000 | 1993.000000 | 296.820000 | 310.460000 | 0.000000 | 0.00000 | 136.000000 |
| 75% | 1.375140e+06 | 111.000000 | 110.000000 | 2539.000000 | 445.230000 | 458.870000 | 0.000000 | 0.00000 | 324.000000 |
| max | 1.499999e+06 | 145.000000 | 186.000000 | 2956.000000 | 866.270000 | 866.270000 | 1.000000 | 1.00000 | 24299.000000 |

In [ ]:
```
print("Information about the Train dataset:")
traindf.info()
```

```
Information about the Train dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 456548 entries, 0 to 456547
Data columns (total 9 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   id                     456548 non-null  int64  
 1   week                   456548 non-null  int64  
 2   center_id              456548 non-null  int64  
 3   meal_id                456548 non-null  int64  
 4   checkout_price         456548 non-null  float64
 5   base_price             456548 non-null  float64
 6   emailer_for_promotion  456548 non-null  int64  
 7   homepage_featured      456548 non-null  int64  
 8   num_orders             456548 non-null  int64  
dtypes: float64(2), int64(7)
memory usage: 31.3 MB
```

In [ ]:

```
# Feature Selection and Data Preprocessing
# Checking for missing values in train dataset
print("Missing values in train dataset:")
traindf.isnull().sum()
```

```
Missing values in train dataset:
```

Out[ ]:

```
                        0
id                      0
week                    0
center_id               0
meal_id                 0
checkout_price          0
base_price              0
emailer_for_promotion   0
homepage_featured       0
num_orders              0

dtype: int64
```

In [ ]:

```
# Checking for missing values in test dataset
print("Missing values in test dataset:")
testdf.isnull().sum()
```

```
Missing values in test dataset:
```

Out[ ]:

```
                        0
id                      0
week                    0
center_id               0
meal_id                 0
checkout_price          0
base_price              0
emailer_for_promotion   0
homepage_featured       0

dtype: int64
```

In [ ]:
```
# Label encoding 'center_id' and 'meal_id'
# Converting these to category type and then use their codes
traindf['center_id'] = traindf['center_id'].astype('category').cat.codes
traindf['meal_id'] = traindf['meal_id'].astype('category').cat.codes

testdf['center_id'] = testdf['center_id'].astype('category').cat.codes
testdf['meal_id'] = testdf['meal_id'].astype('category').cat.codes
```

In [ ]:

```
traindf.head()
```

Out[ ]:

| | id | week | center_id | meal_id | checkout_price | base_price | emailer_for_promotion | homepage_featured | num_orders |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1379560 | 1 | 23 | 22 | 136.83 | 152.29 | 0 | 0 | 177 |
| 1 | 1466964 | 1 | 23 | 26 | 136.83 | 135.83 | 0 | 0 | 270 |
| 2 | 1346989 | 1 | 23 | 38 | 134.86 | 135.86 | 0 | 0 | 189 |
| 3 | 1338232 | 1 | 23 | 29 | 339.50 | 437.53 | 0 | 0 | 54 |
| 4 | 1448490 | 1 | 23 | 42 | 243.50 | 242.50 | 0 | 0 | 40 |

In [ ]:
```
testdf.head()
```

Out[ ]:

| | id | week | center_id | meal_id | checkout_price | base_price | emailer_for_promotion | homepage_featured |
|---|---|---|---|---|---|---|---|---|
| 0 | 1028232 | 146 | 23 | 22 | 158.11 | 159.11 | 0 | 0 |
| 1 | 1127204 | 146 | 23 | 26 | 160.11 | 159.11 | 0 | 0 |
| 2 | 1212707 | 146 | 23 | 38 | 157.14 | 159.14 | 0 | 0 |
| 3 | 1082698 | 146 | 23 | 42 | 162.02 | 162.02 | 0 | 0 |
| 4 | 1400926 | 146 | 23 | 7 | 163.93 | 163.93 | 0 | 0 |

In [ ]:

```
# Splitting the train dataset into features and target variable
x = traindf.drop('num_orders', axis=1)  # Features
y = traindf['num_orders']  # Target variable
```

In [ ]:

```
# Displaying the final shape of the dataset prepared for training
print("Shape of the Features (X):", x.shape)
print("\nShape of the Target (y):", y.shape)
```

```
Shape of the Features (X): (456548, 8)

Shape of the Target (y): (456548,)
```

