Programs Offered

Post Graduate Programmes (PG)

- Master of Business Administration
- Master of Computer Applications
- Master of Commerce (Financial Management / Financial Technology)
- Master of Arts (Journalism and Mass Communication)
- Master of Arts (Economics)
- Master of Arts (Public Policy and Governance)
- Master of Social Work
- Master of Arts (English)
- Master of Science (Information Technology) (ODL)
- Master of Science (Environmental Science) (ODL)

Diploma Programmes

- Post Graduate Diploma (Management)
- Post Graduate Diploma (Logistics)
- Post Graduate Diploma (Machine Learning and Artificial Intelligence)
- Post Graduate Diploma (Data Science)

Undergraduate Programmes (UG)

- Bachelor of Business Administration
- Bachelor of Computer Applications
- Bachelor of Commerce
- Bachelor of Arts (Journalism and Mass Communication)
- Bachelor of Arts (General / Political Science / Economics / English / Sociology)
- Bachelor of Social Work
- Bachelor of Science (Information Technology) (ODL)

AMITY UNIVERSITY ONLINE
Amity Helpline: (Toll free) 18001023434
For Student Support: +91 - 8826334455
Support Email id: studentsupport@amityonline.com | https://amityonline.com

Product code
81030112001133

AMITY

Introduction to Machine Learning

Introduction to Machine Learning (MCA)

AMITY UNIVERSITY ONLINE

Introduction to Machine Learning

AMITY UNIVERSITY ONLINE

SLM & Learning Resources Committee

Chairman	:	Prof. Abhinash Kumar
Members	:	Dr. Ranjit Varma
		Dr. Maitree
		Dr. Divya Bansal
		Dr. Arun Som
		Dr. Sunil Kumar
		Dr. Reema Sharma
		Dr. Winnie Sharma
Member Secretary	:	Ms. Rita Naskar

Published by Amity University Press for exclusive use of Amity Directorate of Distance and Online Education, Amity University, Noida-201313

Contents

Module - I: Introduction to Machine Learning and Artificial Intelligence

Learning Objectives

At the end of this module, you will be able to:

- Discuss machine learning and artificial intelligence
- Describe linear and logistic regression
- Recognise neural network and deep learning

Introduction

Since the invention of programmable computers, artificial intelligence (AI) has been a hot topic. The distinctions between man and machine were questioned by academics and philosophers. Could the human brain, with all of its complexities, be programmed into a computer? Will a computer be able to think at that point?

We have yet to find answers to these fascinating, mind-numbing problems, but we are getting closer to making computers smarter. Some may claim that even the most intelligent computers have less intelligence than a cockroach. Take a moment to consider that. Even the most advanced computers are unable to perform multiple tasks at the same time. Instead, they excel at the one task for which they were programmed.

Let's establish a few crucial words before we go any further. We choose one of the many definitions available on the internet.

- The first three are arranged in a hierarchical order, with AI being the largest and most general category. Deep Learning (DL) is a subset of Machine Learning (ML), which is a subset of ML.
- Artificial intelligence (AI) is a term used to describe a computer system that can execute tasks that would ordinarily require human intelligence, such as visual perception, speech recognition, decision-making and language translation.
- Arthur Samuel remarked on machine learning. "The ability to learn without being explicitly programmed" is defined as "machine learning."
- MIT News reports on deep learning: A neural net is a tightly connected network of thousands or even millions of simple processing nodes that is loosely modelled after the human brain. This is analogous to axon-dendrite synaptic interactions.
- Image recognition is the process of identifying the contents of an image using machine and deep learning algorithms.
- The scaffolding and blueprints of the algorithm model used to anticipate an outcome are referred to as architecture.

1.1.1 History of Artificial Intelligence

In 1955, computer scientist John McCarthy coined the term "artificial intelligence" in a proposal for a conference at Dartmouth College. He later established AI laboratories at MIT and Stanford, where he also taught mathematics. The Dartmouth Conference held in 1956 was a pivotal event where the concept of "thinking machines" and various AI topics, including neural networks and natural language processing, were formally discussed, shaping the foundation of AI as a field.

Notes

The roots of AI can be traced back to ancient times, where Aristotle's syllogistic logic and Al-Jazari's programmable humanoid robot in the 13th century demonstrated early attempts at artificial entities. The invention of the printing press, Pascal's mechanical calculating machine and mathematical advances in the 20th century, such as Bertrand Russell and Alfred North Whitehead's Principia Mathematica, also contributed to the development of AI concepts.

Over time, the rise of technology companies, advancements in computer technology and various programs have led to significant revolutionary strides in AI's understanding and application. Today, artificial intelligence plays a prominent role in various industries, showcasing its potential and transforming the way we interact with technology and machines. Before John McCarthy coined the term "artificial intelligence," Alan Turing made significant contributions to the field. During World War II, Turing played a crucial role in deciphering the German Enigma machine, aiding the Allied efforts. He is best known for conceptualizing the "Turing machine," a theoretical device designed to simulate human thought using precise mathematical principles of computability. Turing's vision extended to creating a chess-playing automaton that mimics human-like mental processes.

History of AI with Chronological Order:

- **May 1. 1960:** In antiquity, Alexander Heron created automatons using mechanical mechanisms that used water and steam power.
- **1623:** Wilhelm Schickard designed a mechanic and a four-operation calculator.
- **1672:** Gottfried Leibniz has developed a binary counting system that forms the abstract basis of today's computers.
- **1822-1859:** Charles Babbage is credited with inventing the mechanical calculator. Because of his work with Babbage's punched cards on his computers, Ada Lovelace is considered as the first computer programmer. Algorithms are among Lovelace's contributions.
- **1923:** In the stage piece Rossum's Universal Robots (RUR - Rossum's Universal Robots), Karel Capek originally proposed the robot concept.
- **1931:** Kurt Gödel introduced the theory of deficiency, which is called by his own name.
- **1936:** Konrad Zuse developed a programmable computer named Z1 named 64K memory.
- **1946:** ENIAC (Electronic Numerical Integrator and Computer), the first computer in a room size of 30 tons, started to work.
- **1948:** John von Neumann introduced the idea of self-replicating program.
- **1950:** Alan Turing, founder of computer science, introduced the concept of the Turing Test.
- **1951:** The first artificial intelligence programs for the Mark 1 device were written.
- **1956:** The logic theorist (Logic Theory-LT) program for solving mathematical problems is introduced by Neweell, Shaw and Simon. The system is regarded as the first artificial intelligence system.
- The end of the **1950s -** the beginning of the 1960s: A schematic network for machine translation was developed by Margaret Masterman et al.
- **1958:** John McCarty of MIT created the LISP (list Processing language) language.

Notes

- **1960:** JCR Licklider described the human-machine relationship in his work.
- **1962:** Unimation was established as the first company to produce robots for the industrial field.
- **1965:** An artificial intelligence program ELIZA is written.
- **1966:** The first animated robot "Shakey" was produced at Stanford University.
- **1973:** DARPA begins development for protocols called TCP / IP.
- **1974:** The Internet has begun to be used for the first time.
- **1978:** Herbert Simon earned a Nobel Prize for his limited Rationality Theory, which is an important work on Artificial Intelligence.
- **1981:** IBM produced the first personal computer.
- **1993:** Production of Cog, a human-looking robot at MIT, began.
- **1997:** Deep Blue named supercomputer defeated world famous chess player Kasparov.
- **1998:** Furby, the first artificial intelligence player, was driven to the market.
- **2000:** Kismet named robot which can use gesture and mimic movements in communication is introduced.
- **2005:** Asimo, the closest robot to artificial intelligence and human ability and skill, is introduced.
- **2010:** Asimo is made to act using mind power

1.1.2 Definition of AI

The term "AI" encompasses various definitions depending on the context or profession. Here are some examples:

- AI pertains to the capacity of a digital computer or computer-controlled robot to carry out tasks that are commonly linked with intelligent entities.
- AI involves the exploration and creation of intelligent agents, which are systems capable of perceiving their surroundings and making decisions to enhance their likelihood of achieving success.
- AI encompasses the study and advancement of computer systems that can execute tasks typically necessitating human intelligence, including visual perception, speech recognition, decision-making and language translation.
- AI, also known as machine intelligence, refers to the intelligence exhibited by machines, contrasting with natural intelligence displayed by humans and other animals.

According to current U.S. legislation, AI refers to any artificial system that can learn from experience and improve its performance through exposure to data, or can carry out tasks in various and unpredictable situations without extensive human oversight. This can be in the form of computer programs, physical hardware, or other context-based artificial systems capable of executing tasks that require human-like perception, cognition, planning, learning, communication, or physical action. AI may involve various techniques like cognitive architectures, neural networks, or a combination of methods such as machine learning to simulate cognitive tasks. In essence, AI encompasses systems that use perception, planning, reasoning, learning, communication, decision-making and action to achieve their objectives, such as intelligent software agents or embodied robots.

Notes

1.1.3 Definition of Machine Learning

Machine Learning is a versatile and powerful technology that enables computers to learn from data and experiences. A wide range of machine-learning algorithms and methodologies have been developed to facilitate this learning process. It is built upon the foundations of computer science and statistics, making it a valuable tool for various applications. One well-known Machine Learning library is Scikit-Learn, which is built upon popular Python scientific libraries like NumPy, SciPy and Matplotlib. This integration makes it easy to incorporate additional models into Scikit-Learn. The library supports various tasks such as classification, regression, clustering, model selection, dimensionality reduction and data preprocessing. Some notable features of Scikit-Learn include parallelization integration, consistent APIs, comprehensive documentation, and its availability under a BSD license, with the option of a commercial license for full support.

Machine learning is all about making computers adjust or adapt their activities (whether these activities are making predictions or commanding an automaton) to become more accurate. The accuracy is determined by how closely the selected actions resemble the correct ones. Assume you are competing in Scrabble (or another game) against an automaton. Initially, you may win every time, but after a few bouts, it begins to win and you inevitably lose every time. Either you're getting worse at Scrabble, or the computer is learning how to beat you. After learning how to defeat you, it can apply the same strategies to other players without having to start from scratch with each new opponent; this is known as generalisation.

In the last decade or so, the inherent multidisciplinary of machine learning has been recognised. Combining concepts from neurology and biology, statistics, mathematics, and physics enables computers to learn. The container of water and electricity (along with a few trace elements) nestled between your ears is a beautiful example of the possibility of learning.

In Section 3.1, we will investigate briefly whether there are any principles from which machine learning algorithms can draw inspiration. Intriguingly, we discover that neural networks are one of these outcomes, despite the fact that they have undergone significant transformations to become statistical learners. Data mining, which involves extracting valuable insights from large datasets using computers and pocket protectors instead of conventional instruments such as pickaxes and hazardous helmets, also influences the transition in machine learning research. This change has prompted a refocus of machine learning research on computer science in order to develop efficient algorithms for data analysis.

We'll be interested in the computational complexity of machine learning methods because we're creating algorithms. It's especially essential because we may want to employ some of the methods on very large datasets, making algorithms with high degree polynomial complexity in the dataset size (or worse) an issue. The complexity of a problem is sometimes divided into two parts: the difficulty of training and the difficulty of using the learned method. Because training isn't done very regularly and isn't usually time-sensitive, it can take longer. However, we often need to make a choice regarding a test point quickly and when an algorithm is used, there may be a lot of test points, therefore this must be inexpensive in computing cost.

Arthur Samuel, a pioneering American expert in computer gaming and artificial intelligence, introduced the term "Machine Learning" in 1959 during his tenure at IBM. He described it as "the branch of research that empowers computers to learn without explicit programming." However, it is essential to note that machine learning does not possess

Notes

a universally agreed-upon definition and varies across different sources. Below are two additional definitions of the term.

1. Machine learning involves training computers to enhance performance by maximizing a performance criterion using past experiences or example data. It is achieved through the execution of a computer program that optimizes the model's parameters using training data or prior experience. The resulting model can be used for predictive purposes, making future predictions, data-driven learning, or a combination of both.
2. The field of study known as machine learning focuses on how to create computer programmes that enhance themselves over time.

Definition of learning

When the performance of a computer programme on tasks T improves with experience E, as measured by P, it is considered to have learned from experience E within a specific class of tasks T and performance measure P.

Examples

1. Handwriting recognition learning problem
   - Task T: Recognising and classifying handwritten words within images
   - Performance P: Percent of words correctly classified
   - Training experience E: A dataset of handwritten words with given classifications
2. A robot driving learning problem
   - Task T: Driving on highways using vision sensors
   - Performance measure P: Average distance traveled before an error
   - training experience: A sequence of images and steering commands recorded while observing a human driver
3. A chess learning problem
   - Task T: Playing chess
   - Performance measure P: Percent of games won against opponents
   - Training experience E: Playing practice games against itself

A machine learning program, also known as a learning program or learner, is software designed to acquire knowledge through experience.

This process commences with observations or data, such as examples, firsthand experience, or instruction. The program searches for patterns in the data to draw conclusions based on the provided examples. The ultimate goal of machine learning is to enable computers to learn autonomously, adapt their behaviour accordingly and function without the need for human intervention.

The significance of machine learning lies in its ability to solve problems faster and on a larger scale compared to human intelligence. By directing immense computing power towards a single or multiple tasks, machines can be programmed to recognize patterns and correlations in incoming data and automate routine tasks efficiently.

Machine Learning is Widely Adopted

Machine learning is not a futuristic concept; it is already being actively employed across various industries to drive innovation and improve operational efficiency. In 2021, 41% of organisations accelerated their adoption of AI due to the pandemic, joining the 31% that already had AI technologies in production or were evaluating them.

Some prominent applications of machine learning include:

- **Data Security:** Machine learning algorithms can identify data security vulnerabilities

Notes

before they lead to breaches. By leveraging past experiences, these algorithms can predict high-risk activities, enabling proactive risk management.

- **Finance: Banks,** trading brokerages and Fintech companies utilise machine learning algorithms to automate trading and provide financial advisory services to investors. For example, Bank of America's chatbot Erica automates client support.
- **Healthcare:** Machine learning is used to analyse vast healthcare datasets, accelerating the discovery of new therapies, and improving patient outcomes. Routine tasks can also be automated to eliminate human error. IBM's Watson, for instance, employs data mining to provide physicians with personalized patient care information.
- **Fraud Detection:** In the financial and banking sectors, artificial intelligence evaluates large transaction volumes to detect fraudulent activities in real time. Machine learning-based fraud detection systems reduce investigation time by 70% and improve detection accuracy by 90%, according to Capgemini, a technology consulting firm.
- **Retail:** AI researchers and developers utilise machine learning algorithms to create AI recommendation engines that offer tailored product suggestions based on past purchases, historical, regional, and demographic data.

Applications of Machine Learning

Data mining involves the application of machine learning technologies to vast databases, aiming to analyse a large quantity of data and construct fundamental models with high predicted accuracy. Here are some prevalent applications of machine learning:

1. In the retail industry, machine learning is employed to analyse consumer behaviour.
2. Banks use historical data to build models for loan applications, fraud detection and stock market transactions.
3. Manufacturing utilises learning models for optimization, control, and troubleshooting.
4. Machine learning programs aid in medical diagnoses within the field of medicine.
5. Telecommunications analyse call patterns for network optimization and improved service quality.
6. In fields like physics, astronomy and biology, machine learning is indispensable for evaluating massive amounts of data rapidly, as the World Wide Web continuously expands, making manual searches impractical.
7. Artificial intelligence uses machine learning to train systems to learn and adapt to change, eliminating the need for creators to anticipate and provide solutions for every possible scenario.
8. Machine learning plays a vital role in solving problems related to vision, speech recognition and robotics.
9. Machine learning techniques are applied in the design of computer-controlled vehicles to steer appropriately on different roadways.
10. Machine learning methods are used in programming games like chess, backgammon and Go.

1.1.4 Machine Learning: How and What It Does?

Basic components of Learning Process

Whether conducted by a human or a machine, the learning process consists of four

Notes

components are data storage, abstraction, generalisation, and evaluation. The components and phases involved in the learning process are depicted in the diagram below.

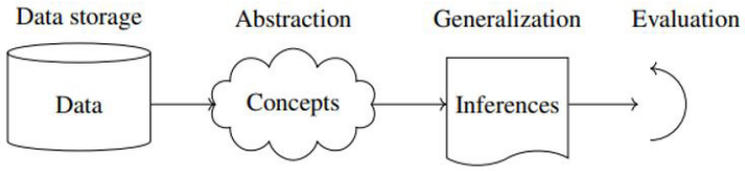


Figure: Components of learning process

Data Storage

Both humans and computers rely on data storage to retain and retrieve vast amounts of information, which is crucial to the learning process.

- In the human brain, information is stored and retrieved through electrochemical impulses.
- Computers store data using hard disc drives, flash memory, random access memory and other similar devices and they retrieve data through cables and other equipment.

Abstraction

Abstraction involves eliminating unnecessary information from stored data, creating comprehensive concepts that represent the entire data set. This process includes applying known models and developing new ones to generate knowledge. The trained model transforms the data into an abstract form that summarizes the original information.

Generalization

Generalization is the process of transforming recorded factual knowledge into a form applicable to future actions. It applies to similar tasks, even if not identical to the ones seen before. The goal of generalization is to identify data properties that will be beneficial for future endeavors.

Evaluation

Evaluation is the final step in the learning process, providing users with feedback to assess the usefulness of newly acquired knowledge. This feedback is used to improve and enhance the overall learning process.

The key difference between AI and ML are:

Artificial Intelligence	Machine learning
Artificial Intelligence (AI) is a technology that empowers machines to imitate human behaviour.	Machine learning, as a subset of artificial intelligence, allows machines to learn without the need for explicit programming.
The primary objective of AI is to create computer systems capable of solving complex problems in a human-like manner.	Machine learning aims to empower machines to learn from data and produce precise outputs.
Within artificial intelligence, the focus is on constructing intelligent computers that can perform various tasks similar to humans.	Machine learning involves using data to train machines in performing tasks and producing accurate results.
AI is divided into two major subgroups: machine learning and deep learning.	Deep learning is a subcategory within the broader field of machine learning.

Notes

The applications of AI are vast and diverse.	Machine learning has certain limitations in its utility.
The ultimate goal of AI is to develop a system intelligent enough to handle a wide range of complex tasks, increasing the likelihood of success.	The objective of machine learning is to create machines that can perform tasks beyond those explicitly programmed for them.
The AI system is designed to increase the probability of success.	Machine learning primarily aims to achieve precision and identify patterns.
Some common examples of AI applications include Siri, customer service through chatbots, Expert Systems, online game play, intelligent humanoid robots and more.	Machine learning finds applications in various areas, including online recommender systems, Google search algorithms and Facebook auto-friend labelling suggestions.
AI can be classified into three types based on their capabilities: weak AI, general AI, and strong AI.	Machine learning can be categorised into three major groups: supervised learning, unsupervised learning, and reinforcement learning.
AI encompasses the abilities of learning, reasoning, and self-correction.	When presented with new data, AI utilises learning and self-correction mechanisms.
AI is involved in handling data in structured, semi-structured and unstructured formats.	Machine learning is applied to treat both structured and semi-structured data.

Real-world applications of machine learning include:

- **Speech Recognition:** Using natural language processing (NLP), machines convert human voice into text, enabling voice queries and messaging accessibility on mobile devices (e.g., Siri).
- **Customer Service:** Chatbots are replacing human agents in customer engagement across websites and social media platforms, providing personalized advice and responding to frequently asked questions.
- **Computer Vision:** This AI technique extracts valuable information from digital images and videos, with applications in photo labelling, radiological imaging, and self-driving vehicles.
- **Recommendation Engines:** AI algorithms use consumer behaviour data to identify trends and generate effective cross-selling strategies, enabling online retailers to recommend relevant products to customers.
- **Automated Stock Trading:** AI-powered high-frequency trading platforms optimize stock portfolios and execute millions of transactions without human intervention each day.

1.1.5 Deep Learning: How Is It Different

The human brain is the most amazing organ in the body. It shapes how we perceive what we see, hear, smell, taste, and touch. It allows us to remember things, feel emotions and even dream. We would be primordial organ isms without it, incapable of more than the most basic responses. Inherently, the brain is what makes us intelligent.

Even though the infant brain is only a pound in weight, it manages to solve issues that even our most powerful supercomputers can't. Infants can recognise their parents' features, distinguish separate objects from their surroundings and even distinguish voices within months after life. They've gained an intuition for natural physics in less than a year, can track things even when they're partially or entirely obstructed and can correlate

Notes

noises with specific meanings. They have a comprehensive comprehension of grammar and hundreds of words in their vocabularies by early childhood.

For decades, we've fantasised of creating intelligent machines with minds like ours: robotic cleaning assistants, self-driving automobiles, and disease-detecting microscopes. However, creating artificially intelligent machines necessitates solving some of the most difficult computing problems we've ever faced, issues that our brains can already solve in microseconds. To address these issues, we'll need to create a completely new approach of programming a computer, based on techniques that have mostly been created in the last decade. Deep learning is a popular term for a branch of artificial computer intelligence that is quite active.

Deep learning is a specialized branch of machine learning that revolves around neural networks with multiple layers. These networks attempt to mimic the functioning of the human brain by learning from vast amounts of data, although their capabilities are still far from matching the complexity of the human brain. While a neural network with a single hidden layer may provide approximate predictions, additional hidden layers can significantly improve accuracy and optimization.

Many artificial intelligence (AI) applications and services leverage deep learning to enhance automation, performing analytical and physical tasks without human intervention. Everyday products like digital companions, voice-enabled TV remotes and credit card fraud detection, as well as future innovations like self-driving vehicles, utilise deep learning technology.

In deep learning, algorithms are structured hierarchically, with increasing complexity and abstraction, unlike the linear structure of traditional machine learning algorithms. This hierarchy allows for the automation of predictive analytics in a more sophisticated manner.

A helpful analogy for understanding deep learning is observing how a young child learns the concept of "dog." As the child points to different objects, the parent confirms or corrects their understanding of what a dog is. With each interaction, the child gains a deeper comprehension and constructs a hierarchy of abstractions, building upon prior knowledge. Similarly, deep learning algorithms learn progressively complex representations of data, enabling them to recognize patterns and make predictions.

How Deep Learning Works

Deep learning algorithms undergo a process similar to a child learning to recognize a dog. Each algorithm in the hierarchy transforms its input in a nonlinear way before producing a statistical model as its output. This iterative process continues until the output becomes precise enough to be useful and the term "deep" originates from the number of layers the data must traverse.

In traditional machine learning, the learning process is supervised, where the programmer explicitly instructs the computer what features to search for when determining if an image contains a dog. This process, known as feature extraction, heavily relies on the programmer's ability to precisely characterize a feature set for a dog, which can impact the success rate of the computer's predictions. The advantage of deep learning is that the program autonomously constructs the feature set without human intervention, making use of unsupervised learning, which is not only faster but also more accurate in many cases.

Initially, the computer program is provided with training data, a collection of labelled photographs indicating whether they contain a dog. The program uses this information

Notes

to develop a feature set for dogs and a predictive model. In the beginning, the program's model may be simplistic, identifying anything with four legs and a tail as a dog, as it only looks for pixel patterns in the digital data. However, with each iteration, the prediction model becomes more complex and precise.

Deep learning systems require vast amounts of training data and processing power to achieve acceptable accuracy levels. The availability of big data and cloud computing has facilitated deep learning's ability to generate accurate predictive models from vast amounts of unstructured data. As the internet of things (IoT) becomes more prevalent, this capability is crucial, as much of the collected data is unstructured and unannotated.

Deep neural networks, also known as artificial neural networks, emulate the human brain by using data inputs, weights, and biases to recognize, categorise and characterize data items. These networks consist of multiple layers of interconnected nodes and forward and backward propagation techniques are used to train the model. Forward propagation progresses through the network layers, while back propagation computes prediction errors and adjusts weights and biases to improve accuracy.

Convolutional neural networks (CNNs) are commonly used in computer vision and image classification tasks, recognizing patterns and characteristics in images for object detection and recognition. Recurrent neural networks (RNNs) are utilised in natural language and speech recognition applications, particularly for sequential or time series data. Deep learning techniques are highly complex and various neural network types are used to address specific problems or datasets, each with its own strengths and applications. For example, in 2015, a CNN outperformed humans in an object identification competition for the first time.

Deep Learning Methods

Various approaches can be employed to build robust deep learning models, encompassing learning rate decay, transfer learning, training from scratch and dropout.

Learning Rate Decay:

The learning rate is a crucial hyperparameter that influences the extent of model adjustments in response to estimated errors when updating its weights. It is determined before the learning process and plays a significant role in defining the system's behaviour. If the learning rate is too high, it may lead to unstable training or suboptimal weight configurations. Conversely, overly slow learning rates can cause training to become time-consuming and stall progress.

To address these challenges, the learning rate decay method, also known as adaptive learning rates or learning rate annealing, comes into play. This technique aims to enhance performance while reducing training time. Commonly used approaches involve gradually reducing the learning rate during training, which effectively adjusts the model's behaviour and facilitates better convergence to optimal solutions.

Transfer Learning: Transfer learning is a technique that involves improving an existing trained model by accessing its internal structure. Initially, users provide new data to the model, including previously unidentified categories. After modifying the model, it becomes capable of performing new tasks with greater accuracy in categorisation. The advantage of transfer learning is that it requires far less data compared to other methods, resulting in significantly reduced computation time, typically taking only minutes or hours.

Training From Scratch: Training from scratch involves gathering a substantial amount of labelled data and setting up a network architecture capable of learning the model's intricacies. This approach is particularly beneficial for new applications or those

Notes

with numerous output categories. However, it is not as commonly used due to the significant data requirement, which can lead to lengthy training periods lasting days or even weeks.

Dropout: The dropout method addresses the issue of overfitting in neural networks with a vast number of parameters by randomly deactivating units and their connections during training. This technique has been proven to enhance the performance of neural networks in supervised learning tasks, such as speech recognition, document categorisation and computational biology. By employing dropout, the network becomes more robust and generalizes better to unseen data, leading to improved accuracy and efficiency in various applications002E

Deep Learning Neural Networks

Deep learning models, a sophisticated form of machine learning, are primarily built upon artificial neural networks. Hence, deep learning is often referred to as deep neural learning or deep neural networking.

Various forms of neural networks, such as recurrent neural networks, convolutional neural networks, artificial neural networks, and feedforward neural networks, offer specific advantages for different applications. They all function by inputting data into the model and allowing it to interpret and evaluate the data elements.

Neural networks rely on a trial-and-error process during training, which demands a substantial amount of training data. The popularity of neural networks surged as businesses embraced big data analytics and accumulated vast datasets. During the initial training iterations, the model makes educated guesses about the contents of data elements like images or speech sections. To assess the accuracy of these assumptions, the training data must be labelled. Consequently, although many corporations have extensive data repositories, unstructured data is less useful for deep learning as it requires labelled data for training to achieve acceptable accuracy levels. Deep learning models cannot be trained on unstructured data until they have achieved the necessary level of accuracy through supervised learning with labelled data.

Applications Based on Deep Learning

Deep learning applications have seamlessly become an integral part of our daily lives, embedded in various products and services without users realizing the intricate data processing happening in the background. Here are some instances where deep learning is widely employed:

Law Enforcement

Deep learning algorithms are used to analyse transactional data, enabling the detection of potential fraudulent or illicit activities. These algorithms can extract valuable patterns and evidence from various sources, such as audio and video recordings, images, and documents. By incorporating speech recognition and computer vision, deep learning applications enhance the efficiency and accuracy of investigative analysis, empowering law enforcement to analyse vast amounts of data swiftly and accurately.

Services in the Financial Sector

Financial institutions leverage predictive analytics powered by deep learning to support algorithmic stock trading, evaluate business risks for loan approvals, identify potential fraud and provide clients with personalized credit and investment portfolio management services.

Notes

Customer Service

Many organisations have embraced deep learning technologies to enhance their customer service operations. One popular application is the use of chatbots, which are a form of artificial intelligence that can be found in various applications, businesses, and customer service platforms. These chatbots come in different levels of complexity, with traditional ones resembling call center menus and utilising natural language and visual recognition.

Advanced chatbot solutions, powered by machine learning, are adept at handling ambiguous questions by evaluating multiple potential responses. Depending on the input they receive, these chatbots can either directly answer queries or seamlessly transfer the conversation to a human operator if necessary.

Moreover, virtual assistants like Apple's Siri, Amazon Alexa and Google Assistant have taken the chatbot concept a step further by incorporating speech recognition capabilities. This advancement enhances consumer engagement, providing a more personalized and interactive experience for users.

Healthcare

The healthcare sector has experienced significant benefits from embracing deep learning techniques, particularly in the realm of digitizing medical records and images. Through the use of image recognition software, medical imaging specialists and radiologists can now analyse and assess a greater volume of images in less time, resulting in enhanced efficiency and effectiveness in their tasks.

Machine Learning Vs. Deep Learning

Deep learning is a specialized branch of machine learning that distinguishes itself by the types of data it handles and the learning algorithms it employs. In conventional machine learning, predictions are derived from structured, labelled data organised in tables with defined features. While machine learning can also handle unstructured data, it often requires preprocessing to convert it into a structured format.

To illustrate the relationship between AI, machine learning, neural networks, and deep learning, you can think of them as Russian nesting dolls, where each term is a subset of the one before it. This hierarchy demonstrates how deep learning is a specialized branch of machine learning and machine learning is a subset of AI and so on.

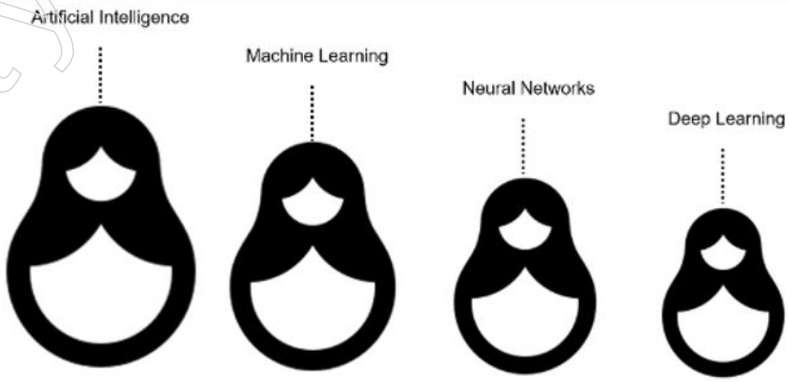


Figure: Relationship between AI, machine learning, neural networks, and deep learning

Deep learning is a subset of machine learning and what sets it apart is the way each algorithm learns and the amount of data they can handle. In deep learning, a significant portion of the feature extraction process is automated, reducing the need for human

Notes

intervention. This automation allows deep learning models to handle massive datasets, making it highly scalable for large-scale machine learning tasks. This capability becomes particularly advantageous as organisations increasingly deal with unstructured data, which makes up a substantial portion of their data (around 80-90%).

In contrast, classical machine learning, often referred to as “non-deep” machine learning, requires more human involvement. Experts need to construct hierarchies of features to understand the differences between data inputs and they often rely on structured data for learning. For example, when presented with images of fast-food items like pizza, burger and taco, a human specialist would manually identify distinguishing features, such as the appearance of bread in each food variety.

1.2 Linear and Logistic Regression

Regression in machine learning is a supervised learning technique used for predicting a continuous output variable (also known as a dependent variable) based on one or more input features (independent variables). The primary goal of regression is to establish a mathematical relationship between the input variables and the continuous target variable, allowing you to make predictions or estimate values for the target variable given new input data.

In a regression problem, you have a dataset consisting of input-output pairs, where the input features are used to predict the corresponding output. The output variable in regression can take on a wide range of values, making it a continuous variable. Some common examples of regression tasks include:

Linear Regression: Linear regression is one of the simplest and most widely used regression techniques. It assumes a linear relationship between the input features and the target variable. The goal is to find the best-fitting linear equation (a straight line in the case of simple linear regression) that minimizes the difference between predicted and actual values.

Multiple Linear Regression: This is an extension of linear regression when there are multiple input features. The relationship between the inputs and the output is modeled as a linear combination of the features.

Polynomial Regression: In cases where the relationship between the variables is nonlinear, polynomial regression can be used. It fits a polynomial function to the data instead of a straight line.

Ridge Regression and Lasso Regression: These are regularization techniques used to prevent overfitting in linear regression. They introduce penalty terms to the linear regression equation to constrain the model’s complexity.

Support Vector Regression (SVR): SVR is a regression technique that uses the principles of support vector machines to find a hyperplane that best fits the data, with a margin of tolerance for error.

Decision Tree Regression: Decision trees can also be used for regression tasks. They partition the feature space into regions and assign an output value to each region.

Random Forest Regression: Random forests combine multiple decision trees to improve prediction accuracy in regression tasks.

Gradient Boosting Regression: Gradient boosting algorithms like Gradient Boosting Regression Trees (GBRT) build an ensemble of weak learners (typically decision trees) to create a strong predictive model.

Notes

Regression is widely used in various domains, including finance (predicting stock prices), economics (predicting economic trends), healthcare (predicting patient outcomes), and many other fields where predicting continuous values is essential. The choice of regression technique depends on the specific problem, the nature of the data, and the underlying assumptions about the relationship between the variables.

Linear regression is a statistical technique used to analyse the relationship between one or more independent variables (often denoted as x) and a dependent variable (usually denoted as y). The primary aim of linear regression is to identify the most suitable linear relationship between the independent and dependent variables.

In simple linear regression, there is only one independent variable and the connection between the dependent and independent variables is assumed to be linear. The fundamental equation for a linear regression model can be expressed as follows:

$y = b_0 + b_1 * x + e$

where:

y is the dependent variable

x is the independent variable

b_0 is the intercept or constant term

b_1 is the slope coefficient

e is the error term or residual

The primary goal of linear regression is to determine the intercept and slope coefficients that minimize the sum of squared errors between the predicted and actual values of the dependent variable. This process is commonly performed using the Ordinary Least Squares (OLS) regression method.

In multiple linear regression, an extension of simple linear regression, the model involves several independent variables. The equation for a multiple linear regression model can be expressed as follows:

$y = b_0 + b_1 * x_1 + b_2 * x_2 + ... + b_n * x_n + e$

where:

y is the dependent variable

$x_1, x_2, ..., x_n$ are the independent variables

b_0 is the intercept or constant term

$b_1, b_2, ..., bn$ are the slope coefficients

e is the error term or residual

In multiple linear regression, the aim is to calculate the values of the intercept and slope coefficients based on the provided values of the independent variables. These coefficients are determined in such a way that they minimize the sum of squared errors between the predicted values and the actual values of the dependent variable.

1.2.1 Introduction to Linear Regression

Linear regression is a supervised machine learning model widely used for predictive modelling. In supervised learning, the model is trained using a labelled dataset to learn and optimize its parameters through a loss function. Linear regression is particularly

Notes

popular for time series forecasting and involves assuming a linear relationship between a set of independent variables and the dependent variable of interest.

To approximate the given data, regression and log-linear models are common choices. In basic linear regression, the data is modeled to fit a straight line. For example, the equation represents a random variable, y (the response variable), as a linear function of another random variable, x (the predictor variable).

$y = wx + b$

In data mining, the linear regression model assumes that the dependent variable, y, exhibits a constant variance. Both x and y are numerical attributes in the database. The model includes regression coefficients, denoted as w and b, representing the slope and y-intercept of the line, respectively. These coefficients are calculated using the method of least squares, which seeks to minimize the discrepancy between the actual data distribution and the estimated line.

Multiple linear regression is an expansion of basic linear regression, allowing the response variable, y, to be modeled as a linear function of two or more predictor variables. This extension enables the consideration of multiple factors simultaneously, providing a better understanding of how they collectively influence the dependent variable.

Log-linear Models:

Log-linear models serve as approximations for discrete multidimensional probability distributions. In these models, each tuple in an n-dimensional set can be envisioned as a coordinate within that space, with n attributes describing each tuple. Utilising log-linear models enables the estimation of probabilities for each point in the multidimensional space, given a discretized set of characteristics based on a limited subset of dimensional combinations. This approach facilitates the construction of a higher-dimensional data space by combining existing data sets with additional information.

The significance of log-linear models lies in their ability to smooth data and reduce dimensionality. As a result, aggregate estimates in the lower-dimensional space are less impacted by sampling variations compared to estimates in the higher-dimensional space. This practical feature makes log-linear models valuable for handling complex data structures and enhancing data analysis.

On sparse data, both regression and log-linear models are applicable, but their utility is diminished. Although both methods can manage skewed data, regression is more adept at doing so. When working with high-dimensional data, regression may be computationally expensive, but log-linear models scale well up to 10 dimensions.

Simple Linear Regression

Basic linear regression focuses on understanding the relationship between a single independent variable, also known as an input and a corresponding dependent variable, also known as an output.

Limitations of Simple Linear Regression

Despite providing accurate information, regression analysis may not present the complete picture. It is frequently employed in research to demonstrate relationships between variables. However, it is essential to recognize that a correlation between two factors does not necessarily imply causation. Just because two variables show a relationship in a regression analysis does not mean that one directly causes the other to occur. Even in a simple linear regression, where the data points align closely with the

Notes

regression line, it is not always indicative of a cause-and-effect relationship between external factors and logical outcomes. Therefore, interpreting regression results should be done with caution and further investigation is necessary to establish causation and understand the full complexity of the underlying relationships.

You may ascertain if there is any link between variables using a linear regression model. It will be necessary to do more research and statistical analysis to ascertain the precise nature of the relationship and if one variable is the cause of the other.

1.2.2 Linear Regression: Important Terms and Assumptions

Linear Regression is the most fundamental algorithm in machine learning, and it can be considered the introduction to the field. It will be beneficial if you have a good understanding of linear regression before moving on to more complicated methods.

Call

In this analysis, we utilised a formulaic function to examine the relationship between two variables: “disp” as the dependent variable and “mpg” as the response variable. The data for the response variable, “mpg,” was extracted from the “mtcars” data frame. Through this formulaic function, we aimed to understand how the values of “disp” influence the values of “mpg” and explore any potential correlations between these two variables.

Residuals

In our model, the residuals signify the vertical distance between each data point and the fitted line. We have collected summary data for all these vertical distances in this example. A smaller value for these residuals indicates a better fit of the model, implying that it accurately captures the relationships between the variables and the data points.

Coefficients

These are the calculated coefficients for our linear equation, which are shown below in a list. In this example, our equation would look like this: y = 0.04x + 29.59.

- **Std. Error:** The coefficients table’s Std. Error component gives error estimates for those coefficients. Our equation would seem as follows: y = (−0. 04 ± 0. 005)x + (29. 59 ± 1. 23).
- **t-value:** The difference between our data and the variation is measured in this way. While they are similar, p-values are far more often used.
- **p-value:** P-values measure statistical significance. A p-value of less than 0.05 shows statistical significance. If the p-value is more than 0.05, we should assume statistical significance. Importance codes explain star ratings.

Best Fit Line

The line of best fit represents the relationship between the independent and dependent variables. It is used to predict the dependent variable based on the independent variable(s). The primary goal of the line of best fit is to minimize the distance between the observed data points and the predicted values, aiming to achieve the closest possible match between the two.

Error

The best fit line will provide us with predicted values. Error, in this context, refers to the difference between the observed or expected values and the actual values.

Notes

Mean Absolute Error: L1 loss

The mean absolute error is calculated by averaging the total absolute error of each example over the number of data points. We can prevent cancelling out mistakes that are too high or too low the genuine values by adding up all absolute values of errors in a model and getting an overall error metric to evaluate a model on by adding up all absolute values of errors in a model. The inaccuracy can be anywhere between 0 and ∞.

Mean Squared Error: L2 loss

The furthermost extensively used error to assess model performance is the mean squared error. The residual error (the difference between the anticipated and true value) is squared, unlike the absolute error. For mean squared error loss, the range is also 0 to ∞. Squaring the residual error offers the advantages of having positive error terms, emphasising greater errors over smaller errors and being differentiable. Being differentiable allows us to apply calculus to identify minimum and maximum values, which can save time in the long run.

Huber Loss

The Huber loss is less sensitive to data outliers compared to the squared error loss. It is also differentiable at 0. When the error is small, it behaves like an absolute error, but as the error increases, it transitions to a quadratic form. The hyperparameter (delta) determines the threshold at which the transition occurs, and it can be adjusted to control the balance between the absolute and quadratic error components.

R-Squared

It's also referred to as the coefficient of determination. R squared values range from - to 1 and they assist us figure out if our model can explain variation in Y by variation in X. A model with a R squared of 0 is equivalent to one that always forecasts the target variable's mean, whereas a model with a R squared of 1 completely predicts the target variable.

$$R^2 = 1 - \sum \frac{(Y_{actual} - Y_{predicted})\text{^}2}{(Y_{actual} - Y_{mean})\text{^}2}$$

Adjusted R-Squared

If we add additional predictors to our model, R-Squared will rise or remain constant. So, there's no way of knowing whether increasing the model's complexity makes it more accurate. The R-Squared calculation is tweaked to account for the number of predictors in the model. Only if the new term improves model accuracy does the modified R-Square increase.

$$R^2 = 1 - \sum \frac{(Y_{actual} - Y_{predicted})\text{^}2}{(Y_{actual} - Y_{mean})\text{^}2}$$

Assumptions on Linear Regression

Linear regression is a commonly employed statistical technique used to establish the connection between a dependent variable and one or more independent variables. However, for the outcomes of a linear regression analysis to be considered valid and trustworthy, several assumptions need to be met. These assumptions encompass:

Linearity: In linear regression, one of the fundamental assumptions is the existence of a linear relationship between the dependent variable and each independent variable. This assumption implies that the slope of the regression line remains consistent regardless of the specific value of the independent variable used for regression. Ensuring

Notes

a constant linear association between the variables enables the model to make reliable predictions and valid inferences based on the data.

Independence of errors: It is assumed that the errors—that is, the discrepancies between the dependent variable's actual values and its anticipated values based on the regression equation—are unrelated to one another. To put it another way, the error term's value for one observation should not be influenced by the error term's value for any other observation.

Homoscedasticity: In regression analysis, a crucial assumption is that the standard deviation of the errors remains constant, regardless of the values of the independent variable. In simpler terms, the residuals, which represent the differences between the observed and predicted values of the dependent variable, should have a consistent distribution across all levels of the independent variable. Upholding this assumption is essential to ensure the reliability and validity of the regression model. It allows for more accurate and meaningful interpretations of the relationship between the variables.

Normality: It is presumable that the mistakes will follow a normal distribution. This indicates that the distribution of the residuals needs to take on the form of a bell curve and be symmetrical.

No multicollinearity: The independent variables in multiple regression shouldn't have a strong correlation with one another. Multicollinearity may make it challenging to estimate the model's parameters and can provide results that are unstable and untrustworthy.

Violation of these assumptions can lead to biased and inefficient estimates of the parameters of the regression model, as well as inaccurate predictions and hypothesis tests. Therefore, it is important to carefully check these assumptions before conducting a linear regression analysis and to take appropriate corrective actions if any of the assumptions are violated.

To perform a simple linear regression, certain assumptions about the data must be made. This is since it is a parametric test. The following are the assumptions made when running a simple linear regression:

- **Homogeneity of variance (homoscedasticity)-** The amount of the error stays constant is one of the primary expectations of a simple linear regression approach. This basically indicates that the error magnitude does not change much while the independent variable's value varies.
- **Independence of observations**- Nothing is hidden in the relationships between the observations and only valid sampling procedures are employed during data gathering.
- **Normality**- The data is flowing at a typical rate.

However, while running a linear regression, there is one more assumption that must be considered.

The line is always a straight line: During the execution of a linear regression, there is no curve or grouping factor. The variables are linked in a linear fashion (dependent variable and independent variable). A nonparametric test may be employed if the data violates the assumptions of homoscedasticity or normality. (Take, for instance, the Spearman rank test.)

The following is an example of data that does not match the assumptions: One might believe that the consumption of cured pork and the incidence of colorectal cancer in the United States are linked. However, it is later discovered that there is a significant range

Notes

disparity in the data collection for both variables. There can be no linear regression test since the homoscedasticity condition is violated. A Spearman rank test, on the other hand, can be used to determine the link between the variables.

Homoscedasticity

Explanation

The following assumption of linear regression states that residual variance is constant at all x levels. The term for this is homoscedasticity. When this is not the case, it is presumed that heteroscedasticity exists in the residuals.