In [ ]:

```
# Splitting the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=
42)
```

In [ ]:

```
# Training different models and evaluating their performance
# 1. Linear Regression
lr_model = LinearRegression()
lr_model.fit(x_train, y_train)
lr_pred = lr_model.predict(x_test)
```

In [ ]:

```
# 2. Decision Tree Regressor
dt_model = DecisionTreeRegressor()
dt_model.fit(x_train, y_train)
dt_pred = dt_model.predict(x_test)
```

In [ ]:

```
# 3. Random Forest Regressor
rf_model = RandomForestRegressor()
rf_model.fit(x_train, y_train)
rf_pred = rf_model.predict(x_test)
```

In [ ]:

```
# 4. XGBoost Regressor
xg_model = XGBRegressor()
xg_model.fit(x_train, y_train)
xg_pred = xg_model.predict(x_test)
```

In [ ]:

```
# Function to calculate RMSE and R2 score
def evaluate_model(model_name, true_values, predicted_values):
    rmse = sqrt(mean_squared_error(true_values, predicted_values))
    r2 = r2_score(true_values, predicted_values)
    print(f"{model_name} - RMSE: {rmse}, R2 Score: {r2}")
```

In [ ]:

```
# Evaluating the models
evaluate_model("Linear Regression", y_test, lr_pred)
evaluate_model("Decision Tree Regressor", y_test, dt_pred)
evaluate_model("Random Forest Regressor", y_test, rf_pred)
evaluate_model("XGBoost Regressor", y_test, xg_pred)
```

```
Linear Regression - RMSE: 349.92422829186705, R2 Score: 0.19724527662483704
Decision Tree Regressor - RMSE: 236.96595434761792, R2 Score: 0.6318653041278197
Random Forest Regressor - RMSE: 179.70409643284836, R2 Score: 0.7882854055620016
XGBoost Regressor - RMSE: 173.5590002347905, R2 Score: 0.8025172352790833
```

In [ ]:

```
# Expanding the parameter grid and performing hyper parameter tuning
param_grid = {
    'n_estimators': [100, 200],  # Reduced options
    'learning_rate': [0.05, 0.1], # Reduced options
    'max_depth': [5, 8],          # Reduced options
    'subsample': [0.8, 1.0],      # Reduced options
    'colsample_bytree': [0.8, 1.0], # Reduced options
    'gamma': [0],                 # Fixed to a common value
    'reg_lambda': [1],            # Fixed to a common value
    'reg_alpha': [0]              # Fixed to a common value
}

# XGBoost base model
xgb = XGBRegressor(random_state=42, n_jobs=-1)

# Randomized Search CV
random_search = RandomizedSearchCV(
    estimator=xgb,
    param_distributions=param_grid,
    n_iter=5,  # Significantly reduced for speed
    scoring='neg_root_mean_squared_error',
    cv=2,      # Reduced cross-validation folds
    verbose=1,
    random_state=42,
    n_jobs=-1
)

# For early stopping, we need a validation set.
# Creating a small validation set from the training data for this purpose.
# Note: This is a simplified approach. For production, consider a dedicated validation
split.
x_train_split, x_val_split, y_train_split, y_val_split = train_test_split(
    x_train, y_train, test_size=0.1, random_state=42
)

random_search.fit(x_train, y_train) # Using the original x_train, y_train for the sear
ch

# Best model and prediction
best_model = random_search.best_estimator_
best_model_pred = best_model.predict(x_test)
```