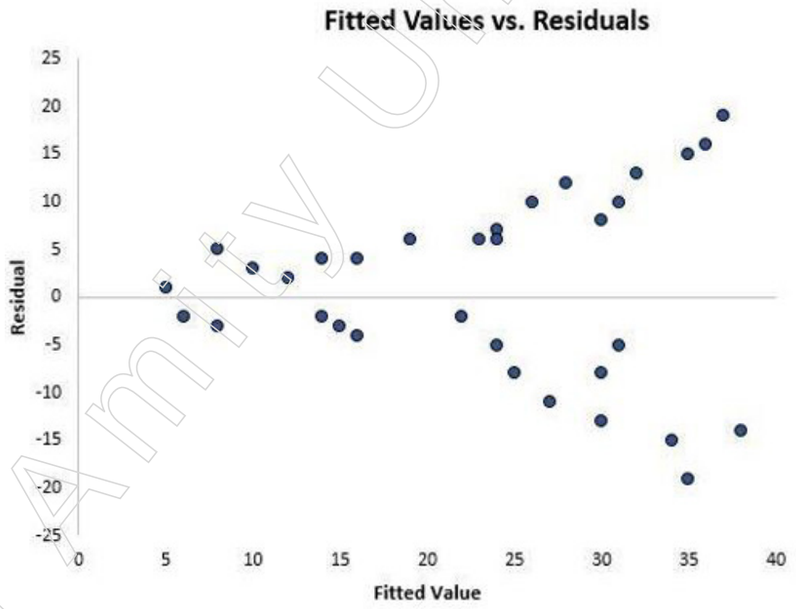
The presence of heteroscedasticity in a regression analysis poses challenges in interpreting its outcomes. Heteroscedasticity refers to a situation where the variability of the errors (residuals) is not constant across different values of the independent variable. As a result, the regression model underestimates or overestimates the uncertainty in the estimated regression coefficients, leading to potential misinterpretations.

Specifically, heteroscedasticity can increase the variance of the estimated coefficients, making them less reliable. The regression model assumes constant variance, but when heteroscedasticity exists, this assumption is violated. Consequently, the model may incorrectly identify certain terms as statistically significant when they are not and vice versa. This undermines the accuracy and trustworthiness of the regression analysis and requires careful consideration and remedial actions to ensure the validity of the results.

How can you tell whether this supposition is true?

The easiest method to identify heteroscedasticity is a fitted value vs. residual plot.

A scatterplot can be utilised to visualize the relationship between the fitted values of a regression model and the corresponding residuals. After fitting the regression line to a dataset, plotting the fitted values against their respective residuals can reveal patterns that indicate the presence of heteroscedasticity.

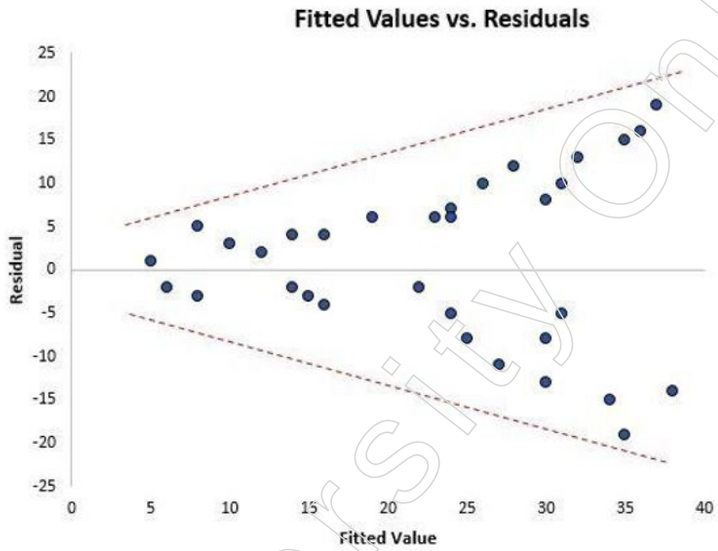


The scatterplot displayed below represents a typical example of a fitted value vs. residual plot with heteroscedasticity. In such a plot, if the spread or dispersion of the residuals varies noticeably as the fitted values change, it suggests that the assumption of constant variance is violated. This phenomenon is known as heteroscedasticity, and it

Notes

can have implications on the reliability and accuracy of the regression analysis. Detecting heteroscedasticity through the scatterplot helps in identifying potential issues and may lead to further investigation and adjustments in the model to ensure the validity of the results.

As the fitted values grow greater, the residuals become considerably more spread out. Heteroscedasticity is symbolised by this "cone" shape:



There are three common ways to fix heteroscedasticity:

1. **Transform the Dependent Variable**: Taking the log of the dependent variable is a typical transformation. For example, instead of predicting the number of flower shops in a city using population size (independent variable), we could use population size to predict the log of the number of flower stores in a city. Heteroskedasticity is often eliminated by using the log of the dependent variable rather than the original dependent variable.
2. **Redefine the Dependent Variable**: Instead of using the raw value, a rate is a typical technique to redefine the dependent variable. Instead of predicting the number of flower shops in a city based on population size, we might use population size to forecast the number of flower stores per capita. Because we're calculating the number of flower shops per person rather than the total number of flower businesses, this decreases the fluctuation that naturally exists across larger populations.
3. **Use weighted regression:** Weighted regression is another method for correcting heteroscedasticity. Each data point is given a weight based on the variance of its fitted value in this sort of regression. This reduces the squared residuals of data points with higher variances by assigning tiny weights to them. Heteroscedasticity can be eliminated when the appropriate weights are employed.

Normality

The residuals must be regularly distributed, according to the next premise of linear regression.

How do you know if this assumption is correct?

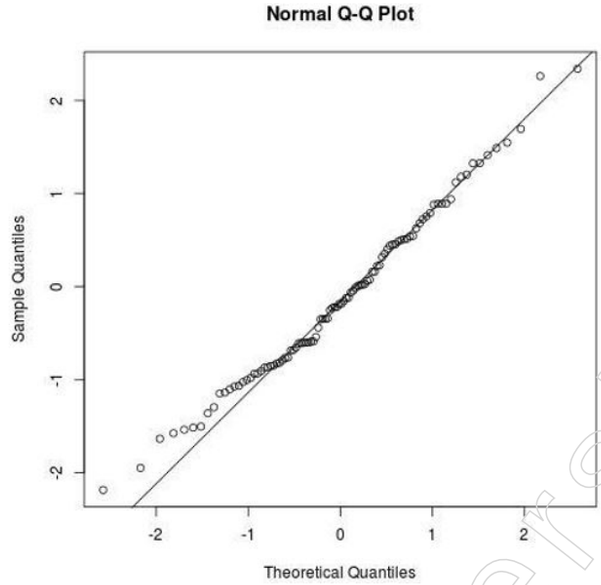
There are two popular methods for determining whether this assumption is correct:

1. Check the assumption visually using Q-Q plots.

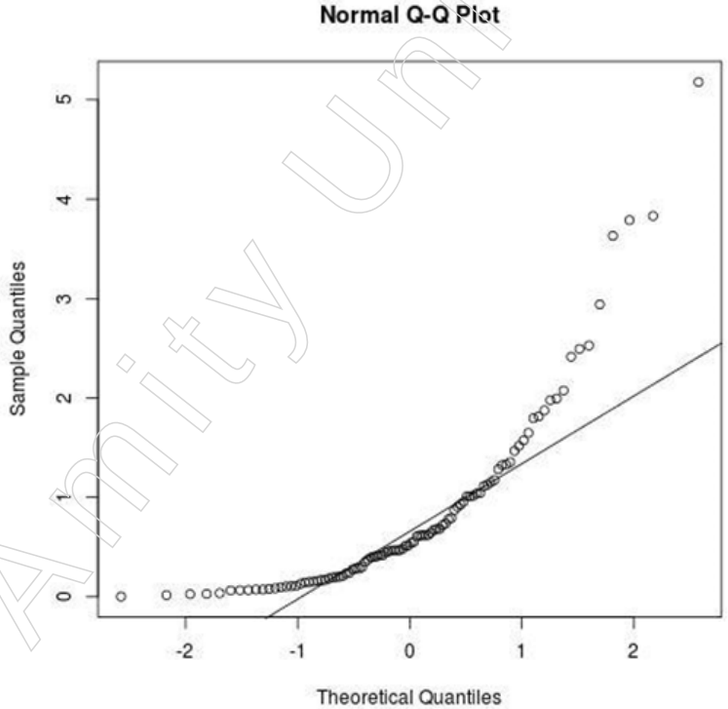
Notes

A Q-Q plot, also known as a quantile-quantile plot, is a kind of graph that we may use to determine whether a model's residuals follow a normal distribution. If the plot's points often form a straight diagonal line, the normality assumption is met.

The Q-Q graphic below is an example of residuals that almost follow a normal distribution:



The Q-Q plot below, on the other hand, illustrates an example of when residuals obviously deviate from a straight diagonal line, indicating that they do not follow a normal distribution:



2. To verify the normality assumption, you may also use formal statistical tests like Shapiro-Wilk, Kolmogorov-Smirnov, Jarque-Barre, or D'Agostino-Pearson. Nevertheless, keep in mind that these tests are sensitive to high sample sizes, which means that when your sample size is large, they commonly draw the conclusion that the residuals are

Notes

not normal. This is why using graphical techniques like a Q-Q plot often makes it simpler to test this assumption.

What should be done if this assumption is broken?

You have a few choices if the normalcy assumption is broken:

What should you do if this assumption is not true?

You have a few choices if the normalcy assumption is broken:

- Make sure that no outliers are significantly affecting the distribution before continuing. Verify again that any outliers are real figures and not the result of incorrect data entry if there are any.
- To tackle nonlinearity in the relationship between the independent and/or dependent variables, a possible approach is to employ nonlinear transformations. Common transformations involve taking the logarithm, square root, or reciprocal of the independent and/or dependent variable. These transformations can help reshape the data and potentially capture a more linear relationship, making the regression model more accurate and reliable.

Independence

According to the subsequent principle of linear regression, the residuals need to be free from influence from one another. This is particularly true when dealing with data that is organised in time series. In a perfect environment, we would like that there not be a pattern between each pair of successive residuals. For instance, the size of residuals should not grow with the passage of time.

How to determine if this assumption is met

To assess if the assumption of no serial correlation is met, a straightforward approach is to create a residual time series plot by plotting the residuals against time. Ideally, most of the residual autocorrelations should fall within the 95 percent confidence bands positioned around zero. These bands are typically at around +/- 2 divided by the square root of the sample size (n).

For a more formal evaluation, the Durbin-Watson test can be utilised to determine whether this assumption holds or not. This test provides a statistical measure to assess the presence of serial correlation in the residuals of the regression model.

What to do if this assumption is violated

When the assumption of no serial correlation in a regression model is violated, there are several options to address it:

- **For positive serial correlation:** If there is evidence of positive serial correlation (where the error terms are correlated across time), you can consider using lags of the dependent and/or independent variables in the model. Introducing lagged variables can help account for the correlation between successive observations.
- **For negative serial correlation:** To handle negative serial correlation (where the error terms exhibit a negative correlation across time), it's essential to ensure that none of the variables in the model are over-differenced. Over-differencing can create spurious negative serial correlation, so careful consideration of the differencing approach is necessary.
- **For seasonal correlation:** If there is evidence of seasonal correlation (where

Notes

the error terms show patterns related to seasonal variations), it might be appropriate to include seasonal dummy variables in the model. These dummy variables can help capture the effects of seasonality in the data and improve model accuracy.

By selecting the appropriate method based on the nature of the serial correlation present in the data, you can enhance the reliability and validity of your regression model.

1.2.3 Statistical View of Regression

Regression analysis is a fundamental statistical technique used to establish the connection between a dependent variable and a set of independent variables. Its primary objective is to estimate model parameters that can effectively predict the value of the dependent variable based on the independent variables.

The process involves fitting a curve or line to a set of data points, allowing for the prediction of additional data points. To assess the accuracy of the regression model, analysts often examine the residuals, which represent the differences between the observed and predicted values of the dependent variable.

Various types of regression models exist, including linear regression, logistic regression, and polynomial regression, each with its own assumptions and limitations. Selecting the most appropriate model for a given dataset requires careful consideration of these factors.

Regression analysis not only calculates model parameters but also enables hypothesis testing concerning the relationship between independent and dependent variables. Hypothesis tests can help determine whether a linear or higher-order polynomial provides the best description of the variable relationship.

Regression is a powerful method for modelling complex interrelationships and making predictions about future outcomes. However, it is crucial to interpret the results with caution, considering the assumptions and limitations of the chosen model.

In Minitab Statistical Software, after training and confirming the fit of the regression model using residual plots, analysts can proceed with analysing the results and deriving the equation describing the statistical relationship between predictor variables and the response variable.

How Do One Interprets The P-Values in Linear Regression Analysis?

In regression analysis, the p-value for each coefficient tests the hypothesis that the corresponding predictor has no effect on the response variable (null hypothesis). If the p-value is small (typically less than 0.05), we can reject the null hypothesis, indicating that the predictor is likely to be beneficial for the model. This implies that variations in the predictor's value are associated with changes in the response variable.

Conversely, a higher p-value (non-significant) suggests that changes in the predictor are not related to changes in the response variable. In such cases, we fail to reject the null hypothesis, indicating that the predictor may not be a significant contributor to the model's predictive power.

The fact that South and North's p-values in the result below are both 0.000 demonstrates the significance of these predictor factors. Although East's p-value of 0.092 is greater than the standard alpha threshold of 0.05, it still does not meet the criteria for statistical significance.

Notes

Coefficients

Term	Coef	SE Coef	T	P
Constant	389.166	66.0937	5.8881	0.000
East	2.125	1.2145	1.7495	0.092
South	5.318	0.9629	5.5232	0.000
North	-24.132	1.8685	-12.9153	0.000

To choose which terms to keep in the regression model, you usually utilise the coefficient p-values. We should consider deleting East from the model above.

How Do One Interprets the Regression Coefficients For Linear Relationships?

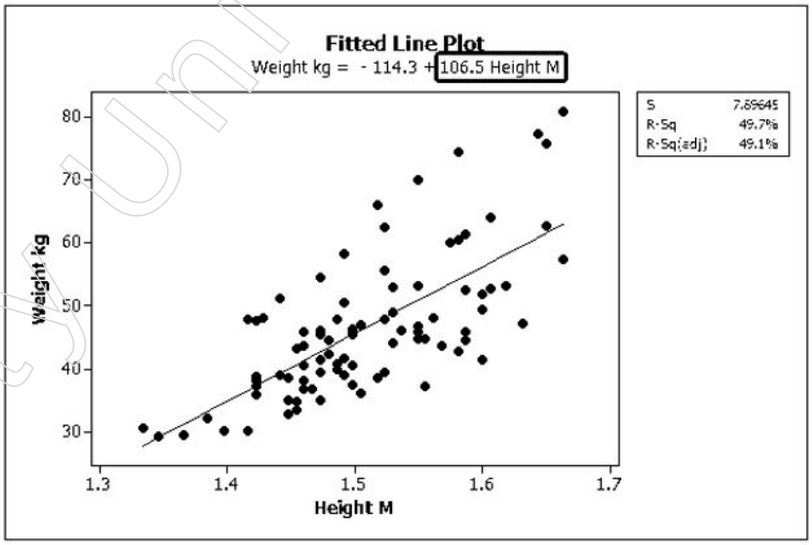
Regression coefficients indicate the mean change in the response variable for each unit of change in the predictor variable, if all other predictors in the model remain constant. Since it distinguishes the function of one variable from the roles performed by the other variables in the model, regression's statistical control is indispensable.

To comprehend the coefficients, they must be viewed as slopes, which is why they are frequently referred to as slope coefficients. In the fitted line plot that follows, it will demonstrate how this works by using a person's height to predict their weight. First, the output session window of Minitab:

Coefficients

Term	Coef	SE Coef	T	P
Constant	-114.326	17.4425	-6.55444	0.000
Height M	106.505	11.5500	9.22117	0.000

The fitted line plot graphically depicts the same regression results.



The calculation gives 106.5 kilograms for height in meters. The average weight gains every meter of height is 106.5 kg.

The blue fitted lines represent the same data. For every one-meter movement along the x-axis, the fitted line shows a corresponding change of 106.5 kg. This relationship is valid only within the range of 1.3 to 1.7 meters for middle-school females; outside this range, we shouldn't extrapolate the line by a meter up or down.

If the slope coefficient is 0, the projected weight remains constant regardless of the position on the fitted line. Therefore, a low p-value indicates that the slope is not zero,

Notes

suggesting that changes in the predictor variable have an impact on changes in the response variable.

Fitted line plots add a visual aspect to mathematical concepts. However, they are limited to illustrating basic regression findings involving one predictor variable and one response. For multiple linear regression, additional predictor variables would require a spatial dimension to display the results effectively.

How Do One Interprets the Regression Coefficients for Curvilinear Relationships and Interaction Terms?

In the previous example, height demonstrates a linear effect, as indicated by the constant slope along the fitted line, showing that the effect is consistent and continuous. However, when your model involves polynomial or interaction terms, the interpretation becomes more complex.

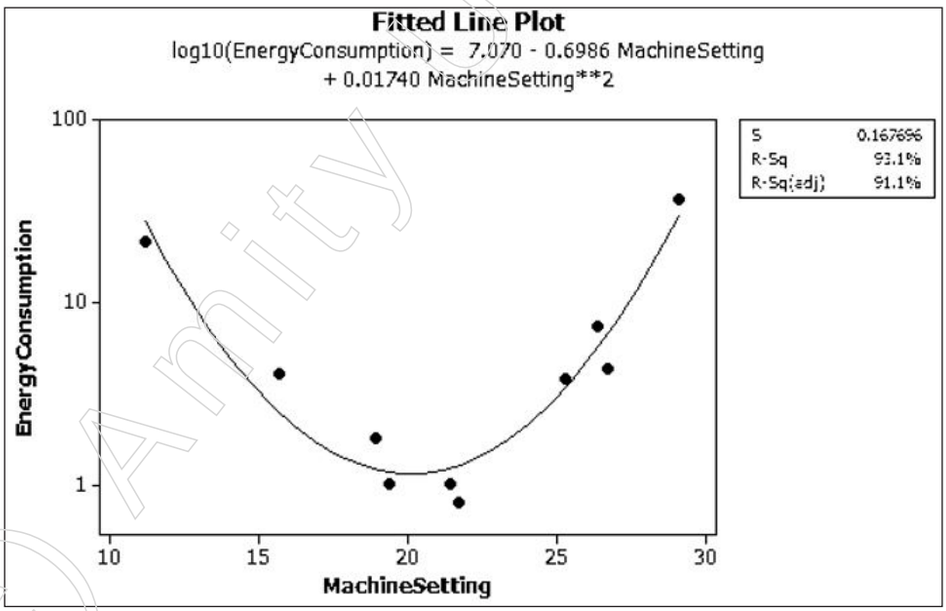
Interaction terms suggest that the impact of one predictor is dependent on the value of another predictor, introducing potential nonlinearity. On the other hand, polynomial terms account for the curvature of the data, capturing nonlinear relationships.

For describing curvature in the data, the example below includes a quadratic (squared) component. The results indicate that both the linear and quadratic terms have significant p-values, suggesting that both the straight-line relationship and the curvature are statistically significant and contribute to the predictive power of the model.

Coefficients

Term	Coef	SE Coef	T	P
Constant	7.06962	0.734504	9.62502	0.000
MachineSetting	-0.69863	0.074321	-9.40015	0.000
MachineSetting*MachineSetting	0.01740	0.001804	9.64557	0.000

We can proceed with the interpretation because the residual plots (not displayed) indicate a satisfactory fit. But how are these coefficients to be interpreted? Graphing it in a fitted line plot is quite helpful.



The relationship between machine parameters and energy consumption can vary based on the starting point on the fitting line. For example, if you begin with a machine setting of 12 and increase it by 1, energy consumption may decrease. On the other hand,

Notes

if energy consumption starts at 25, increasing the machine setting by one could result in higher energy consumption. Additionally, for individuals under 20 years old, energy consumption may not significantly vary.

The presence of a large polynomial component in the model can make interpretation difficult since the effect of modifying a predictor depends on its value. However, a significant interaction term indicates that the influence of one predictor is influenced by the magnitude of another.

When evaluating a regression model that incorporates these variables, caution is essential. Simply looking at the largest effect (linear term) may not provide a complete understanding of what is happening. Visual representation through a fitted line plot may not be suitable for multiple regression analysis, making subject-matter expertise crucial in interpreting the results accurately.

1.2.4 Logistic Regression

Logistic regression, originally employed in the biological sciences during the late 19th and early 20th centuries, has since found applications in various social scientific contexts. It is specifically used when dealing with a categorical dependent variable (target).

For instance,

- To determine if a message is spam (1) or (0)
- The tumor's malignancy (1) or lack thereof (0)

Imagine we are dealing with an email filtering scenario, where our goal is to determine whether an email is spam or not using linear regression. To make this classification, we need to select a threshold value that separates spam emails from non-spam ones based on projected continuous values.

For example, let's say a projected continuous value for a particular email is 0.4 and our chosen threshold value is 0.5. In this case, the email will be classified as non-spam, even though its actual class is spam. This decision of classification can have significant real-time implications, as misclassifying spam emails may lead to them landing in a user's inbox and causing inconvenience. Therefore, choosing the appropriate threshold value is critical to achieving accurate email classification and ensuring an efficient email filtering system.

The logistic regression model was created to use linear functions in x to express the posterior probability of the K classes while making sure that they sum to one and stay within the range [0, 1]. The simulation has a form:

$$\log \frac{\Pr(G=1|X=x)}{\Pr(G=K|X=x)} = \beta_{10} + \beta_1^T x$$

$$\log \frac{\Pr(G=2|X=x)}{\Pr(G=K|X=x)} = \beta_{20} + \beta_2^T x$$

$$\log \frac{\Pr(G=K-1|X=x)}{\Pr(G=K|X=x)} = \beta_{(K-1)0} + \beta_{K-1}^T x.$$

To specify the model, K-1 log-odds or logit transformations are used (reflecting the constraint that the probabilities sum to one). Even though the final class is used as the denominator in the odds-ratios, the selection of the denominator is left up to the

Notes

researcher since the estimates are consistent across all scenarios. The results of a fast computation show that

$$\Pr(G = k|X = x) = \frac{\exp(\beta_{k0} + \beta_k^T x)}{1 + \sum_{\ell=1}^{K-1} \exp(\beta_{\ell 0} + \beta_\ell^T x)}, \; k = 1, \ldots, K-1,$$

$$\Pr(G = K|X = x) = \frac{1}{1 + \sum_{\ell=1}^{K-1} \exp(\beta_{\ell 0} + \beta_\ell^T x)},$$

and they clearly sum to one. To illustrate the importance of the complete parameter set's interdependence θ = {, , . . . , , }, we denote the probabilities Pr(G = k|X = x) = pk(x; θ).