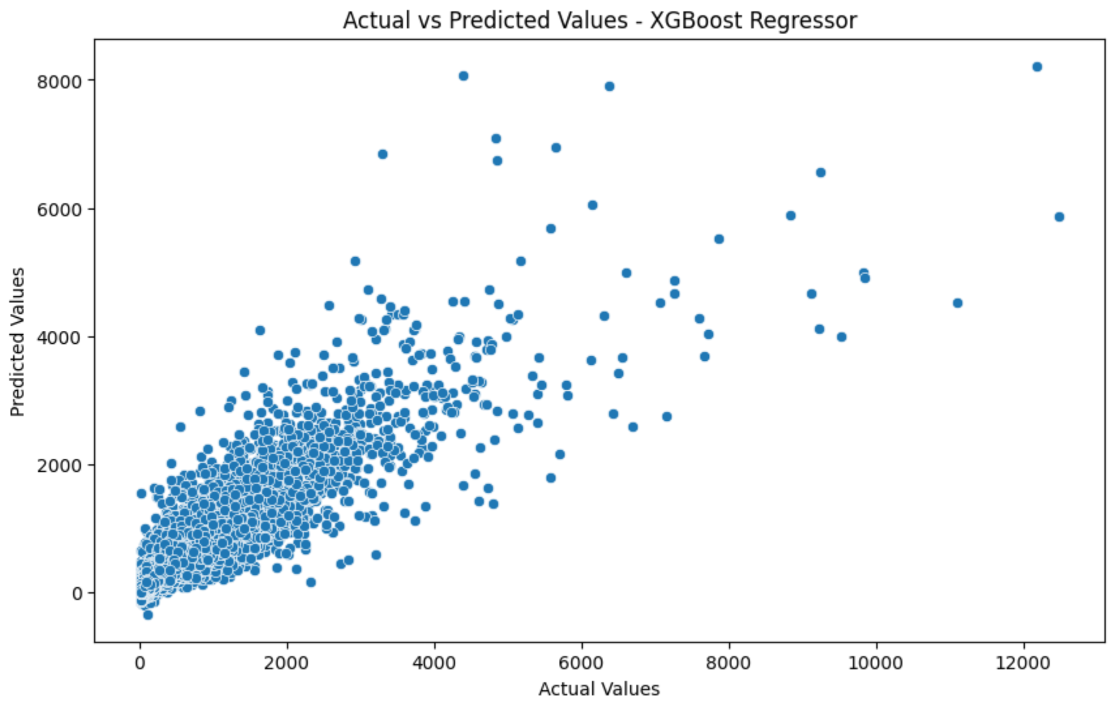
## Model Evaluation (MSE, R²):

```
# Evaluation
print("Best Parameters:", random_search.best_params_)
print("RMSE:", sqrt(mean_squared_error(y_test, best_model_pred)))
print("R² Score:", r2_score(y_test, best_model_pred))
```

```
Fitting 2 folds for each of 5 candidates, totalling 10 fits
Best Parameters: {'subsample': 1.0, 'reg_lambda': 1, 'reg_alpha': 0, 'n_estimators': 2
00, 'max_depth': 8, 'learning_rate': 0.1, 'gamma': 0, 'colsample_bytree': 0.8}
RMSE: 163.54523217981622
R² Score: 0.8246480226516724
```

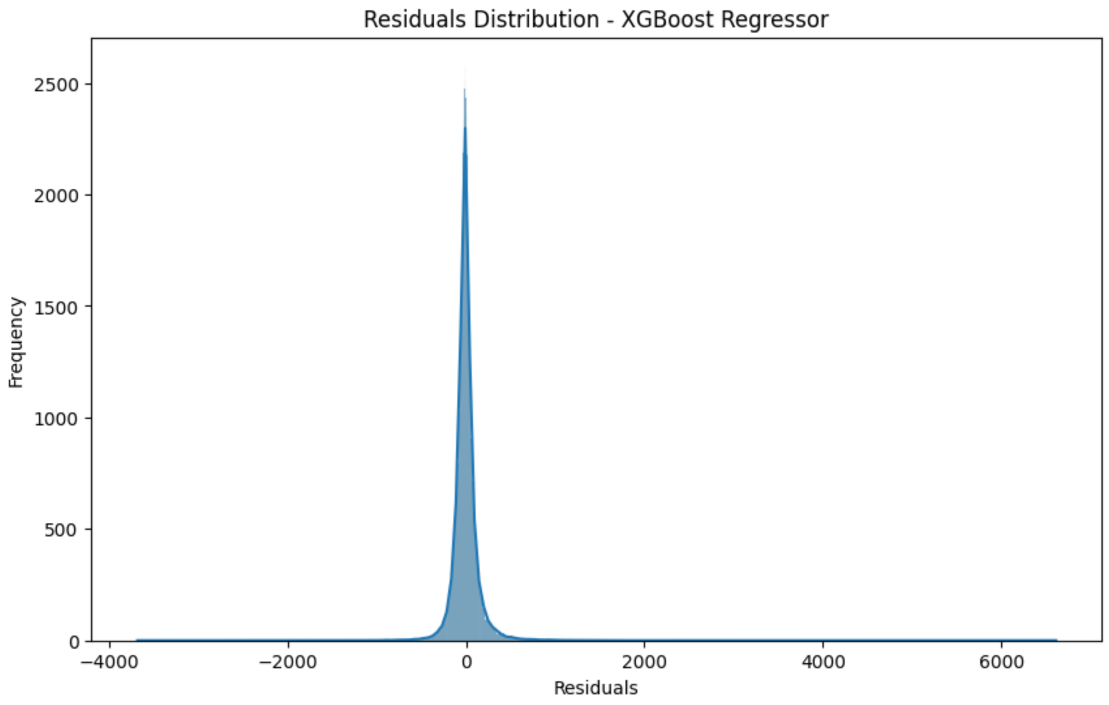
In [ ]:

```
# Plotting the Actual vs Predicted values
plt.figure(figsize=(10, 6))
sns.scatterplot(x=y_test, y=best_model_pred)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted Values - XGBoost Regressor')
plt.show()
```