With K = 2, there is just one linear function, making this model straightforward. It is often used in applications of biostatistics that include a lot of binary answers (two classes). A condition may be present or absent, a patient may live or die, and they may have heart illness.

Fitting Logistic Regression Models

The conditional probability of G given X is often used to construct logistic regression models using maximum likelihood. The conditional distribution is fully characterised by Pr(G|X), hence the multinomial distribution is appropriate. The log-likelihood is for N observations.

$$\ell(\theta) = \sum_{i=1}^{N} \log p_{g_i}(x_i; \theta),$$

where ((; θ) = Pr(G = k|X = xi ; θ).

While the methods are much simplified in the two-class situation, we go into great depth. The two-class gi may be conveniently coded with a 0/1 response y_i, where y_i= 1 when g_i= 1 and y_i = 0 when g_i = 2. Let p1(x, y) equal p(x, y) and p2(x, y) equal 1 p(x, y). You may write the log-likelihood down.

$$\begin{aligned} \ell(\beta) &= \sum_{i=1}^{N} \Big\{ y_i \log p(x_i; \beta) + (1 - y_i) \log(1 - p(x_i; \beta)) \Big\} \\ &= \sum_{i=1}^{N} \Big\{ y_i \beta^T x_i - \log(1 + e^{\beta^T x_i}) \Big\}. \end{aligned}$$

Here β = {, } and To accommodate the intercept, we assume that the vector of inputs x iincludes the constant term 1.

We set the derivatives of the log-likelihood to zero to maximise it. These are the score equations.

$$\frac{\partial \ell(\beta)}{\partial \beta} = \sum_{i=1}^{N} x_i (y_i - p(x_i; \beta)) = 0,$$

Those are nonlinear p+1 equations in The first scoring equation indicates that because the first component of x_i equals 1,

$$\sum_{i=1}^{N} y_i = \sum_{i=1}^{N} p(x_i; \beta);$$

Notes

The expected number of class one students is the same as the actual amount (and hence also class twos.)

We utilise the Newton–Raphson approach to solve the score equations, which requires the second-derivative or Hessian matrix.

$$\frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^T} = -\sum_{i=1}^{N} x_i {x_i}^T p(x_i;\beta)(1-p(x_i;\beta)).$$

Starting with , a single Newton update is

$$\beta^{\text{new}} = \beta^{\text{old}} - \left(\frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^T}\right)^{-1} \frac{\partial \ell(\beta)}{\partial \beta},$$

Where the derivatives are evaluated at

The best way to write the score and Hessian is in matrix notation. In this example, y will stand for the vector of [y ivalues, X will represent the N (p + 1) matrix of x I values, p wil $\frac{\partial \ell(\beta)}{\partial \beta} = \mathbf{X}^T(\mathbf{y}-\mathbf{p})$ tted probabilities with the ith element p(xi; old) and W will repre of weights with the ith diagonal element. p(;)).(1− p(;)). Ther $\frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^T} = -\mathbf{X}^T\mathbf{W}\mathbf{X}$

The Newton step is thus

$$\begin{aligned}\beta^{\text{new}} &= \beta^{\text{old}} + (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{y}-\mathbf{p}) \\ &= (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}\left(\mathbf{X}\beta^{\text{old}} + \mathbf{W}^{-1}(\mathbf{y}-\mathbf{p})\right) \\ &= (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}\mathbf{z}.\end{aligned}$$

The Newton step has been re-expressed in the second and third lines as a weighted least squares step, with the response

$$\mathbf{z} = \mathbf{X}\beta^{\text{old}} + \mathbf{W}^{-1}(\mathbf{y}-\mathbf{p}),$$

It is sometimes referred to as the compensated response. Since p and therefore W and z, vary with each iteration, these equations must be solved numerous times. This method is called iteratively reweighted least squares, or IRLS since each iteration solves the weighted least squares issue.

$$\beta^{\text{new}} \leftarrow \arg\min_{\beta}(\mathbf{z}-\mathbf{X}\beta)^T\mathbf{W}(\mathbf{z}-\mathbf{X}\beta).$$

While convergence is not guaranteed, initializing β = 0 is often considered an acceptable starting point for the iterative approach. The log-likelihood's concavity typically facilitates convergence, but there might be cases of overshooting. To ensure convergence in such rare instances where the log-likelihood declines, step size halving can be employed. The Newton method can also be described as an iteratively reweighted least squares algorithm, involving a vector of K-1 responses and, in the multiclass case, a non-diagonal weight matrix per observation (K-3). Dealing with the extended vector directly is numerically more convenient, as it eliminates the possibility of simpler approaches.

Utilising logistic regression models for data analysis and inference aims to understand how the input variables contribute to explaining the outcome. Researchers

Notes

often fit multiple models in search of a simple model that incorporates a subset of variables, possibly with interaction terms. The following example illustrates some of the challenges encountered in this process.

Table: Outcomes of fitting a logistic regression model to the data on cardiac disease in South Africa.

	Coefficient	Std. Error	Z Score
(Intercept)	−4.130	0.964	−4.285
sbp	0.006	0.006	1.023
tobacco	0.080	0.026	3.034
ldl	0.185	0.057	3.219
famhist	0.939	0.225	4.178
obesity	-0.035	0.029	−1.187
alcohol	0.001	0.004	0.136
age	0.043	0.010	4.184

Example: South African Heart Disease

We use binary data from the Coronary Risk-Factor Study (CORIS) baseline survey conducted in three rural South African regions to demonstrate the widespread use of logistic regression in statistics. The study's objective was to assess the severity of risk factors for ischemic heart disease in a high-incidence region, focusing on Caucasian men aged 15 to 64 and whether myocardial infarction (MI) was present during the survey (with a cumulative frequency of 5.1%)

In our data set of 302 controls and 160 cases, we applied maximum likelihood to the logistic regression model and obtained Z scores for each coefficient, which are the coefficients divided by their standard errors. A nonsignificant Z score indicates that a coefficient can potentially be eliminated from the model. Although not directly testing the null hypothesis that the coefficient is zero, the Z scores are technically identical to the Wald test. A Z score greater than about 2 in absolute value is considered significant at the 5% significance level.

Upon inspecting the table of coefficients, we encountered some surprising results that require careful interpretation. For example, the difference between systolic and diastolic blood pressure (sbp) was found to be insignificant, as was obesity, which is not a reliable indicator. The reason for this perplexity lies in the correlation between the predictors. Both sbp and obesity are individually important and have a positive sign, but in the presence of other related factors, they may no longer be necessary (and could even show a negative sign). To address this, the analyst may engage in model selection by iteratively eliminating the coefficient with the least significance and re-fitting the model. This process is repeated until no further terms can be eliminated, resulting in a subset of variables that sufficiently explain their combined effect on the prevalence of chd (coronary heart disease).

Figure: A scatterplot matrix is used to display the statistics on heart illness in South Africa. Each figure shows two risk variables, with cases and controls represented by contrasting colours (blue is a case). Family history of heart disease is a binary variable called "famhist" (yes or no).

Notes

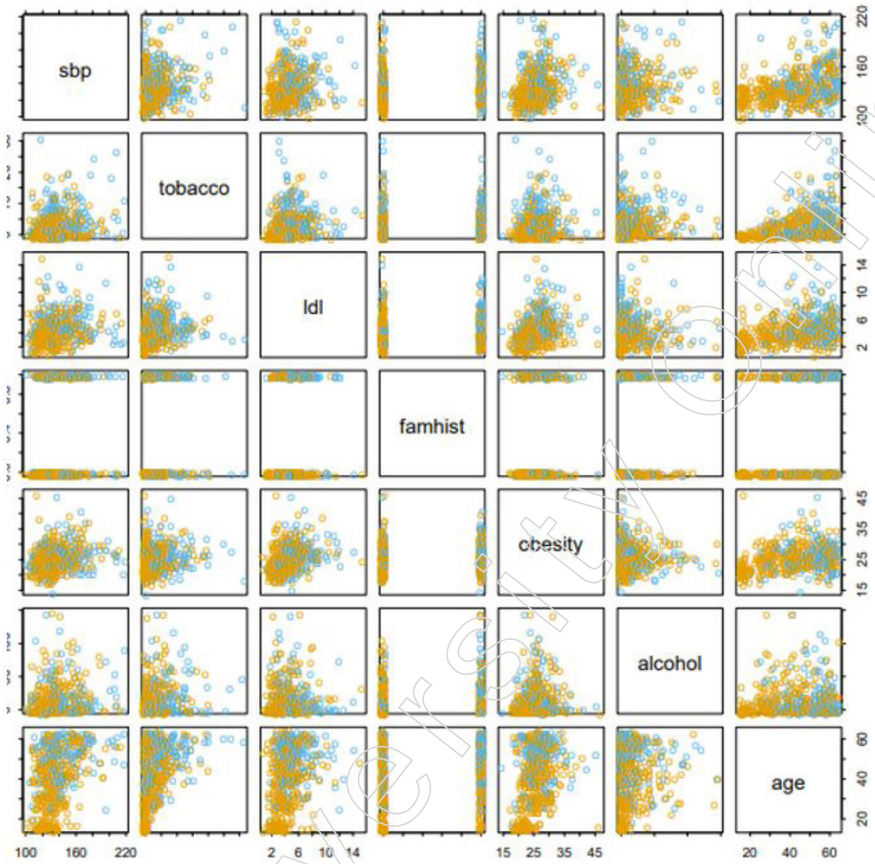


Table: The results of stepwise logistic regression were fitted to the data on cardiac disease in South Africa.

	Coefficient	Std. Error	Z score
(Intercept)	−4.204	0.498	−8.45
tobacco	0.081	0.026	3.16
ldl	0.168	0.054	3.09
famhist	0.924	0.223	4.14
age	0.044	0.010	4.52

Refitting each model with one variable removed and analysing deviance to identify which variable to drop is superior but time-consuming. The difference between two fitted models' residual deviances is their deviance (in analogy to sums-of squares). This method yielded the same model as before.

What does a tobacco coefficient of 0.081 (Standard Error = 0.026) mean? Throughout a lifetime, controls smoke 1.0kg and cases 4.1kg. Hence, lifelong cigarette smoking increases coronary heart disease risk by 8.4% per kilogramme (exp(0.081) = 1.084). The 95 percent confidence interval is exp(0.081 2 0.026) when the standard error is included (1.03, 1.14).

1.2.5 Odds vs Probability

Probability is the measure of the likelihood of an event occurring in a series of trials, expressed as a percentage between 0 and 1. To compute probabilities, we divide the likelihood of the event occurring by the total likelihood of both the event happening and not happening.

Notes

For example, if an event has a Y percent chance of happening, then the probability of it not happening is 1 - Y percent. To clarify, if something is 80% likely to happen, its chance of not happening is 1 - 0.80 = 0.20, or 20%.

In mathematical terms, the probability of an event is the ratio of the probability that the event will occur to the probability that it will not occur.

Odds of event = Y / (1-Y)

In this context, when the probability of an event occurring is 0.80, the odds can be calculated as 0.80 / (1 - 0.80) = 0.80 / 0.20 = 4 (i.e., 4 to 1).

Let's consider some examples to understand the concept of odds and probabilities:

- If a racehorse runs in 100 races and wins 25 while losing 75, the probability of winning is 25/100 = 0.25 or 25% and the odds of winning are 25/75 = 0.333, which can be expressed as one win for every three losses.
- If a horse competes in 100 races, wins 5 of them and loses the remaining 95, the probability of winning is 0.05 or 5% and the odds of winning are 5/95 = 0.0526.
- If a horse competes in 100 races and wins 50 of them, the probability of winning is 50/100 = 0.50 or 50% and the odds of winning are 50/50 = 1 (even odds).
- If a horse competes in 100 races and wins 80 of them, the odds are 80/20 (4 to 1) and the probability of winning is 80/100 (0.80) or 80%.

In case-control studies, where we cannot determine the exact chance of illness in each exposure group, we can use the odds ratio to compare illness risks between different exposure groups. The odds ratio in a hypothetical pesticide study can be calculated as follows:

OR = (7/10) / (5/57) = 6.65

When the outcome being studied is uncommon, the odds ratio closely resembles the risk ratio that would have been obtained if the entire source population had been analysed. This is because the risk of disease in the exposed group will be like the risk of disease in the non-exposed group when the outcome is rare. Therefore, the odds ratio provides an estimation of the risk ratio in the source population and serves as a relative measure of impact in case-control studies, assuming that the outcome is uncommon.

In case-control studies, only the odds ratio can be calculated as a measure of association. On the other hand, in cohort studies that follow exposure groups and evaluate the occurrence of an event, both a risk ratio and an odds ratio can be calculated.

If we randomly name the cells in a contingency table as follows: (the details of the contingency table are not provided in the text).

	Diseased	Non-diseased
Exposed	a	b
Non-exposed	c	d

The odds ratio (OR) is computed by comparing the odds of an event in two groups. To calculate the odds ratio, we use the following formula:

OR = (a/b) / (c/d)

Where 'a' and 'b' are the counts of events in the exposed group and 'c' and 'd' are the counts of events in the unexposed group. To handle odds ratios, which, like risk ratios, do not have a normal distribution, we use a logarithmic transformation to make

Notes

them more normal. This transformation involves taking the natural logarithm of the odds ratio (Ln(OR)).

Creating a confidence interval for an odds ratio involves two steps. First, we generate a confidence interval for Ln(OR) using standard statistical methods. Then, to obtain the upper and lower bounds of the confidence interval for the odds ratio, we take the antilog of the upper and lower bounds of Ln(OR), respectively. This two-stage process ensures a comprehensive and accurate representation of the confidence interval for the odds ratio, which provides a range of values within which the true odds ratio is likely to lie with a certain level of confidence.

Computing the Confidence Interval for an Odds Ratio

Use the formula to get the confidence interval for an odds ratio.

$$Ln(OR) \pm z\sqrt{\frac{1}{a}+\frac{1}{b}+\frac{1}{c}+\frac{1}{d}}$$

- Determine the confidence interval for Ln using the equation (OR).
- Determine the antilog of the result from step 1 to get the confidence interval for OR, which is exp(Lower Limit), exp(Lower Limit), exp(Lower Limit), exp(Lower Limit), etc (Upper Limit).

The odds ratio's confidence interval's null value is 1, indicating that there is no difference. The odds are considered statistically significant if the 95% confidence interval for the odds ratio excludes one. We go back to the earlier examples and estimate odds ratios, which we then compare to our estimates of relative risks and risk differences.

Comparison Chart

Basis For Comparison	Odds	Probability
Meaning	Odds refers to the chances in favor of the event to the chances against it.	Probability refers to the likelihood of occurrence of an event.
Expressed in	Ratio	Percent or decimal
Lies between	0 to ∞	0 to 1
Formula	Occurrence/Non-occurrence	Occurrence/Whole

Definition of Odds

In arithmetic, "odds" refers to the ratio between the total number of favourable occurrences and the total number of unfavourable occurrences. In contrast to odds in favour of an event, which indicate the likelihood that the event will occur, odds against an event indicate the probability that the event will not occur. When statisticians and gamblers discuss statistics, they are referring to the probability of a particular event occurring or not taking place. Odds may vary anywhere from 0 to infinity; if the chances are 0 then the event is very unlikely to take place, however, if the odds are then the event is more likely to take place.

As an example, Let's say there are 20 marbles in a bag, with eight of them being red, six of them being blue and six of them being yellow. In a game of chance where one marble is selected at random, the odds of receiving a red marble are eight out of twelve, or two to three.

Notes

Definition of Probability

Chance is a scientific concept that examines the likelihood that a particular event will occur. It provides the foundation for both estimation and hypothesis testing theories. It is calculated by dividing the total number of occurrences by the number of positive occurrences.

A probability is a value that ranges from 0 to 1, inclusive. When an event's probability is 0, it means it's impossible and when it's 1, it means it's likely or certain to happen. In other words, the chances of an event occurring increase with increasing probability.

Think about the following example: Let's imagine a dartboard with 12 sections—one for each of the 12 zodiac signs—on it. If a dart is aimed towards a certain location, the chances of that area happening are 1/12 since the favorable occasion is 1 (Aries) and there are 12 overall events, which may be expressed as 0.08 or 8% of the total events.

Key Differences Between Odds and Probability

The following points describe the distinctions between odds and probability:

- The word "odds" is used to describe the likelihood that an event will occur. On the other hand, probability determines the likelihood of an event happening, or how often the occurrence will occur.
- Probability is stated as a percentage or a decimal, as opposed to the odds, which are expressed as a ratio.
- Odds normally range from zero to infinity, where zero indicates that there is no chance of an event happening and infinite indicates that there is a chance it might happen. Contrarily, the range of probability is from 0 to 1. So, the possibility of anything happening depends on how near it is to either zero or one; the closer it is to one, the more likely it is to happen.
- Probability is a fundamental concept in mathematics that deals with the likelihood or chance of events happening. It involves calculating the ratio of favorable outcomes to total possible outcomes. This ratio allows us to determine the odds of events occurring.

In probability, we can express these odds using either probability or ratios. The odds represent the relationship between the occurrence and non-occurrence of an event, while probability measures the proportion of times an event happens relative to all possible outcomes. Both probability and odds are valuable tools for understanding and quantifying uncertainty and randomness in various scenarios.

1.2.6 Hypothesis Testing

The theories of statistics and probability play a crucial role in the scientific aspect of data science, helping us develop and test hypotheses about our data and the underlying processes. As data scientists, we often need to assess the probability that a specific hypothesis is true. Hypotheses are statements that can be translated into statistical statements about the data, like "this coin is fair," "data scientists prefer Python to R," or "pop-up ads drive users away from a webpage." These statistical statements are essentially observations of random variables from known distributions under different hypotheses. By analysing these observations, we can infer the likelihood that the hypotheses are true.

In the traditional framework, we have a null hypothesis, H0, which represents the default position or assumption. We then have an alternative hypothesis, H1, which we compare to H0. Through statistical methods, we determine whether we can reject H0 as

Notes

false and accept H1. This process allows us to draw conclusions about the likelihood of certain hypotheses being valid.

To better illustrate this, consider an example: Suppose we want to determine if a coin is fair (H0) or biased (H1). We can flip the coin multiple times and record the outcomes (observations of random variables). By analysing the distribution of these outcomes, we can assess the likelihood of the coin being fair or biased. In essence, statistics and probability provide the tools and framework to evaluate hypotheses, making data science a powerful and evidence-based approach to understanding the world around us.

Example: Flipping a Coin

To test whether a coin is fair, we set up a hypothesis test with a null hypothesis that the coin has a probability of landing heads (p) equal to 0.5. The alternative hypothesis is that p is not equal to 0.5. We perform the test by tossing the coin n times and recording the number of heads, denoted as X. In this scenario, each coin flip is considered a Bernoulli trial, which means X follows a Binomial(n, p) distribution and can be approximated by a normal distribution.

When dealing with a random variable that follows a normal distribution, we can calculate the probability that its actual value falls within a specified interval or outside of it. To determine the non-tail region or symmetric interval around the mean that contains a specific probability level, we find the cutoff points where the upper and lower tails each contain half of the specified probability level, leaving the desired probability level in the center. For example, if we want an interval centered at the mean with 60% probability, we find the cutoffs where the upper and lower tails each contain 20% probability, resulting in a 60% probability interval.

1.2.7 Advantages and Pitfall of Linear Regression

Advantages of Linear Regression

Simple implementation

Linear regression is a straightforward and efficient strategy that can be easily implemented and delivers satisfactory results. Unlike more complex approaches, linear regression models can be trained quickly and effectively, even on systems with limited processing capabilities. In comparison to other machine learning methods, linear regression exhibits significantly lower temporal complexity. The equations involved in linear regression are relatively simple to understand and interpret, making the concept of linear regression easy to grasp.

Performance on linearly separable datasets

It is common practice to make use of linear regression to ascertain the nature of the connection that exists between the many variables in a dataset that may be divided linearly.

Over-fitting can be reduced by regularization

Over-fitting: Overfitting occurs when a machine learning model fits a dataset too closely, including the noisy data present in the training set. Therefore, the model's performance may suffer, leading to reduced accuracy when applied to unseen data in the test set.

Regularization: This simple approach effectively reduces the complexity of a function, thereby minimizing the risk of overfitting the data.

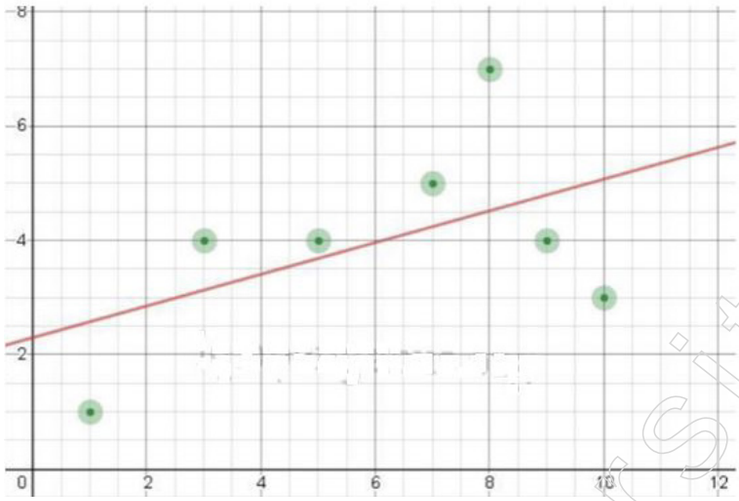
Disadvantages of Linear Regression

Notes

Prone to under-fitting

Under-fitting: It is possible for a scenario to arise where a machine learning model does not sufficiently capture the data. Something takes place whenever the hypothesis function is unable to fit the facts in an adequate manner.

Example

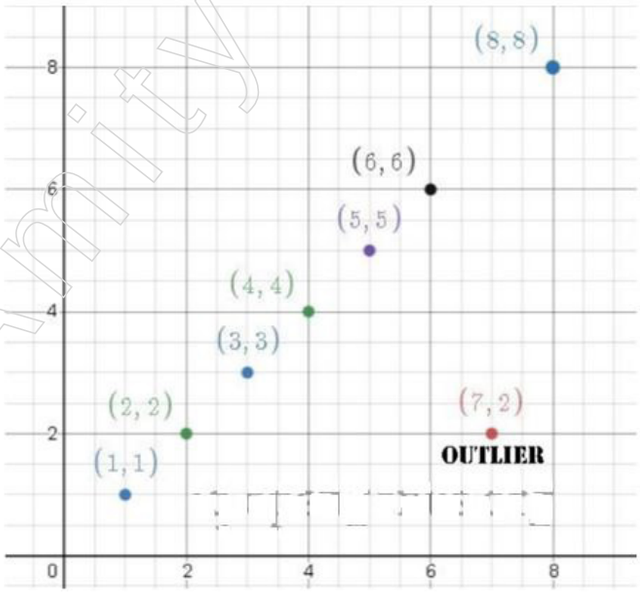


The assumption that the input and output variables have a linear relationship prevents linear regression from fitting complex data sets. Given that the relationships between variables in the dataset are frequently nonlinear, a straight line cannot accurately represent the data. In this situation, a more complex function may capture the data more accurately. As a result, most linear regression models are inaccurate.

Sensitive to outliers

Data outliers are uncommon or extreme values that deviate significantly from the rest of the distribution in a dataset. These outliers can adversely impact the performance of a machine learning model, leading to inaccurate predictions and reduced model efficacy.

Example:



Notes

Because outliers have the potential to have a significant impact on the overall performance of linear regression, appropriate action must be taken prior to employing linear regression to the dataset.

Linear Regression assumes that the data is independent

Prior to employing linear regression, multicollinearity must be eradicated, as the inputs are frequently not independent of one another.

Limitations of regression analysis

Regression analysis is a versatile statistical technique widely applied in natural, physical, and social sciences for various purposes, offering numerous benefits and applications:

- **Rapid Estimation and Prediction:** It allows for quick estimation or prediction of unknown values of one variable based on known values of another, establishing a meaningful relationship between related variables.
- **Error Estimation:** Regression analysis provides a measure for estimating regression line errors. A limited dispersion of observed values around the regression line indicates accurate estimates, while higher dispersion suggests inaccuracies.
- **Correlation Coefficient:** It calculates the correlation coefficient (r) between two variables, indicating the strength and direction of their relationship.
- **Coefficient of Determination:** The coefficient of determination (r^2) depicts the influence of the independent variable on the dependent variable, giving an idea of the predictive ability of the regression analysis.
- **Business Forecasting:** Regression analysis is a powerful tool in business and commerce for forecasting future occurrences like consumption, production, investment, sales, and profits.
- **Causal Relationships:** It helps identify cause-and-effect connections between economic variables, used to estimate demand and supply curves, production functions and more.
- **Sociological Research:** Regression analysis is commonly used in sociological research to estimate birth and death rates, tax rates, yield rates and various other variables.
- **Variance Estimation:** The method provides estimates of relative variance in a series, aiding in understanding data variability.

1.3 Overview of Neural Network and Deep Learning

In the field of artificial intelligence and machine learning, neural networks and deep learning are effective algorithms. They are designed to process and analyses complex data and are inspired by how the human brain operates.

Neural networks are comprised of layered, interconnected structures, or neurons. Each neuron receives input, processes it with an activation function and generates output that is transmitted to the next layer. During training, the weights of the connections between neurons are modified to learn patterns in the data.

Notes

Deep learning is a subfield of machine learning concerned with the training of neural networks with multiple hidden layers. These deep neural networks can autonomously discover hierarchical data representations, enabling them to make more precise predictions.

There are numerous neural network varieties, each designed for specific tasks:

- **Feedforward Neural Networks:** Without feedback connections, data flows from input to output layers in one direction.
- **Recurrent Neural Networks (RNNs):** They contain loops that retain information from previous inputs, making them suitable for sequential data.
- **Convolutional Neural Networks (CNNs):** Designed for image recognition tasks, these neural networks use convolutional and pooling layers to discover hierarchical image features.

Deep learning models are capable of both feature learning and representation learning, which means they can automatically identify relevant patterns and construct multiple levels of abstraction from raw data, thereby eradicating the need for manual feature engineering. Neural networks and deep learning have revolutionised several applications, such as image and speech recognition, natural language processing and game playing, and they continue to advance the capabilities of artificial intelligence.

1.3.1 Introduction to Neural Network

A neural network is calm of series stacked layers. Each layer consists of units that are connected to the units of the previous layer via a set of weights. As we shall see, there are types of layers, but one of the most common is the completely linked (or dense) layer, which connects all units in the layer to every unit in the previous layer layer.

Neural networks where all adjacent layers are fully connected are called *multilayer perceptrons* (MLPs). This is the first type of neural network that we will study. An example of an MLP is shown in Figure.

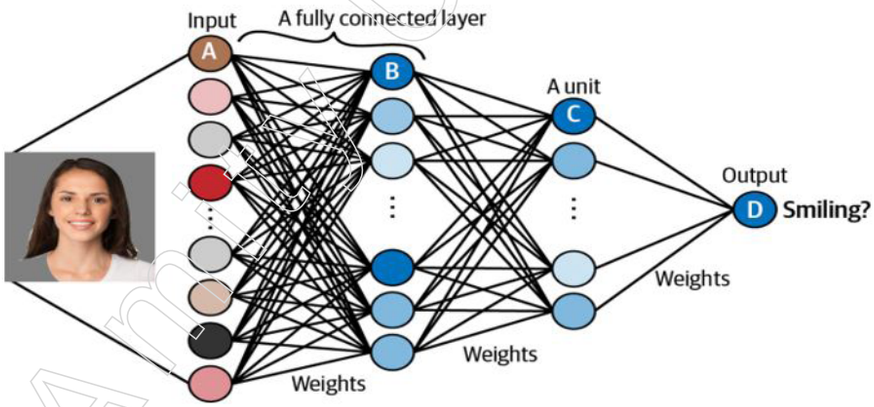


Figure. An example of a multilayer perceptron that predicts if a face is smiling

The input (e.g., an image) is transformed by each layer in turn, in what is known as a *forward pass* through the network, until it reaches the output layer. Specifically, each unit applies a nonlinear transformation to a weighted sum of its inputs and passes the output through to the subsequent layer. The final output layer is the culmination of this process, where the single unit outputs a probability that the original input belongs to a particular category (e.g., *smiling*).

Notes

Finding the set of prediction weights for each layer that yields the most accurate predictions is the key to the success of deep neural networks. What we mean by "training the network" is the process of finding these weights.

Throughout the training process, collections of images are fed through the network and the predicted outputs are compared to the ground truth. For instance, the network may output a probability of 80% for an image of a person who is actually smiling and a probability of 23% for an image of a person who is not actually smiling. There is a small measure of error, as a precise prediction would yield 100% and 0% for these instances.

After making a prediction, the error in the prediction is calculated. This error is then propagated backward through the neural network and for each set of weights, a small adjustment is made in the direction that improves the prediction the most. This process is repeated iteratively, fine-tuning the weights at each step, to gradually improve the overall accuracy of the network's predictions. This process is referred to as backpropagation. Gradually, each unit becomes adept at recognising a specific feature, which ultimately enables the network to make more accurate predictions.

1.3.2 Applications of Neural Network

Nature has often inspired human inventions, such as flight inspired by birds and velcro by burdock plants. Similarly, the idea of creating sentient machines led to the development of artificial neural networks (ANNs), drawing inspiration from the brain's structure. However, over time, ANNs have evolved and diverged significantly from their organic counterparts. Some researchers propose moving away from the biological analogy in order to embrace more diverse and creative systems. For instance, they suggest using the term "units" instead of "neurons" to avoid being constrained by biological plausibility.

Artificial Neural Networks (ANNs) serve as the foundation of Deep Learning and have demonstrated remarkable versatility, power, and scalability. These attributes make them highly suitable for tackling large and complex Machine Learning tasks. For example, ANNs excel at classifying billions of images in Google Images, enabling speech recognition services in Apple's Siri, providing personalized video recommendations to millions of users on YouTube and even defeating world champions in games like Go, as showcased by DeepMind's AlphaGo. The adaptability and capabilities of ANNs have propelled the field of Deep Learning to achieve extraordinary accomplishments across various applications.

1.3.3 Introduction to Deep Learning

To comprehend deep learning and differentiate it from other machine learning approaches, it is essential to understand the role of machine learning algorithms. Machine learning algorithms are designed to learn patterns and rules from data in order to perform specific tasks. For successful machine learning, three key components are required:

- **Input data points:** These are the data instances used as input for the learning process. For instance, in speech recognition, the input data points could be sound files of people speaking, while in image annotation, photographs serve as the input objects.
- **Examples of the expected output:** The machine learning algorithm needs to be provided with labelled data that represents the expected outcomes. In speech recognition, these examples could be human-created transcripts of the sound files and in image tasks, the expected outputs could include labels like "dog," "cat," and so on.

Notes

- **A performance evaluation metric:** To assess the algorithm's effectiveness, a way to measure its performance is essential. In speech recognition, this could involve comparing the algorithm's transcriptions to human-created transcripts, while in image tasks, the evaluation metric may involve comparing the algorithm's predicted labels to the correct labels.

Machine learning models operate by transforming input data into meaningful outputs, a process learned from examples of known inputs and outputs. A key challenge in both machine learning and deep learning is to discover valuable representations of input data that lead to the desired outputs.

Deep learning, a subfield of machine learning, specifically focuses on progressively learning more meaningful representations through a series of layers. The term "deep" in deep learning refers to these layered representations rather than an advanced level of understanding. Typically, deep learning models consist of tens or hundreds of successive layers, automatically learned from training data. In contrast, other machine learning approaches may only learn one or two layers of representations, often referred to as shallow learning.

Deep learning achieves these layered representations using neural networks, which are models composed of stacked layers. While the term "neural network" is inspired by neurobiology, it is crucial to understand that deep learning models are not brain models. They do not operate like the brain and there is no evidence suggesting that the brain employs the same learning mechanisms as modern deep learning models. Any suggestion of direct connection between deep learning and neurobiology should be avoided as it can be misleading. Instead, deep learning should be recognized as a mathematical framework for learning representations from data.

How do a deep learning algorithm's learned representations appear?

Consider how a network with multiple layers transforms an image of a digit in order to recognise it (see Figure 1.5).

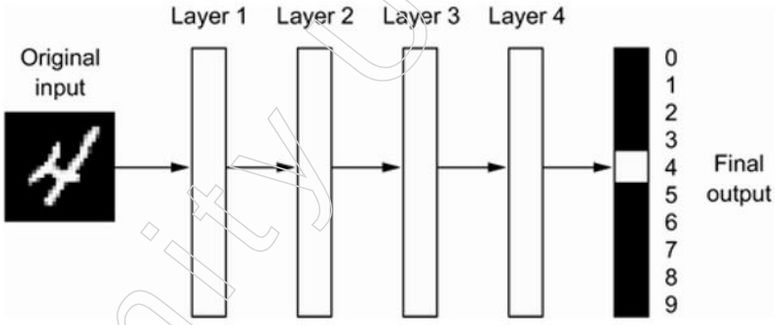


Figure : deep neural network for digit classification

As shown in the figure, the network turns the digital image into representations that are increasingly distinct from the original image and increasingly informative about the ultimate result. A deep network can be conceptualised as a multistage information-distillation process in which information passes through successive filters and emerges increasingly refined (i.e., useful for some task).

Notes

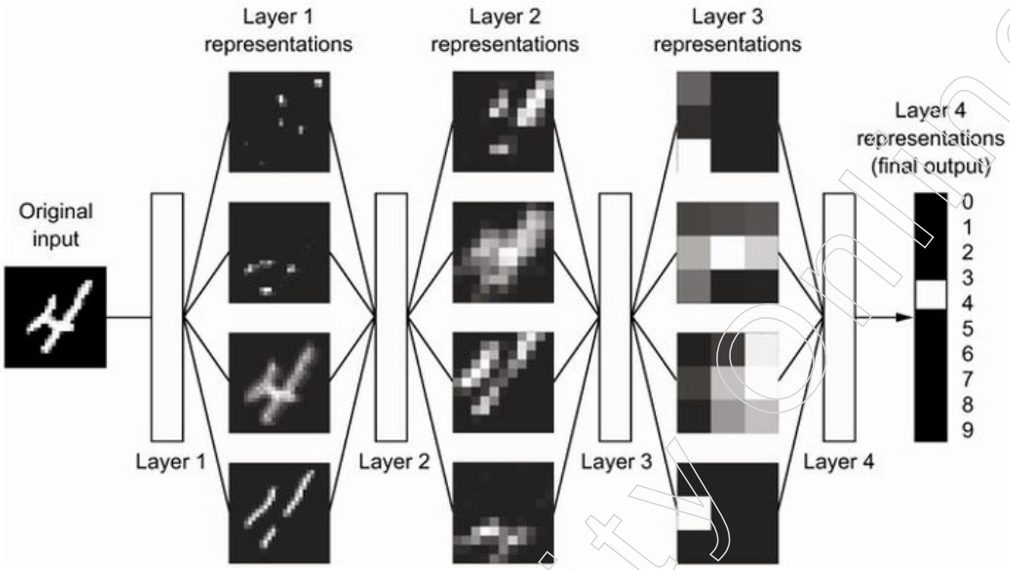


Figure: Data representations learned by a digit-classification model

Deep learning involves a multi-step process of learning data representations. While the concept may seem straightforward, the magic lies in the fact that simple mechanisms can appear almost mystical when skilfully scaled up.

1.3.4 Applications of Deep Learning

Deep learning can be applied to structured data, but its true strength, particularly in terms of generative modelling, comes from its ability to work with unstructured data. Typically, we want to generate unstructured data such as new images or original text sequences, which is why deep learning has had such a significant impact on the field of generative modelling.

Deep learning, though an older subfield of machine learning, truly gained widespread recognition and importance in the early 2010s. Since then, it has sparked a revolutionary transformation in the field, achieving remarkable success in challenging areas like perceptual tasks and natural language processing. These tasks involve skills that have always been effortless for humans but historically posed immense difficulties for machines. Nonetheless, deep learning has overcome these barriers and achieved impressive results.

Deep learning has been instrumental in driving breakthroughs in historically challenging areas of machine learning, leading to remarkable technological advancements.

- Near-human-level image classification
- Near-human-level speech transcription
- Near-human-level handwriting transcription
- Dramatically improved machine translation
- Dramatically improved text-to-speech conversion
- Digital assistants such as Google Assistant and Amazon Alexa
- Near-human-level autonomous driving

Notes

- Improved ad targeting, as used by Google, Baidu, or Bing
- Improved search results on the web
- Ability to answer natural language questions
- Superhuman Go playing

The potential of deep learning continues to be explored and has achieved remarkable feats in recent years. Tasks once deemed impossible, like automatically transcribing ancient manuscripts, detecting plant diseases with a smartphone, and aiding in medical image interpretation, are now becoming realities. This progress extends to various fields, from science and medicine to manufacturing, energy, transportation, agriculture, and the arts.

While deep learning has accomplished extraordinary things, some expectations for its future capabilities may be overly optimistic. Although we have made significant strides, certain goals, like creating truly human-level dialogue systems or language comprehension, may remain out of reach for some time. It's important to be cautious with claims of achieving general intelligence comparable to humans.

Unrealistically high expectations can lead to disappointment if technology falls short, resulting in reduced research funding and slowing down progress. While some groundbreaking applications like autonomous vehicles are on the horizon, it's essential to maintain a balanced perspective on what deep learning can achieve in the near future. This way, we can continue making advancements without discouraging or delaying further development.

1.3.5 Neural Network vs. Deep Learning

Artificial Intelligence (AI) serves as the broadest category of intelligent systems. Within AI, we find Machine Learning, a subset that focuses on algorithms learning from data. Deep Learning, a subfield of Machine Learning, utilises neural networks as its foundation. A neural network is a collection of artificial neurons organised in layers: input, hidden and output.

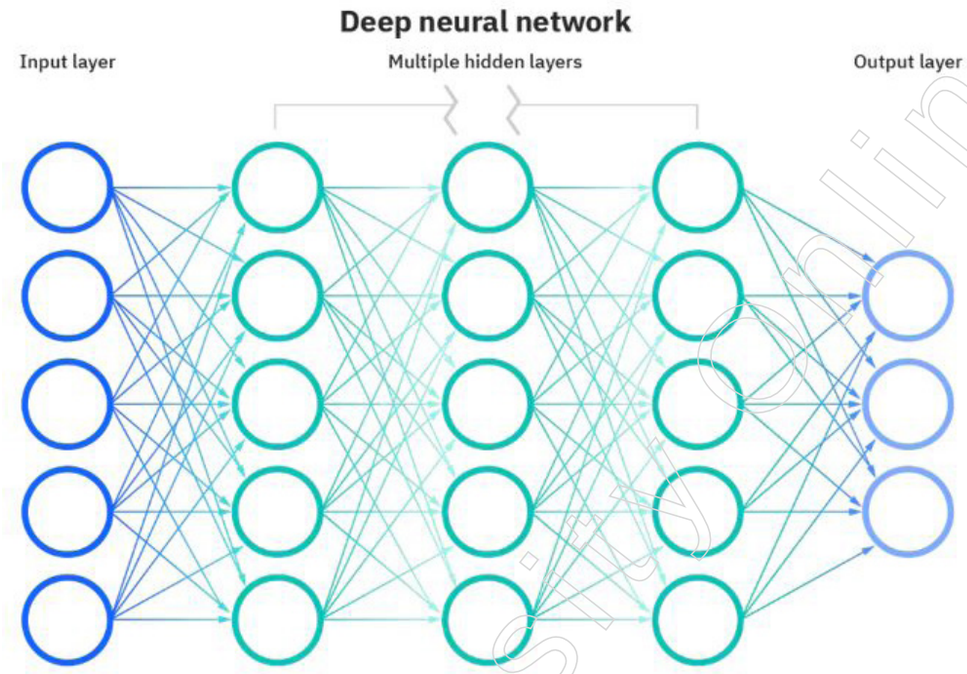
These artificial neurons are like brain cells, communicating with each other through connections with specific weights and threshold values. When a neuron's output surpasses the threshold, it activates and passes its information to the next layer. Otherwise, no transmission occurs.

Neural networks learn from training data, improving their accuracy over time. Once refined, they become potent tools in computer science and AI, quickly handling tasks like data classification and aggregation. For instance, speech and image recognition tasks that would take hours manually can now be completed within minutes using neural networks. Google's search algorithm is a well-known application of such a network.

How are neural networks and deep learning dissimilar?

"Deep" refers to the number of layers in a neural network. This point was made in the explication of neural networks, but it is important to reiterate it. A neural network with more than three layers, including inputs and outputs, can be used to define a deep learning algorithm. This is represented by the following diagram:

Notes



Most deep neural networks work in a one-way manner, where data flows from the input to the output. Nevertheless, there's a powerful training technique called back-propagation that operates in the opposite direction, moving from the input towards the output. Through back-propagation, we can identify and measure the errors made by individual neurons, which empowers us to fine-tune and improve the algorithm accordingly.

Summary

- Artificial intelligence (AI) has been a subject of fascination and debate since the early days of programmable computers, with questions about the possibility of computers thinking like humans.
- AI is a broad term, with subsets like Machine Learning (ML) and Deep Learning (DL), which involve computers learning from data without explicit programming.
- The history of AI dates back to ancient times, with significant milestones in the 20th century, leading to the establishment of AI as a field in 1956.
- Alan Turing's work during World War II and his concept of "machine intelligence" laid the groundwork for AI research.
- There are three types of AI: Artificial Narrow Intelligence (ANI), Artificial General Intelligence (AGI) and Artificial Superintelligence (ASI).
- Machine Learning is a versatile and powerful technology that enables computers to learn from data and experiences. A wide range of machine-learning algorithms and methodologies have been developed to facilitate this learning process. It is built upon the foundations of computer science and statistics, making it a valuable tool for various applications.

Glossary

- **Artificial Intelligence (AI):** The broad field of creating intelligent machines and systems capable of performing tasks that typically require human intelligence.
- **Machine Learning (ML):** A subset of AI where computers learn from data without being explicitly programmed, enabling them to improve their performance on a task

Notes

through experience.

- **Deep Learning (DL):** A subfield of machine learning that involves neural networks with multiple hidden layers, enabling the learning of more abstract and intricate data representations.
- **Alan Turing:** An influential mathematician and computer scientist known for his work during World War II and his concept of "machine intelligence," which laid the groundwork for AI research.
- **Artificial Narrow Intelligence (ANI):** AI that is specialized and proficient in performing a specific task, but lacks general cognitive abilities.
- **Artificial General Intelligence (AGI):** AI with human-like cognitive abilities, capable of understanding, learning, and reasoning across various tasks.
- **Artificial Superintelligence (ASI):** AI that surpasses human intelligence and may have far-reaching implications, both positive and negative.
- **DeepMind:** A prominent AI research organisation focused on creating AGI-like neural networks and applying AI technology to various fields.
- **Supervised Machine Learning:** A type of ML where the model learns from labelled training data, associating input data with corresponding output labels.
- **Linear Regression:** A widely used supervised ML model for predictive modelling, assuming a linear relationship between independent and dependent variables.
- **Residuals:** The vertical distance between data points and the fitted line in linear regression, representing the prediction errors.
- **Coefficients:** Parameters of the linear equation in linear regression, determining the slope and intercept of the fitted line.
- **Heteroscedasticity:** A condition in linear regression where the variance of errors varies across different levels of the independent variable.
- **Huber Loss:** A robust loss function in linear regression that is less sensitive to outliers compared to mean squared error and mean absolute error.