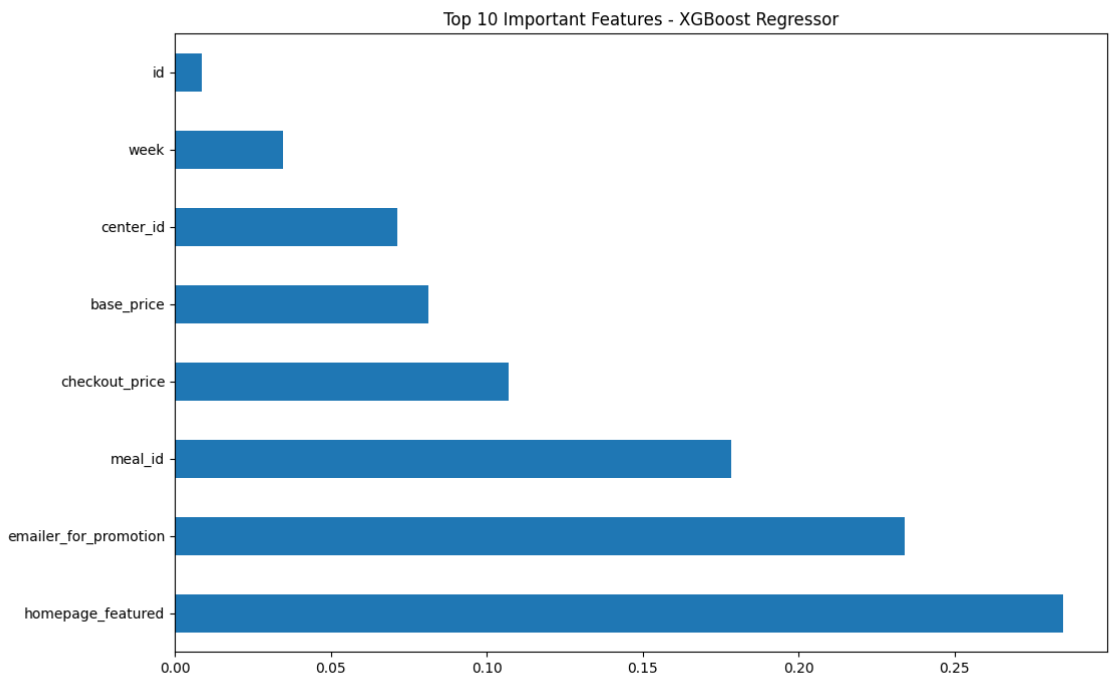


In [ ]:

```
# Residuals plot
residuals = y_test - best_model_pred
plt.figure(figsize=(10, 6))
sns.histplot(residuals, kde=True)
plt.title('Residuals Distribution - XGBoost Regressor')
plt.xlabel('Residuals')
plt.ylabel('Frequency')
plt.show()
```



In [ ]:
```
# Results Visualization
# Feature Importance Visualization
feature_importances = pd.Series(best_model.feature_importances_, index=x.columns)
plt.figure(figsize=(12, 8))
feature_importances.nlargest(10).plot(kind='barh')
plt.title('Top 10 Important Features - XGBoost Regressor')
plt.show()
```