Check Your Understanding

1. What is the primary focus of DeepMind in AI research?
   a) Creating robots with human-like cognitive abilities
   b) Replicating human-like cognitive abilities in neural networks
   c) Developing advanced natural language processing algorithms
   d) Applying AI technology to space exploration
2. Which type of AI aims to surpass human intelligence?
   a) Artificial Narrow Intelligence (ANI) b) Artificial General Intelligence (AGI)
   c) Artificial Superintelligence (ASI) d) Artificial Emotional Intelligence (AEI)
3. Which AI subset involves computers learning from data without explicit programming?
   a) Robotics b) Vision
   c) Natural Language Processing (NLP) d) Machine Learning (ML)
4. When was the field of AI officially established?
   a) 1945 b) 1956
   c) 1965 d) 1976

Notes

5. What did Alan Turing's work during World War II lay the groundwork for?
   a) Deep Learning (DL) b) Natural Language Processing (NLP)
   c) Artificial General Intelligence (AGI) d) AI research
6. Which technology enables computers to learn from data and experiences
   a) Robotics b) Deep Learning (DL)
   c) Artificial Intelligence (AI) d) Machine Learning (ML)
7. In the context of AI, what does AGI aim to replicate?
   a) Human-like cognitive abilities
   b) Advanced speech recognition capabilities
   c) Vision processing algorithms d) Robotic movements
8. What is the foundation of Machine Learning (ML)?
   a) Biology b) Chemistry
   c) Computer Science and Statistics d) Physics
9. Linear regression is primarily used for:
   a) Unsupervised machine learning
   b) Predictive modelling with a linear relationship
   c) Reinforcement learning d) Image classification
10. What type of models are commonly used to approximate discrete multidimensional probability distributions?
   a) Linear regression models Log-linear models
   b) Log-linear models
   c) Neural networks d) Decision trees
11. Multiple linear regression allows for the consideration of:
   a) One predictor variable b) Two predictor variables only
   c) Three or more predictor variables d) Only dependent variable
12. What is a potential limitation of linear regression?
   a) It can handle non-linear relationships between variables
   b) It requires no assumptions about the data
   c) It can predict categorical outcomes
   d) It assumes a linear relationship between variables
13. What does R-squared measure in linear regression?
   a) The number of predictor variables in the model
   b) How well the model explains variation in the dependent variable
   c) The average of absolute errors in the model
   d) The correlation between independent and dependent variables
14. Which condition refers to varying residual variance across different levels of the independent variable in linear regression?
   a) Heteroscedasticity b) Homoscedasticity

Notes

c) Multicollinearity d) Normality of errors

15. What is the inspiration behind the design of neural networks?
    a) Humanoid robots b) The functioning of the human brain
    c) Ancient computer architectures d) Quantum mechanics
16. How are neurons organised in a neural network?
    a) Randomly connected b) In parallel processing units
    c) Sequentially without layers d) Interconnected in layers
17. What is the process through which neural networks learn from data?
    a) Training b) Testing
    c) Prediction d) Feature engineering
18. What is one of the key strengths of neural networks?
    a) Their ability to perform complex mathematical calculations
    b) Their capacity to store vast amounts of data
    c) Their automatic learning of meaningful representations from raw data
    d) Their capability to mimic human emotions
19. Which type of data are neural networks particularly well-suited for?
    a) One-dimensional data b) Low-dimensional data
    c) High-dimensional data d) Textual data
20. What is the primary focus of deep learning in machine learning?
    a) Reinforcement learning b) Linear regression
    c) Neural networks with multiple hidden layers
    d) Unsupervised learning

Exercise

1. Describe the historical development of Artificial Intelligence (AI) and its milestones from ancient times to the establishment of AI as a field in 1956.
2. Define Artificial Intelligence (AI) and Machine Learning (ML) and explain the key differences between these two concepts.
3. What is Linear Regression and how does it work as a supervised machine learning model for predictive modelling?
4. Explain the statistical view of regression, focusing on how Linear Regression approximates the relationship between independent variables and the dependent variable.
5. Describe their composition with interconnected neurons arranged in layers and their ability to learn from data through training.
6. Explain the advantages and limitations of Linear Regression as a widely used supervised machine learning model for predictive modelling.

Learning Activities

- Mind Map Creation:

  Divide students into groups and assign each group one of the topics: “Machine

Notes

Learning" or "Artificial Intelligence." Instruct them to create a mind map representing the key concepts, components and applications related to their assigned topic. After completing the mind maps, each group can present their work to the class, promoting discussion and knowledge sharing.

- AI and ML Timeline:

  Ask students to research and create a timeline of significant milestones and events in the history of Artificial Intelligence and Machine Learning. They can use posters, digital tools, or presentations to display the timeline. This activity will help students gain a historical perspective on the development of AI and ML.

Check Your Understanding- Answers

1. b)	2. c)
3. d)	4. b)
5. d)	6. d)
7. a)	8. c)
9. b)	10. b)
11. c)	12. d)
13. b)	14. a)
15. b)	16. d)
17. a)	18. c)
19. c)	20. c)

Further Readings and Bibliography

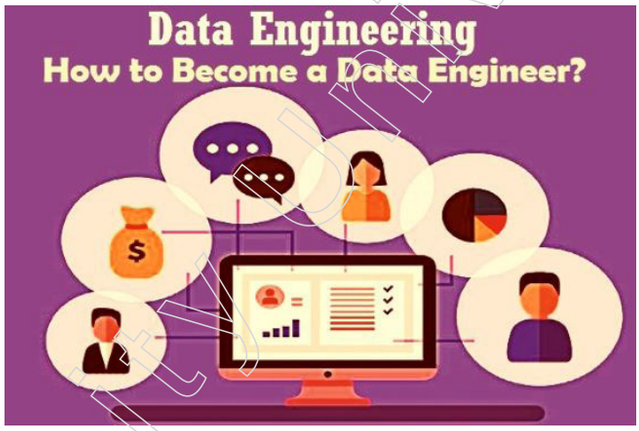
1. "Machine Learning" by Tom Mitchell, McGraw-Hill Education, 1st Edition.
2. "Artificial Intelligence: A Modern Approach" by Stuart Russell and Peter Norvig, Pearson, 3rd Edition.
3. "Deep Learning" by Ian Goodfellow, Yoshua Bengio and Aaron Courville, Publication: The MIT Press, 1st Edition.
4. "Pattern Recognition and Machine Learning" by Christopher M. Bishop, Springer, 1st Edition.
5. "Natural Language Processing with Python" by Steven Bird, Ewan Klein and Edward Loper, O'Reilly Media, 1st Edition.

Module - II: Data Engineering Basics for Everyone

Learning Objectives

At the end of this module, you will be able to:

- Learn introduction to data engineering
- Know overview of data engineering
- Summarise data engineering ecosystem
- Describe sources of gathering data
- Learn data engineering life cycle
- Know architecting the data platform
- Summarise introduction to data wrangling
- Know overview of RDBMS and No SQL
- Learn introduction to RDBMS
- Summarise tools of RDBMS
- Describe role of RDBMS in data engineering
- Learn overview of NoSQL database
- Know applications of NoSQL database
- Summarise important commands in NoSQL
- Describe insight of various database tools



Introduction

If you work in data or software, you may have noticed that data engineering is emerging from the shadows and now shares the stage with data science. Data technology is one of the hottest areas in data technology and for good reason. It is the foundation for data science and analytics in manufacturing. This module explores what data engineering is, how the field was born, its development, data engineer skills and who they work with.

2.1 Introduction to Data Engineering

Despite the current popularity of data engineering, there is a lot of confusion about what data engineering means and what data engineers do. Data engineering has been

Notes

around in some form since companies started doing things with data—such as predictive analytics, descriptive analytics, and reporting—and came to sharper focus alongside the rise of data science in the 2010s. For the purposes of this course/modules, it refers to data technology and data engineer. It is important to define what it is.

2.1.1 Overview of Data Engineering

Let us first look at how data engineering is defined and develop some terms that can be used in these modules. There are endless definitions of data technology. At the beginning of 2022, "what is data engineering?" It offers over 91,000 unique features. Before providing our definition, here are a few examples of how some experts in the field define data engineering.

> ***Data engineering is a set of operations aimed at creating interfaces and mechanisms for the flow and access of information. It takes dedicated specialists—data engineers—to maintain data so that it remains available and usable by others. In short, data engineers set up and operate the organisation's data infrastructure, preparing it for further analysis by data analysts and scientists.***
>
> *From "Data Engineering and Its Main Concepts" by AlexSoft*

> ***The first type of data engineering is SQL-focused. The work and primary storage of the data is in relational databases. All of the data processing is done with SQL or a SQL-based language. Sometimes, this data processing is done with an ETL tool. The second type of data engineering is Big Data–focused. The work and primary storage of the data is in Big Data technologies like Hadoop, Cassandra and HBase. All of the data processing is done in Big Data frameworks like MapReduce, Spark and Flink. While SQL is used, the primary processing is done with programming languages like Java, Scala, and Python.***
>
> *Jesse Anderson*

> ***In relation to previously existing roles, the data engineering field could be thought of as a superset of business intelligence and data warehousing that brings more elements from software engineering. This discipline also integrates specialization around the operation of so-called "big data" distributed systems, along with concepts around the extended Hadoop ecosystem, stream processing and in computation at scale.***
>
> *Maxime Beauchemin*

> ***Data engineering is all about the movement, manipulation, and management of data.***
> *Lewis Gavin*

If you are confused about data engineering, this makes perfect sense. That is just a few definitions and it has a great idea of what data engineering means.

Data Engineering Defined

Unravelling the common threads of how different people define data technology, a clear pattern emerges: a data engineer acquires data, stores it, and prepares it for data scientists, analysts, etc. to consume role We define data technology and data engineers as follows .

Data engineering is the process and process of taking raw data and creating high-quality, consistent information that supports downstream use cases, such as analytics and machine learning data engineering security, data management, DataOps, data architecture, . orchestra and software engineering There is an extension. A data engineer is responsible for the data engineering lifecycle, starting with acquiring data from source systems and ending with serving the data for information use such as analytics or machine learning

Notes

2.1.2 Data Engineering Ecosystem

There are different elements of data engineering ecosystems.

Consumers are at the heart of this ecosystem: they consume product offerings and reap the benefits of an ever-evolving user experience. They are the gold mine of this world and their interactions with various digital products result in raw data running through the ecosystem.

Product Managers and Business Prospects are tasked with continually creating value for both consumers and the product/organisation they work for. They work together to meet consumer expectations, enhance your product's value proposition while optimizing the return on investment of your efforts and achieving business goals.

Data architects work with product managers to create a foundation for what kind of raw data to generate and how to use it. They provide a technical roadmap for the vision of a data ecosystem and this roadmap facilitates the development of data processing pipelines, business intelligence dashboards, data sources, etc.

Data Engineers build on this foundation by configuring resources and creating systems to store and process data and support the analytical needs of their stakeholders. His work revolves around building data pipelines and delivering data in a variety of ways.

Quality Assurance (QA) engineers play a key role in overseeing the quality of work products created by data engineers and ensuring that stakeholder expectations are successfully met. Data analysts at play an interesting role because they work with the unknown. They answer questions that come to mind for every member of the organisation. They help understand unique user behaviour, make market forecasts, troubleshoot issues and anomalies and more. You can work with technical, product, or business teams and work on a variety of issues.

An Analyst is the wrench in your toolbox that can be adjusted to drive varied sizes of nuts and bolts and adds value beyond that.

Data Scientists: You bear the burden of pulling a rabbit out of a hat. His job is to gain deeper insights through data analysis and the use of data science and advanced analytical techniques. From predictive modelling to building recommender systems, his work remains unstructured but extremely valuable.

These technical minds come together to create amazing analytical work products that inform the company about its consumer behaviour and help stakeholders make better decisions every day.

2.1.3 Sources of Gathering Data

When learning about the various underlying operating patterns of the systems that generate data, it is important to understand how data is created. Data is a disorganised and contextless collection of facts. It can be created in many ways, both analogue and digital. The production of analogue data takes place in the real world, for example when speaking, signing, writing on paper, or playing an instrument. This analogue data is usually transient; How many times have you had an oral conversation, the content of which is lost in the ether after the conversation ends?

Digital data results from the conversion of analogue data into a digital format or is the native product of a digital system. An example of analogue to digital is a mobile text messaging application that converts analogue voice to digital text. An example of digital data creation is a credit card transaction on an e-commerce platform. A customer places

Notes

an order, the transaction is charged to their credit card and the transaction information is stored in various databases.

Source Systems: Main Ideas

Source systems generate data in diverse ways. This section discusses the main ideas that you often come across when working with source systems.

Files and Unstructured Data

A file is a sequence of bytes that is typically stored on disk. Applications often write data to files. Files can store local parameters, events, logs, images, and audio.

In addition, files are a universal medium of data exchange. As much as data engineers wish they could retrieve data programmatically, much of the world still sends and receives files. For example, if you receive data from a government agency, there is a good chance you will download the data as an Excel or CSV file or receive the file via email. The main types of source file formats you will encounter as a data engineer (files obtained manually or as a result of a source system process) are Excel, CSV, TXT, JSON, and XML. These files have their peculiarities and can be structured (Excel, CSV), semi-structured (JSON, XML, CSV) or unstructured (TXT, CSV).

APIs

Application Programming Interfaces (APIs) are a standard way of exchanging data between systems. In theory, APIs simplify the task of data collection for data engineers. In practice, many APIs still represent a large amount of data complexity for engineers to manage. Despite the emergence of numerous services and frameworks, as well as services to automate API data ingestion, data engineers often have to invest a lot of energy to maintain custom API connections. We will go into more detail about APIs later in this module.

Application Databases (OLTP Systems)

An application database stores the state of an application. A common example is a database that stores bank account balances. As customer transactions and payments occur, the app updates the account balance.

Typically, an application database is an online transaction processing system (OLTP), a database that reads and writes single records at high speed. OLTP systems are often referred to as transactional databases, but that does not necessarily mean that the system in question supports atomic transactions.

Logs

A log captures information about events that occur on systems. For example, a log can capture usage and traffic patterns on a web server. For example, your desktop operating system “Windows, macOS, Linux” logs events at system startup and when applications start or crash.

Datasets are a rich source of data that is potentially valuable for downstream data analysis, machine learning and automation. Here are some known log sources:

- Operating systems
- Applications
- Servers

Notes

- Containers
- Networks
- IoT devices

Source System Practical Details

This section discusses the practical details of interfacing with modern source systems. We will delve into the details of the databases, APIs, and other more general aspects. This information will have a shorter lifespan than the main ideas discussed above; Popular API frameworks, databases and other details will continue to change rapidly.

However, these details are essential knowledge for working data engineers. We encourage you to study this information for basic knowledge but read it extensively to keep up to date with ongoing developments.

Databases

In this section we look at common source system database technologies you will encounter as a data engineer and the general considerations for working with these systems. There are as many types of databases as there are data use cases.

APIs

APIs are now a standard and widely used way to exchange data in the cloud, for SaaS platforms and between internal enterprise systems. There are many types of API interfaces on the web, but we are interested in those based on HTTP, the most popular type on the web and in the cloud.

REST

First, let us talk about REST, the currently dominant API paradigm. REST stands for Representative State Transfer. This set of practices and philosophies for building HTTP web APIs was presented by Roy Fielding in a 2000 PhD thesis. REST is based on HTTP verbs like GET and PUT; In practice, modern REST uses only a handful of the verb mappings described in the original dissertation.

One of the main ideas of REST is that interactions are stateless. Unlike a Linux terminal session, there is no notion of a session with associated state variables, such as a working directory; Each REST call is independent. REST calls can change the state of the system, but these changes are global and apply to the entire system and not to a current session. Critics point out that REST is by no means a complete specification.5 REST specifies the basic characteristics of interactions, but developers using an API must acquire a significant level of domain knowledge in order to build applications or extract data effectively .

We see big differences in API abstraction levels. In some cases, APIs are just a thin shell over internal components that provide the minimum functionality needed to protect the system from user requests. In other examples, a REST data API is a technical feat that preps data for analytics applications and supports advanced reporting.

Some developments have made it easier to set up data ingestion pipelines via REST APIs. First, data providers often provide client libraries in multiple languages, most notably Python. Client libraries do much of the repetitive work of creating API interaction code for you. Client libraries handle vital details like authentication and assign basic methods to the exposed classes.

Notes

Second, a number of open-source libraries and services have emerged that interact with APIs and manage data synchronization. Many open source and SaaS providers offer out-of-the-box connectors to popular APIs. The platforms also simplify the process of creating custom connectors as needed.

There are numerous data APIs with no client libraries or standard plug-in support. As we emphasise throughout the modules, engineers would do well to reduce the indiscriminate lifting of heavy loads through the use of standard tools. However, simple plumbing jobs still consume a lot of resources. In every large enterprise, data engineers face the problem of writing and maintaining custom code to extract data from APIs. This requires understanding the structure of the provided data, developing appropriate data extraction code, and determining appropriate code. data synchronization strategy.

GraphQL

GraphQL was developed at Facebook as a query language for application data and an alternative to generic REST APIs. While REST APIs limit your queries to a specific data model, GraphQL opens up the possibility of retrieving multiple datalimitngle request. This allows for more flexible and meaningful queries than with REST. GraphQL is based on JSON and returns data in a manner similar to a JSON query. There is something of a holy war going on between REST and GraphQL, with some engineering teams favouring one or the other and others using both. In reality, engineers will encounter both when interacting with source systems.

Webhooks

Webhooks are a simple event-based data transmission pattern. The data source can be an application backend, a web page, or a mobile application. When certain events occur in the originating system, a call to an HTTP endpoint hosted by the data consumer is triggered. Note that the connection is from the source system to the data sink, unlike typical APIs. For this reason, webhooks are often referred to as reverse APIs.

The endpoint can do a number of things with the data in the POST event and trigger further processing or store the data for future use. For analysis purposes, we are interested in collecting these events. Engineers often use message queues to ingest data at high speed and volume. We will talk about message queues and event sequences later in this module.

RPC and gRPC

A remote procedure call (RPC) is commonly used in distributed computing. Allows you to run a procedure on a remote system.

gRPC is a remote procedure call library developed internally at Google in 2015 and later released as an open standard. Just using it on Google would be enough to warrant inclusion in our discussion. Many Google services like Google Ads and GCP offer gRPC APIs. gRPC is based on the open data serialization standard Protocol Buffers, also developed by Google. gRPC emphasises efficient bidirectional data exchange over HTTP/2.Efficiency refers to things like CPU utilisation, power consumption, battery life and bandwidth. Like GraphQL, gRPC enforces much more specific technical standards than REST, allowing the use of common client libraries and allowing engineers to develop skills applicable to any gRPC interaction code.

Third-Party Data Sources

The consumerization of technology means that every company today is a technology company. As a result, these companies and increasingly government agencies, want their

Notes

data to be available to their customers and users, either as part of their service or as a separate subscription. For example, the US Bureau of Labor Statistics publishes various statistics on the US labour market. The National Aeronautics and Space Administration (NASA) publishes various data from their research initiatives. Facebook shares data with companies that advertise on its platform.

Why would companies want their data to be available? The data sticks and creates a flyer by allowing users to integrate and extend your app into a user's app. Higher user adoption and usage means more data, which means users can integrate more data into their applications and data systems. The side effect is that there are now almost endless sources of third-party data.

Direct access to third-party data is typically done through APIs, exchanging data on a cloud platform, or downloading data. APIs often provide deep integrations, allowing customers to retrieve and transfer data. For example, many CRMs offer APIs that their users can integrate with their systems and applications. We see a common workflow of pulling data from a CRM, combining the CRM data via the customer ratings model, and then sending that data back to the CRM using reverse ETL to allow sales reps to reach better qualified leads.

Evaluating source systems: Key engineering considerations

When evaluating source systems, there are many things to consider, including how the system handles ingestion, status, and data generation. The following is an initial set of source system evaluation questions for data\engineers to consider:

- What are the key characteristics of the data source? Is an application? A swarm of IoT devices?
- How is data persisted in the source system? Is the data kept long-term or is it deleted temporarily and quickly?
- How fast is the data generated? How many events per second? How many gigabytes per hour?
- What level of consistency can data engineers expect from the output data? When you perform data quality checks on the output data, how often do you find inconsistencies in the data: null values where you do not expect them, bad formatting, etc.?
  - How often do the errors occur?
  - Does the data contain duplicates?
  - Will some data values arrive late, much later than other concurrently generated messages?
  - What is the schema of the recorded data? Do data engineers need to merge multiple tables or even multiple systems to get a complete picture of the data?
  - How is this handled and communicated with downstream stakeholders when the schema changes (e.g., when a new column is added)?
  - How often should data be retrieved from the source system?
  - Is data such as periodic snapshots or change data capture (CDC) update events provided for stateful systems (e.g., a database that tracks customer account information)?)?What is the logic behind the changes and how are they tracked in the source database?

Notes

- Who/what is the data supplier who transmits the data for further use?
- Does reading a data source affect its performance?
- Does the source system have any upstream data dependencies? What are the characteristics of these upstream systems?
- Are there data quality checks to check for missing or overdue data?

Sources produce data that is processed by downstream systems, including human-made spreadsheets, IoT sensors and web and mobile applications. Each source has its unique volume and data generation rate. A data engineer needs to understand how the source generates data, including all relevant specifics and nuances. Data engineers must also understand the limitations of the source systems they interact with. For example, will analysis queries against a source application database cause performance issues and resource contention?

One of the most challenging nuances of source data is the schema. The schema defines the hierarchical organisation of the data. Logically, we can think of data at the level of an entire source system and go deeper into individual tables, down to the structure of the respective fields. The schema of the data sent from the source systems is handled in diverse ways. Two popular options are no scheme and fixed scheme.

Schemaless does not mean that there is no schema. Rather, it means that the application defines the schema when data is written, whether to a message queue, flat file, blob, or document database like MongoDB. A more traditional model, based on relational database storage, uses a fixed schema applied in the database that application writes must conform to.

Each of these models poses challenges for data engineers. Schemas change over time; In fact, the evolution of the schema is encouraged by the agile approach to software development. An important part of the data engineer's job is taking raw data inputs into the source system schema and turning them into valuable outputs for analysis. This task becomes increasingly challenging as the source schema evolves.

Evaluating storage systems: Key engineering considerations

Here are some key technical questions to ask when choosing a storage system for a data warehouse, data lake, database, or object store:

- Does this storage solution support the write and read speeds required by the architecture?
- Will storage create a bottleneck for downstream processes?
- Do you understand how this storage technology works? Are you making the best use of the save system or committing unnatural acts? For example, are you applying a high rate of random-access updates to an object storage system? (This is an anti-pattern with a significant performance overhead.)
- Will this storage system handle the anticipated future size? You must consider all storage system capacity limits: total available memory, read speed, write volume, etc.
- Will users and intermediate processes be able to retrieve data within the required Service Level Agreement (SLA)?
- Do you collect metadata about schema development, data flows, data lineage, etc.? Metadata has a significant impact on the usefulness of the data. Metadata represents an investment in the future, enhancing discoverability and institutional insight to optimize future projects and architecture changes.

Notes

- Is it a storage-only solution (object storage) or does it support complex query patterns (e.g., a cloud data store)?
- Is the storage system independent of the schema (object storage)? Flexible scheme (Cassandra)?Enforced schema (a cloud data store)?
- How do you track master data, golden record data quality and data lineage for data management?
- How do you manage regulatory compliance and data sovereignty? For example, can you store your data in certain geographic locations but not in others?

2.1.4 Data Engineering Life Cycle

The data landscape is witnessing an explosion of new data technologies and practices with increasing levels of abstraction and ease of use. Due to increasing technical abstraction, data engineers are increasingly becoming data lifecycle engineers, thinking, and acting in terms of data lifecycle management principles.

The Data Engineering Life Cycle is our framework that describes data engineering "from the cradle to the grave." You will also learn about the fundamentals of the data engineering lifecycle, which are the key foundations for any data engineering endeavour.

What Is the Data Engineering Lifecycle?

The data engineering lifecycle includes stages that transform raw data pieces into a useful end product that can be consumed by analysts, data scientists, ML engineers and others. This section introduces the main phases of the data engineering life cycle, focusing on the basics of each phase and saving the details for later topics.

We divide the data engineering life cycle into five phases:

- Generation
- Storage
- Ingestion
- Transformation
- Serving data

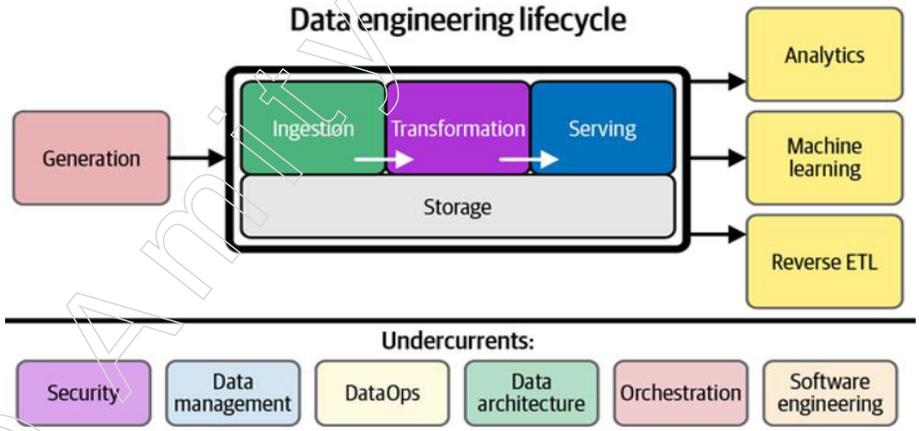


Figure. Components and undercurrents of the data engineering lifecycle

We begin the data engineering lifecycle by retrieving and storing data from source systems. We then transform the data and then move on to our core goal of making data available to analysts, data scientists, ML engineers and others. In reality, storage occurs

Notes

throughout the lifecycle as data flows from one end to the other. Therefore, the diagram shows the storage "tier" as a baseline that supports other tiers.

In general, the intermediate stages (storage, consumption, conversion) can be confusing. And that is okay. Although we break down the various parts of the data engineering lifecycle, the flow is not always seamless and orderly. Distinct phases of the life cycle can repeat themselves, happen out of order, overlap, or intertwine in interesting and unexpected ways.

It is built on undercurrents that traverse multiple stages of the data engineering lifecycle: security, data management, DataOps, data architecture, orchestration, and software engineering. No part of the data engineering lifecycle can function properly without these undercurrents.

The Data Lifecycle Versus the Data Engineering Lifecycle

You may be wondering what the difference between the general data lifecycle and the data engineering lifecycle is. There is a subtle difference between the two. The data engineering lifecycle is a subset of the overall data lifecycle. While the full data lifecycle encompasses data throughout its lifecycle, the data engineering lifecycle focuses on the phases that a data engineer controls.

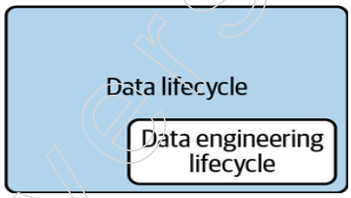


Figure. The data engineering lifecycle is a subset of the full data lifecycle

Generation: Source Systems

A source system is the data source used in the data engineering lifecycle. For example, a source system could be an IoT device, an application message queue, or a transactional database. A data engineer consumes data from a source system, but typically does not own or control the source system itself. The data engineer must have a solid understanding of how source systems work, how they generate data, the frequency and velocity of data and the variety of data they generate.

Engineers must also maintain open communication with source system owners about changes that could disrupt pipelines and analysis. The application code can change the structure of the data in a field, or the application team can even choose to migrate the backend to an entirely new database technology.

A major challenge in data engineering is the dizzying variety of data sources that systems engineers need to work with and understand. To illustrate, let us look at two common source systems, one very traditional (an application database) and the other a more recent example (IoT swarms).

There are traditional source systems with multiple application servers supported by one database. This source system pattern became popular in the 1980s with the explosive success of relational database management systems (RDBMS). The Application + Database pattern is still popular today in several modern advancements in software development practices. For example, applications often consist of many small microservices service/database pairs rather than a single monolith.

Notes

Another example of a source system is an IoT swarm: a fleet of devices (circles) sends data messages (rectangles) to a central collection system. This IoT source system is used increasingly with the increase of IoT devices such as sensors, smart devices, etc.

Understanding Data Access Frequency

Not all data is accessed in the same way. Recovery patterns vary depending on the data stored and queried. This brings up the term "data temperatures". The frequency of data access determines the temperature of your data.

The data that is accessed most frequently is referred to as hot data. Hot data is often accessed multiple times a day, even multiple times per second, for example in systems that handle user requests. This data should be stored for quick retrieval, where "fast" is use case relative. Warm data may be accessed from time to time, such as every week or month.

Cold data is seldom queried and is suitable for storage in a file system. Cold data is often kept on another system for compliance reasons or in the event of a catastrophic failure. Historically, cold data was stored on tape and sent to remote archival facilities. In cloud environments, providers offer special storage tiers with exceptionally low monthly storage costs but high data recovery prices.

Selecting a Storage System

What kind of storage solution should be used? This depends on your use cases, data volumes, frequency of ingestion, format and size of the data collected; the key considerations outlined in the questions listed above. There is no universal storage recommendation. Each storage technology has its advantages and disadvantages. There are countless different storage technologies out there and it is easy to feel overwhelmed when trying to decide which option is best for your data architecture.

Ingestion

After you understand the data source, the characteristics of the source system you are using and how the data is stored, you need to collect the data. The next phase of the data engineering life cycle is the ingestion of data from source systems.

In our experience, source systems and ingestion represent the biggest bottlenecks in the data engineering life cycle. Source systems are usually out of your direct control and may become unresponsive or provide inferior quality data by accident. Or your data collection service may mysteriously stop working due to assorted reasons. This results in the data flow stopping or not providing enough data for storage, processing, and service. Unreliable source and ingest systems impact the entire data development lifecycle. But you are in decent shape, provided you have answered the big questions about source systems.

Key Engineering Considerations for the Ingestion Phase

When preparing to design or build a system, here are some key questions to ask about the ingestion phase:

- What use cases are there for the data that will be recorded? Can this data be reused instead of creating multiple versions of the same data set?
- Do the systems reliably generate and integrate this data and is it available when one needs it?
- What is the destination of the data after recording?
- How often do one need to access the data?

Notes

- On which medium does the data usually arrive?
- What format is the data in? Can the downstream transformation and storage systems handle this format?
- Is the source data in good condition for immediate further use? If so, how long and what can make it useless?
- If the data comes from a streaming source, does it need to be transformed before it reaches its destination? Would an in-flight transformation, where the data is transformed within the stream itself, be appropriate?

Batch Versus Streaming

All of the data we process is transmitted. Data is always generated at its source and continuously updated. Batch recording is simply a special and convenient way to process that stream in large chunks, e.g., B. to process an entire day's worth of data in a single batch. By including streaming, we can provide data continuously and in real-time to downstream systems, be the other applications, databases, or analytics systems. Real time (or near real time) here means that the data is available to a downstream system shortly after it is created (e.g., B. less than a second later). The latency required to be considered real-time varies by domain and requirements batches of data are collected at a predetermined time interval or when the data reaches a preset size threshold. Batch ingestion is a two-way street: once data is batched, latency for downstream consumers is inherently limited. Due to the limitations of older systems, batch processing has long been the standard method of data collection. Batch processing continues to be an extremely fashionable way to collect data for later consumption, especially in analytics and ML.

However, the separation of storage and processing power in many systems and the ubiquity of streaming and event processing platforms make continuous processing of data streams much more accessible and increasingly popular. The choice depends on the use case and expectations of data freshness.

Key Considerations for Batch Versus Stream Ingestion

Should You Stream First? Despite the appeal of a streaming-first approach, there are many pros and cons to understand and consider. Here are some questions to ask yourself when determining if streaming recording is a viable option over batch recording:

- If one ingests the data in real time, can subsequent storage systems handle the data flow rate?
- Do one need real-time data collection in milliseconds? Or would a microbatch approach work, collecting and ingesting data every minute, for example?
- What are the use cases for streaming capture? What are the specific benefits of implementing streaming? If one is getting real-time data, what actions can one take with that data to improve the stack?
- Does the stream-first approach cost more time, money, maintenance, downtime, and opportunity costs than just batch processing?
- Are the pipelines and transmission system reliable and redundant if the infrastructure fails?
- Which tools are best suited for the use case? Should one use a managed service (Amazon Kinesis, Google Cloud Pub/Sub, Google Cloud Dataflow) or create their own instances of Kafka, Flink, Spark, Pulsar, etc.? If one does the latter, who will make it? What are the costs and trade-offs?

Notes

- How does one benefit from online predictions and continuous training when implementing an ML model?
- Does one get data from a live production instance? If so, what impact does the recording process have on this source system?

As you can see, streaming first seems like a clever idea, but it is not always easy; There are additional costs and inherent complexities. Many great recording frameworks handle both batch and microbatch recording styles. We believe Batch is an excellent approach for many common use cases, such as model training and weekly reporting. Do not adopt real-time streaming until you have identified a business use case that warrants compromises over batch usage.

Push Versus Pull

In the data ingestion push model, a source system writes data to a destination, be it a database, object storage, or file system. In the pull model, data is retrieved from the source system. The line between push and pull paradigms can be quite blurred; Data is frequently pushed and pulled as it moves through the various stages of a data pipeline.

With streaming ingestion, data bypasses a back-end database and is sent directly to an endpoint, typically with data buffered by an event-streaming platform. This pattern is useful with fleets of IoT sensors that transmit data from sensors. Rather than relying on a database to maintain current state, we simply view each recorded read as an event. This pattern is also gaining popularity in software applications because it simplifies real-time processing, allows application developers to customize their messages for further analysis and simplifies the life of data engineers.

Transformation

After you have recorded and stored data, you need to do something with it. The next phase of the data engineering lifecycle is transformation. This means that the data needs to be transformed from its original form into something useful for later use cases. Without the right transformations, the data remains dormant and not in a form useful for reporting, analysis, or ML. Typically, the transformation phase is when the data begins to create value for consumption by downstream users.

Immediately after ingestion, the basic transformations map data to the correct types (e.g., change ingested string data to date and numeric types), kill records to standard formats and remove bad records. Subsequent transformation stages can transform the data schema and apply normalization. Downstream, we can apply large-scale aggregation to generate reports or characterize data for ML processes.

Key Considerations for the Transformation Phase

When considering data transformations within the data engineering lifecycle, it is useful to consider the following:

- What are the costs and the return on investment (ROI) of the transformation? What is the associated commercial value?
- Is the transformation as simple and isolated as possible?
- Which business rules support the transformations?

You can convert data in batch or in-flight transfer. As mentioned in Ingestion, all data begins life as a continuous stream; Batch is just a distinct way of processing a stream of data. Batch transforms are overwhelmingly popular, but with the growing popularity

Notes

of streaming processing solutions and the overall increase in the amount of data being transferred, we expect that streaming transforms will continue to grow in popularity and may soon replace processing entirely.

Logically, we treat transformation as a separate area of the data engineering lifecycle, but the reality of the lifecycle can be much more complicated in practice. The transformation often gets entangled in other phases of the life cycle. Typically, data is transformed during ingestion on source systems or during transmission. For example, a source system can add an event timestamp to a record before sending it to an ingestion process. Or a data set within a streaming pipeline can be "enriched" with additional fields and calculations before being sent to a data store. Transformations are ubiquitous in various parts of the life cycle. Data preparation, data dispute and data cleansing: these transformative tasks create value for end users of data.

Data characterization for ML is another data transformation process. Characterization aims to extract and enhance data features useful for training ML models. Characterization can be a dark art, combining domain expertise (to figure out which features might be important for prediction) with deep data science expertise. The primary focus of these modules is that once data scientists have determined how to characterize the data, data engineers can automate the characterization processes in the transformation phase of a data pipeline.

Serving Data

You have reached the final phase of the data engineering lifecycle. Now that the data has been captured, stored, and transformed into coherent and useful structures, it is time to get value out of your data. "Getting values from data" has different meanings for different users.

Data has value when used for practical purposes. Data that is not consumed or queried is simply inert. Data vanity projects pose a significant risk for companies. In the big data era, many companies pursued vanity projects, accumulating huge datasets in data lakes that were never put to any meaningful use. The cloud age is fuelling a new wave of vanity projects based on the latest data warehouses, object storage systems and streaming technologies. Data projects must be purposeful throughout the lifecycle. What is the ultimate business purpose of the data so carefully collected, cleaned, and stored?

Data maintenance is the most exciting part of the data engineering lifecycle. This is where the magic happens. Here ML engineers can apply the most advanced techniques. Let us look at some of the most popular uses for data: analytics, ML and reverse ETL.

Analytics

Analytics is at the heart of most data efforts. Once your data is stored and transformed, you can create reports or dashboards and perform ad hoc analysis of the data. While most analytics used to include BI, today it also includes other facets such as operational analytics and embedded analytics. Let us briefly review these variations of analysis.

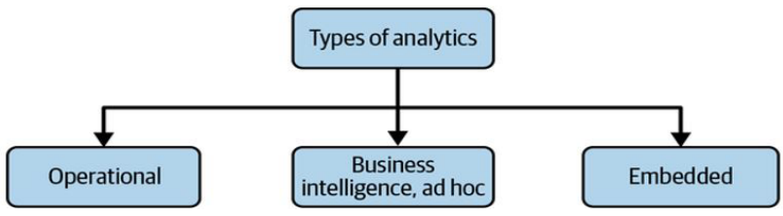


Figure. Types of analytics

Notes

Business Intelligence

BI marshals collected data to describe a company's past and present state. BI requires the use of business logic to process raw data. Keep in mind that making data available for analytics is another area where phases of the data engineering lifecycle can get involved. As previously mentioned, business logic is often applied to data during the transformation phase of the data engineering lifecycle, but a "logic-on-read" approach is gaining popularity. The data is stored in a clean but raw form, with a minimum post-processing business logic. A BI system maintains a repository of business definitions and logic. This business logic is used to query the data warehouse so that reports and dashboards match the business definitions.

As an organisation increases its data maturity, it will move from ad hoc data analytics to self-service analytics, providing business users with democratized data access without the need for IT intervention. The ability to perform self-service analysis means the data is so good that employees across the organisation can easily access it themselves, slice and dice it as they please and uncover instant insights. Although self-service analytics is simple in theory, it is difficult to implement in practice. The main reason for this is that poor data quality, organisational silos and a lack of appropriate data skills often stand in the way of widespread use of analytics.

Operational Analytics

Operational Analytics focuses on the granular details of operations and drives actions that a reporting user can act on immediately. Operational analytics can be a live view of inventory or a real-time dashboard of website or app health. In this case, data is consumed in real-time, either directly from a source system or from a data transfer pipeline. The types of information in operational analysis differ from traditional BI in that operational analysis focuses on the present and does not necessarily relate to historical trends.

Embedded Analytics

You may be wondering why we have broken out embedded analytics (customer-centric analytics) separately from BI. In practice, analytics provided to clients on a SaaS platform come with different requirements and complications. Internal BI has a limited audience and typically offers a limited set of unified views. Access controls are important, but not overly complicated. Access is managed through a handful of roles and access levels.

With embedded analytics, the report request rate and associated load on analytics systems increases dramatically; Access control is significantly more complicated and critical. Businesses may provide separate data and analytics to thousands or more customers. Every customer should see his data and only his data. An internal data access error within a company would lead to a procedural review. A data leak between customers would be a massive breach of trust, which would lead to media attention and a significant loss of customers minimize your blast radius related to data leaks and security breaches. Enforce data or tenant security in your storage and wherever there is a risk of data leakage.

Machine Learning

The emergence and success of ML is one of the most exciting technological revolutions. Once organisations have reached an elevated level of data maturity, they can begin to identify ML-prone issues and begin organizing a practice around them.

Notes

Data engineering responsibilities overlap significantly in analytics and ML and the lines between data engineering, ML engineering and analytics engineering can be blurred. For example, a data engineer might need to support Spark clusters that make it easier to train ML models and analytics pipelines. They may also need to provide a system that orchestrates tasks across teams and supports metadata and cataloguing systems that track data history and lineage. Defining these areas of responsibility and the relevant reporting structures is an important organisational decision.

Feature Store is a newly developed tool that combines data engineering and ML engineering. Feature stores are designed to ease the operational burden on ML engineers by managing feature history and versioning, supporting feature sharing between teams, and providing basic orchestration and operational functionality such as replenishment. In practice, data engineers are part of the core support team for functional storage in support of ML engineering.

Should a data engineer be familiar with ML?It definitely helps. Regardless of the operational boundary between data engineering, ML engineering, business analytics, etc., data engineers must maintain operational knowledge across their teams. A good data engineer is familiar with the basic ML techniques and associated data processing requirements, the use cases for the models in their organisation and the responsibilities of the various analysis teams in the organisation. This helps maintain efficient communication and facilitates collaboration. Ideally, data engineers collaborate with other teams to develop tools that neither team can develop independently.

The following are some considerations for the ML-specific data delivery phase:

- Is the data of sufficient quality to perform reliable feature engineering? Quality requirements and ratings are developed in close collaboration with the teams using the data.
- Can the data be recognized? Can data scientists and ML engineers easily find valuable data?
- Where are the technical and organisational boundaries between data engineering and ML engineering? This organisational question has important architectural implications.
- Does the data set correctly represent the reality of the site? Is it unfairly biased?

While ML is exciting, our experience is that companies often jump into it prematurely. Before investing a ton of resources in ML, take the time to build a solid database. That means configuring the best systems and architectures across the entire ML and data engineering lifecycle. In general, it is best to develop analytical skills before moving on to ML. Many companies have dashed their ML dreams because they launched initiatives that lacked the right foundation.

Reverse ETL

Reverse ETL has long been a practical reality in the data space and is seen as an anti-pattern that we do not like to talk about or put a name to. Reverse ETL takes the processed data from the output side of the data engineering lifecycle and feeds it back to the source systems. In reality, this flow is useful and often necessary; Reverse ETL allows us to take analytics, scored models, etc. and plug them back into production systems or SaaS platforms.

Notes

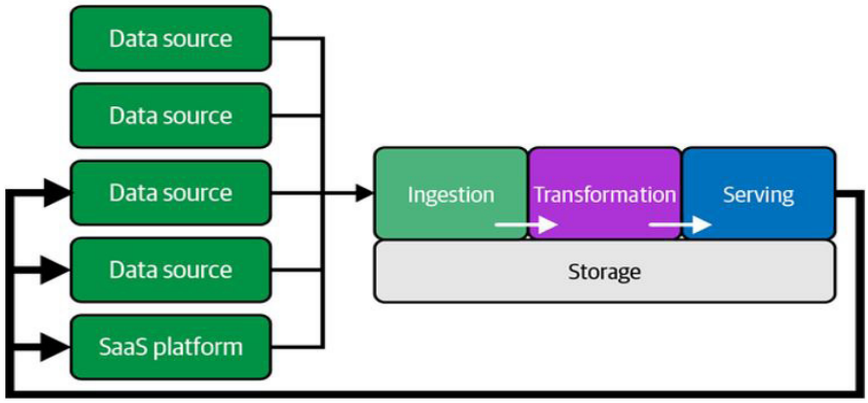


Figure. Reverse ETL

Reverse ETL Marketing analysts at can calculate bids in Microsoft Excel using data from their data warehouse and then upload those bids to Google Ads. This process was often entirely manual and primitive.

Reverse ETL has become particularly important as companies increasingly rely on SaaS and external platforms. For example, companies may want to send specific metrics from their data warehouse to a customer data platform or CRM system. Another everyday use case is advertising platforms, such as Google Ads. Expect more activity in the reverse ETL space with overlaps in both data engineering and ML engineering.

The jury is out on whether the term "reverse ETL" will stick. And practice can evolve. Some engineers claim that we can eliminate reverse ETL by processing data transformations in an event stream and sending those events to source systems when needed. The widespread adoption of this pattern across all organisations is another matter. The essence is that the transformed data must be returned to the source systems in some way, ideally with the correct lineage and business process linked to the source system.

Major Undercurrents Across the Data Engineering Lifecycle

Data technology is developing rapidly. While previous data engineering cycles only focused on the technology layer, the increasing abstraction and simplification of tools and practices has changed this approach. Data engineering now encompasses much more than just tools and technology. The space is now moving up the value chain, integrating traditional business practices like data management and cost optimization with newer practices like DataOps.

We have identified these underlying current practices (security, data management, DataOps, data architecture, orchestration, and software engineering) that support all aspects of the data engineering lifecycle. In this section, we provide a brief overview of these subsurface currents and their main components, which you will explore in more detail throughout the modules.

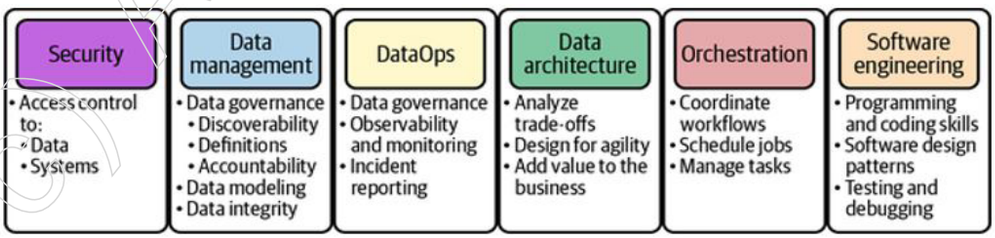


Figure. The major undercurrents of data engineering

Notes

Security

Security should be a top priority for data engineers and those who ignore it do so at their peril. Therefore, safety is the first undercurrent. Data engineers must understand both data and access security while applying the principle of least privilege. The principle of least privilege means granting a user or system access only to data and resources necessary to perform an intended function. A common anti-pattern we see from data engineers with little security experience is to grant all users admin access. This is a disaster just waiting to happen!