In [ ]:

```
# Predictions Visualization
# Displaying a sample of actual vs predicted values
sample_results = pd.DataFrame({'Actual': y_test, 'Predicted': best_model_pred})
sample_results = sample_results.reset_index(drop=True)
print("Sample Actual vs Predicted Values:")
sample_results.head(10)
```

```
Sample Actual vs Predicted Values:
```

Out[ ]:

| | Actual | Predicted |
|---|---|---|
| 0 | 28 | 93.052795 |
| 1 | 176 | 318.836182 |
| 2 | 391 | 197.302612 |
| 3 | 14 | 61.019741 |
| 4 | 405 | 584.964722 |
| 5 | 15 | 89.372154 |
| 6 | 13 | 67.793465 |
| 7 | 96 | 90.886642 |
| 8 | 96 | 100.620613 |
| 9 | 1391 | 1217.708496 |

## Pickle File Saving:

In [ ]:
```
with open('food_demand_forecast.pkl', 'wb') as file:
    pickle.dump(best_model, file)
```

In [ ]:

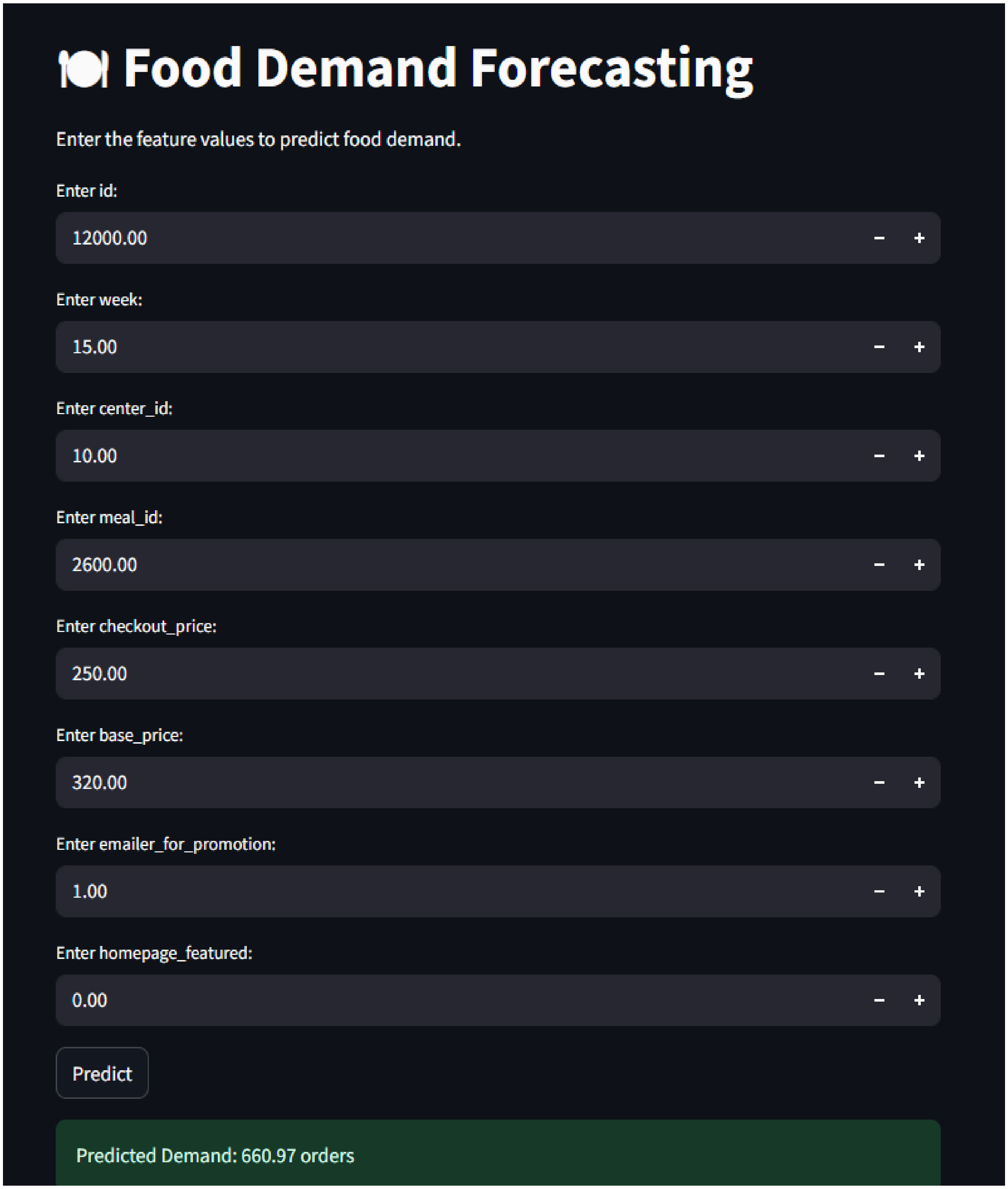
```
from google.colab import files
files.download('food_demand_forecast.pkl')
```

## STREAMLIT WEB APP DEVELOPMENT

```
import streamlit as st
import pandas as pd
import pickle

# Load Model
MODEL_FILE = "food_demand_forecast.pkl"

@st.cache_resource
def load_model():
   with open(MODEL_FILE, "rb") as f:
      return pickle.load(f)

model = load_model()

st.title("🍽 Food Demand Forecasting")
st.write("Enter the feature values to predict food demand.")

# All feature names used in actual model in correct order
feature_names = [
   "id", "week", "center_id", "meal_id",
   "checkout_price", "base_price",
   "emailer_for_promotion", "homepage_featured"
]

# Collect input values
input_data = {}
for col in feature_names:
   input_data[col] = st.number_input(f"Enter {col}:", value=0.0)

# Predict button
if st.button("Predict"):
   input_df = pd.DataFrame([input_data])
   prediction = model.predict(input_df)[0]
   st.success(f"Predicted Demand: {prediction:.2f} orders")
```

### Sample Input Data:

| Feature | Sample Value | Description |
|---|---|---|
| **id** | 12000 | Unique row identifier for each record |
| **week** | 15 | Week number in the dataset timeline |
| **center_id** | 10 | Fulfilment center unique ID |
| **meal_id** | 2600 | Unique meal identifier |
| **checkout_price** | 250.00 | Price charged after discounts |
| **base_price** | 320.00 | Original meal price before discounts |
| **emailer_for_promotion** | 1 | 1 = Promotional email sent, 0 = Not sent |
| **homepage_featured** | 0 | 1 = Featured on homepage, 0 = Not featured |


# 🍽 Food Demand Forecasting

Enter the feature values to predict food demand.

Enter id:

12000.00 − +

Enter week:

15.00 − +

Enter center_id:

10.00 − +

Enter meal_id:

2600.00 − +

Enter checkout_price:

250.00 − +

Enter base_price:

320.00 − +

Enter emailer_for_promotion:

1.00 − +

Enter homepage_featured:

0.00 − +

Predict

Predicted Demand: 660.97 orders

## FUTURE SCOPE

The Food demand forecasting system holds significant potential for real-world applications. In the future:

- **Integration with Real-Time Data**

  Incorporating live order data, weather updates, and ongoing promotional campaigns to improve forecast accuracy dynamically.

- **Seasonality and Event-Based Predictions**

  Enhancing the model to account for seasonal trends, festivals, public holidays, and local events that influence food demand spikes.

- **Advanced Machine Learning Models**

  Exploring deep learning approaches (e.g., LSTM, GRU) for time-series forecasting to capture long-term dependencies in demand patterns.

- **Multi-Location Forecasting**

  Expanding the system to handle simultaneous predictions for multiple fulfillment centers and geographic regions.

- **Menu Optimization**

  Using demand insights to optimize menus by highlighting high-demand meals and adjusting offerings based on predicted popularity.

- **Automated Inventory Management**

  Integrating the forecasting model with procurement systems to automate raw material ordering and reduce wastage.

## CONCLUSION

The Food Demand Forecasting system was developed using historical meal order data combined with fulfillment center and meal information. After preprocessing steps such as encoding categorical features and preparing the dataset, multiple regression models were trained, including **Linear Regression, Decision Tree, Random Forest,** and **XGBoost.** Among these, the **XGBoost Regressor** demonstrated the most reliable predictive performance.

The analysis showed that factors like checkout price, base price, and promotional activities had a significant impact on demand patterns. The developed model can be applied as a practical decision-support tool for optimizing inventory, minimizing wastage, and enhancing operational efficiency in the food supply chain. Future improvements may include incorporating seasonal trends, external market conditions, and real-time order data for more dynamic and adaptive forecasting.