Give users only the access they need to do their jobs today, no more. Do not work from a root shell if you are only looking for files visible with standard user access. When querying tables with a minor role, do not use the superuser role in a database. By imposing the principle of least privilege on us, you can prevent accidental damage and ensure your safety comes first.

People and organisational structures are always the biggest security gaps in any company. When we hear about serious security breaches in the media, it often turns out that someone in the organisation failed to take basic precautions, was the victim of a phishing attack, or acted irresponsibly. The first line of defense for data security is creating a security culture that permeates the entire organisation. Everyone who has access to the data must be aware of their responsibility to protect the confidential data of the company and its customers.

Data security is also about time: granting access to data to the people and systems who need it and only for as long as it is necessary to do their job. Data must be protected from unwanted visibility through the use of simple and strong encryption, tokenization, data masking, obfuscation, and access controls, both in transit and at rest.

Data engineers must be competent security administrators because security is their job. A data engineer must understand cloud and on-premises security best practices. Knowledge of Identity and User Access Management (IAM) roles, policies, groups, network security, password policies and encryption is a good place to start.

Data Management

You are thinking that data management sounds very... corporate. Old-school data management practices are making their way into ML and data engineering. The old is new again. Data management has been around for decades, but until recently it had little acceptance in data engineering. Data tools are becoming simpler and data engineers have less complexity to manage. This advances the data engineer to the next best practice level in the value chain. Data best practices that used to be reserved for large companies — data governance, master data management, data quality management, metadata management — have now leaked out to companies of all sizes and maturity levels. As we like to say, data engineering is becoming a “business.” This is a wonderful thing!

The International Data Management Association (DAMA) Data Management Body of Knowledge (DMBOK), which we consider the authoritative book for enterprise data management, provides this definition:

Data management is the development, execution and oversight of plans, policies, programs, and practices that provide, control, protect and enhance the value of data and information assets throughout their lifecycle.

That is a bit long, so let us see how it relates to data tech. Data engineers manage the data lifecycle and data management encompasses the set of best practices that data

Notes

engineers use to accomplish this task both technically and strategically. Without a data management framework, data engineers are just technicians operating in a vacuum. Data engineers need a broader perspective on the value of data across the enterprise, from source systems to executive level and everywhere in between.

Why is data management important? Data management shows that data is vital to day-to-day operations, just as companies view financial resources, finished products, or real estate as assets. Data management practices provide a coherent framework that anyone can adopt to ensure the organisation is getting value from data and managing it appropriately.

Data management has quite a few facets, including the following:

- Data governance, including discoverability and accountability
- Data modelling and design
- Data lineage
- Storage and operations
- Data integration and interoperability
- Data lifecycle management
- Data systems for advanced analytics and ML
- Ethics and privacy

While these modules are by no means an exhaustive resource on the subject of data management, let us briefly cover some highlights from each area related to data engineering.

Data Governance

According to Data Governance: The Definitive Guide, "Data governance is primarily a data management function to ensure the quality, integrity, security and usability of data collected by an organisation."

We can extend this definition and say that data governance involves people, processes, and technology to maximize the value of data in an organisation while protecting the data with appropriate security controls. Effective data stewardship is built on purpose and supported by the organisation. When data management is random and messy, the side effects can range from unreliable data to security breaches and everything in between. Conscious data management maximizes the organisation's data capabilities and the value generated from the data. It will also (hopefully) prevent a company from making headlines for questionable or downright reckless data practices.

Consider the typical example of poorly implemented data management. A business analyst receives a report request but does not know what data to use to answer the question. You can spend hours searching through dozens of tables in a transactional database, wildly guessing which fields might be useful. The analyst creates a "directional" report but is not entirely sure whether the data underlying the report is accurate or reliable. The recipient of the report also doubts the validity of the data. The integrity of the analyst and all data in the company's systems is in question. The company is confused about its performance, making business planning impossible.

Data governance is the foundation of data-driven business practices and a mission-critical part of the data engineering lifecycle. When data governance is practiced well, people, processes and technology are aligned in a way that treats data as a core business driver. Should data problems arise, they will be rectified immediately.

Notes

The main categories of data governance are discoverability, security, and accountability. Within these main categories there are sub-categories such as data quality, metadata, and privacy. Let us look at each of the main categories one by one.

Visibility – In a data-driven organisation, data must be available and discoverable. End users need fast, reliable access to the data they need to do their jobs. You need to know where the data comes from, how it relates to other data and what the data means.

Key areas of data discovery include metadata management and master data management. Let us briefly describe these areas.

Metadata: Metadata is "data about data" and underlies every stage of the data engineering lifecycle. Metadata is precisely the data needed to make the data discoverable and controllable.

We divide metadata into two main categories: auto-generated and human-generated. Modern data engineering revolves around automation, but collecting metadata is often manual and error prone.

Technology can help with this process and eliminate much of the error-prone work of manual metadata collection. We are witnessing an increasing proliferation of data catalogues, data lineage tracing systems and metadata management tools. The tools can search databases to find relationships and monitor data pipelines to track where data is coming from and where it is going. A manual, low-fidelity approach leverages an internally driven effort where multiple stakeholders collaborate on metadata collection within the organisation. These data management tools are covered extensively throughout the modules because they disrupt much of the data engineering lifecycle.

Metadata becomes a by-product of data and data processes. However, the biggest challenges remain. In particular, there is still a lack of interoperability and standards. Metadata tools are only as good as their connections to data systems and their ability to exchange metadata. Additionally, automated metadata tools should not completely exclude humans.

Dates have a social component; Every organisation accumulates social capital and knowledge around processes, datasets, and pipelines. Human-centric metadata systems focus on the social aspect of metadata. Airbnb has emphasised this in its various blog posts on data tools, most notably in its original concept of the data portal. Such tools should provide a place where data owners, data consumers and domain experts can be exposed. Internal documentation and wiki tools are an important foundation for metadata management, but these tools also need to be integrated with automated data cataloguing. For example, data scanning tools can create wiki pages with links to relevant data objects.

Once metadata systems and processes are in place, data engineers can make meaningful use of metadata. Metadata is becoming the basis for designing pipelines and managing data throughout its lifecycle.

DMBOK identifies four main categories of metadata useful for data engineers:

- Business metadata
- Technical metadata
- Operational metadata
- Reference metadata

Let us briefly describe each category of metadata.

Notes

Business metadata relates to how data is used in the enterprise, including business and data definitions, data rules and logic, how and where the data is used and data ownership.

A data engineer uses business metadata to answer non-technical "who, what, where and how" questions. For example, a data engineer could be tasked with creating a data pipeline for customer sales analytics. But what is a customer? Is it someone who made a purchase in the last 90 days? Or someone who at any point bought the store that was open? A data engineer would use the right data to look up how a "customer" is defined in business metadata (data dictionary or data catalogue). Business metadata provides a data engineer with the right context and definitions to use the data correctly.

Technical metadata describes the data created and used by systems throughout the data engineering lifecycle. It includes the data model and schema, data lineage, field mappings and pipeline workflows. A data engineer uses technical metadata to create, connect and monitor various systems throughout the data development lifecycle.

Here are some common types of technical metadata that a data engineer will use:

- Pipeline metadata (often produced in orchestration systems)
- Data lineage
- Schema

Orchestration is a backbone that coordinates workflow across multiple systems. Pipeline metadata captured in orchestration systems provides workflow schedule details, system and data dependencies, configurations, connection details and more. Data lineage metadata tracks the origin and changes of data and their dependencies over time. As data flows through the data engineering lifecycle, it evolves through transformations and combinations with other data. Data lineage provides an audit trail for the evolution of data as it travels through different systems and workflows.

Schema metadata describes the structure of data stored in a system such as a database, data store, data lake, or file system. It is one of the main differentiators between different storage systems. For example, object stores do not maintain schema metadata; Instead, this should be managed in a metastore. On the other hand, cloud data stores maintain schema metadata internally.

These are just a few examples of technical metadata that a data engineer should be aware of. This is not an exhaustive list and we cover other aspects of technical metadata throughout the modules.

Operational metadata describes the operational results of various systems and includes statistics about processes, job IDs, application runtime logs, data used in a process and error logs. A data engineer uses operational metadata to determine whether a process succeeded or failed and to determine what data is involved in the process.

Orchestration systems can provide a limited picture of operational metadata, but operational metadata is still typically scattered across many systems. The need for better quality operational metadata and better metadata management is a key motivation for next-generation orchestration and metadata management systems.

Reference metadata is data used to classify other data. This is also called search data. Standard examples of reference data are internal codes, geocodes, units of measure and internal calendar standards. Note that much of the referencing data is maintained entirely internally, but things like geocodes can come from standard external referencing.

Notes

Reference data is a standard for interpreting other data. So, when they change, that change happens slowly over time.

Data accountability: Data accountability means assigning an individual to govern a portion of data. The responsible person then coordinates the governance activities of other stakeholders. Managing data quality is tough if no one is accountable for the data in question. The responsible person then coordinates the governance activities of other stakeholders. Data quality management is difficult when no one is responsible for the data in question.

Note that those responsible for the data do not have to be data engineers. The responsible person can be a software developer, product manager or another role. In addition, the person responsible usually does not have all the necessary resources to maintain the quality of the data. Instead, they coordinate with everyone who touches the data, including data engineers.

Data liability can occur at various levels; Accountability can be at the level of a table or record stream but can also be as granular as a single field entity occurring across many tables. In many systems, one person may be responsible for managing a client ID.For enterprise data management, a data domain is the set of all values that can occur for a particular type of field, as in this ID example. This may seem overly bureaucratic and delicate, but it can seriously affect data quality.

Data quality is the optimization of data towards the desired state and revolves around the question, “What are you getting compared to what you expect?” The data must conform to business metadata expectations. Does the data correspond to the definition agreed by the company?

- According to Data Governance: The Definitive Guide, data quality is defined by three main characteristics:
- Accuracy: Is the data collected accurate? Are there duplicate values? Are the numerical values correct?
- Completeness: Are the records complete? Do all required fields contain valid values?
- Timeliness: Are the documents available in enjoyable time?

Data Modelling and Design

In order to derive business insights from data through business analytics and data science, the data must be in a usable form. The process of transforming data into a usable form is called data modelling and design. While we traditionally view data modelling as a problem for database administrators (DBAs) and ETL developers, data modelling can happen anywhere in an organisation. Firmware engineers designing the data format of a record for an IoT device, or web application developers designing the JSON response to an API call, or a MySQL table schema are all examples of data modelling and design.

Data modelling has become more sophisticated due to the variety of new data sources and use cases. For example, strict normalization does not work well with event data. Fortunately, a new breed of data tools increases the flexibility of data models while maintaining the logical separation of measures, dimensions, attributes, and hierarchies. Cloud data warehouses support the ingestion of copious amounts of denormalized and semi-structured data while supporting common data modelling patterns such as Kimball, Inmon and Data Vault. Data processing frameworks like Spark can ingest a full spectrum of data, from flat-structured relational datasets to raw unstructured text.

Notes

Given the wide variety of data that data engineers must deal with, there is a temptation to give up and abandon data modelling. That is a horrible idea with staggering consequences, which becomes apparent when people mutter about the WORN access pattern or refer to a data deluge. Data engineers must understand modelling best practices and develop the flexibility to apply the appropriate level and type of modelling to the data source and use case.

Data Integrity and Lineage

As data moves through its life cycle, how do you know what system influenced the data or what it is made of as it moves through and is transformed? Data lineage describes the record of a data audit trail throughout its lifecycle, tracking both the systems that process the data and the upstream data on which they depend.

Data Lineage helps with error tracking, accountability and cleansing of data and the systems that process it. It has the obvious benefit of providing an audit trail for the data lifecycle and helping with compliance. For example, if a user wants their data removed from their systems, the provenance of that data allows them to know where that data is stored and what dependencies it has. The line of data has been around for a long time in larger organisations with strict compliance standards. With the increasing spread of data management, however, it is also increasingly being used in smaller companies.

DataOps

DataOps maps Agile, DevOps and Statistical Process Control (SPC) best practices to the data. While DevOps aims to improve the release and quality of software products, DataOps does the same for data products.

Data products differ from software products in the way the data is used. A software product makes certain functions and technical characteristics available to end users. Rather, a data product is based on robust business logic and metrics whose users make decisions or build models that perform automated actions. A data engineer must understand both the technical aspects of building software products, as well as the business logic, quality and metrics required to create great data products.

Like DevOps, DataOps relies heavily on lean manufacturing and supply chain management, combining people, process, and technology to accelerate time to value. As described by Data Kitchen (DataOps Experts):

DataOps is a collection of technical practices, workflows, cultural norms, and architectural patterns that enable:

- Rapid innovation and experimentation that delivers new insights to customers faster
- Extremely high data quality and exceptionally low error rates
- Collaboration between complex groups of people, technologies, and environments
- Clear measurement, monitoring and transparency of results

Lean practices (e.g., shorter lead times and minimizing errors) and the resulting improvements in quality and productivity are things we are excited to see gaining momentum in both software and data operations.

First, DataOps is a set of cultural habits; The data engineering team must maintain a cycle of communicating and collaborating with the business, breaking down silos,

Notes

continuously learning from successes and failures and iterating rapidly. Only when these cultural habits are established can the team with the technology and tools achieve the best results.

Depending on an organisation's data maturity, a data engineer has several options to integrate DataOps into the structure of the overall data engineering lifecycle. If the organisation does not already have data practices or infrastructure in place, DataOps is an entirely new opportunity that can be leveraged from day one. If an existing project or infrastructure is missing DataOps, a data engineer can begin adding DataOps to workflows. We recommend starting with observability and monitoring first to gain insight into how a system is performing and then adding automation and incident response. A data engineer can work with an existing DataOps team to improve the data engineering lifecycle in a data-ready organisation. In any case, a data engineer must understand the philosophy and technical aspects of DataOps.

DataOps consists of three fundamental technical elements: automation, monitoring and observability and incident response. Let us look at each of these parts and how they impact the data engineering lifecycle.

Figure. The three pillars of DataOps

Automation

Automation enables reliability and consistency in the DataOps process, enabling data engineers to quickly implement new product features and improvements to existing workflows. DataOps automation has a similar framework and workflow as DevOps and consists of change management (environment, code, and data version control), continuous integration/continuous delivery (CI/CD) and configuration as code. Like DevOps, DataOps practices monitor and maintain the reliability of technology and systems (data pipelines, orchestration, etc.), with the added dimension of verifying data quality, data/model drift, metadata integrity and more.

Let us take a quick look at the evolution of DataOps automation within a hypothetical organisation. An organisation with a low DataOps maturity often tries to schedule multiple stages of data transformation processes using cron jobs. This works fine for a while. As data pipelines become more complicated, a number of things are likely to happen. If the cron jobs are hosted on a cloud instance, there may be an operational issue with the instance causing the jobs to stop running unexpectedly. As the gap between jobs gets smaller, a job runs longer, causing a subsequent job to fail or produce stale data. Engineers may not notice flaws in their work until they hear from analysts that their reports are out of date.

As the organisation's data maturity increases, data engineers typically adopt an orchestration framework, Airflow or Dagster. Data engineers recognize that airflow is an operational burden, but the benefits of orchestration outweigh the complexity. Engineers will gradually switch their cron jobs to airflow jobs. Dependencies are now checked before jobs are run. More transformation jobs can be packed at any time since each job can start as soon as the upstream data is ready, rather than at a predetermined fixed time. The data engineering team still has room for operational improvements. A data scientist eventually deploys a broken DAG, causing Airflow's web server to go down and rendering the data team inoperable. After much heartache, data engineering team members realize they no longer need to allow manual DAG deployments. In their next

Notes

phase of operational readiness, they rely on automated DAG provisioning. DAGs are tested prior to deployment and monitoring processes ensure that new DAGs are working properly. Additionally, data engineers block deployment of new Python dependencies until the installation is validated. Once automation is rolled out, the data team is much happier and has far fewer headaches.

One of the principles of the DataOps Manifesto is Embrace Change. This does not mean change for the sake of change, but targeted change. At every stage of our automation journey, there are opportunities for operational improvements. Even at the elevated level of maturity that we have described here, there is still room for improvement. Engineers can adopt a next-generation orchestration framework that integrates with better metadata capabilities. Or they could try to build a framework that automatically generates DAGs based on data lineage specifications. The bottom line is that engineers are constantly looking for automation improvements that reduce their workload and increase the value they deliver to the enterprise.

Observability and Monitoring

As we tell our customers, "Data is a silent killer. "We have seen countless examples of erroneous data persisting in reports for months or years. Managers can make important decisions based on this faulty data and only discover the error much later. The results are often poor and sometimes disastrous for the company. Initiatives are undermined and destroyed; years of work wasted. In the worst case, bad data can lead to financial ruin for companies.

Another horror story occurs when the systems that generate the data for reports randomly fail, causing reports to be delayed by several days. The data team does not know until stakeholders ask why reports are late or provide outdated information. Eventually, various stakeholders lose confidence in the capabilities of the core data team and create their own separate teams. The result is many disparate unstable systems, inconsistent reports, and silos.

If you do not watch and monitor your data and the systems that generate it, you will inevitably have your own data horror story. Observability, monitoring, logging, alerting, and tracing are critical to stay ahead of any issues throughout the data engineering lifecycle. We encourage you to engage SPC to understand if the events you are monitoring not performing as expected and which incidents are worth responding to.

Petrella's DODD method, mentioned earlier in this module, provides an excellent framework for considering data observability. DODD is remarkably similar to Test Driven Development (TDD) in software development:

The purpose of DODD is to provide visibility into data and data applications for everyone involved in the data chain, so that everyone involved in the data value chain is able to approve changes to data or data applications at every step, from ingestion to transformation recognize. for analysis: to solve or prevent problems with the data. DODD is focused on making data observability a prime aspect of the data engineering lifecycle.

In later sections, we will cover many aspects of monitoring and observability throughout the data engineering lifecycle.

Incident Response

A well-functioning data team using DataOps will be able to ship new data products quickly. But mistakes inevitably happen. A system can experience downtime, a new data model can corrupt subsequent reports, an ML model can be outdated and provide poor

Notes

predictions - myriad issues can disrupt the data development lifecycle. Incident response is all about leveraging the automation and observability capabilities above to quickly identify the root causes of an incident and resolve it as reliably and quickly as possible.

Incident Response is not just about technology and tools, although they are beneficial. It is also about open and smooth communication, both within the data engineering team and across the organisation. As Werner Vogels, CTO of Amazon Web Services, said, “Everything breaks all the time.” Data engineers need to be prepared for disaster and respond as quickly and efficiently as possible.

Data engineers must proactively find problems before the company reports them. Errors happen and when stakeholders or end users spot problems, they report them. They will not like that. The feeling is different when they mention these issues with a team and realize they are already actively working to solve them. As an end user, what state of equipment would you trust the most? Trust takes a long time to build and can be lost in minutes. Incident response is both about responding to incidents retrospectively and proactively addressing them before they occur.

Data Architecture

A data architecture reflects the current and future state of data systems that support an organisation’s long-term data needs and strategy. Because an organisation’s data needs are likely to change rapidly, with new tools and practices emerging daily, data engineers need to understand good data architecture. Data architecture is a background of the data engineering life cycle.

A data engineer must first understand the business needs and collect requirements for new use cases. A data engineer must then translate these requirements to develop new ways of collecting and delivering data that prioritize cost and operational simplicity. This means understanding the trade-offs between design patterns, technologies and tools in data source, ingestion, storage, transformation, and delivery systems.

This does not mean that a data engineer is a data architect as they are typically two separate roles. When a data engineer works with a data architect, the data engineer should be able to adhere to the data architect’s designs and provide feedback on the architecture.

Orchestration

Orchestration is not only a core DataOps process, but also an important part of the engineering and implementation flow for data jobs. So, what is orchestration?

Orchestration is the process of coordinating many jobs to run at a scheduled cadence as quickly and efficiently as possible. For example, orchestration tools like Apache Airflow are often referred to as schedulers. That is not completely right. A pure scheduler like cron only knows the time; An orchestration engine embeds metadata about job dependencies, typically in the form of a directed acyclic graph (DAG). The DAG can be run once or scheduled to run at a fixed interval of daily, weekly, hourly, every five minutes and so on.

Because we discuss orchestration in these modules, we assume that an orchestration system stays online with high availability. This allows the orchestration system to continually discover and monitor new jobs without human intervention and run new jobs with each deployment. An orchestration system monitors the jobs it manages and starts new tasks when internal DAG dependencies are completed. You can also monitor external systems and tools to monitor data entry and compliance with criteria.

Notes

If certain conditions are out of bounds, the system also sets error conditions and sends alerts via email or other channels. You can set an estimated end time from 10:00 a.m. to 5:00 p.m. M. for daily data pipelines overnight. If orders are not completed by this time, notifications will be sent to data engineers and consumers.

Orchestration systems also create order history, visualization, and alerting capabilities. Advanced orchestration engines can populate new DAGs or individual tasks when added to a DAG. They also support dependencies on a time range. For example, a monthly reporting job can be used to verify that an ETL job has completed throughout the month before starting it.

Orchestration has long been a key capability for computing, but it has often not been the focus or accessible to anyone but the largest corporations. Organisations used various tools to manage workflows, but these were expensive, unattainable for small startups and not scalable.

Apache Oozie was extremely popular in the 2010s, but it was designed to work within a Hadoop cluster and was difficult to use in a more heterogeneous environment. Facebook developed Dataswarm for internal use in the late 2000s; This inspired popular tools like Airflow, launched by Airbnb in 2014.

Airflow was open source from the beginning and widely used. It was written in Python, which makes it highly extensible to almost any use case imaginable. While there are many other interesting open-source orchestration projects like Luigi and Conductor, Airflow is the leader in mindshare by far. Airflow emerged just as computing was becoming more abstract and accessible and engineers were increasingly interested in orchestrating complex operations across multiple processors and memory systems, particularly in cloud environments.

As of this writing, several early open-source projects aim to emulate the best elements of Airflow's core design while improving it in key areas. Some of the most interesting examples are Prefect and Dagster, which aim to improve the portability and testability of DAGs to help engineers transition from local development to production. Argo is an orchestration engine based on Kubernetes primitives. Metaflow is a Netflix open-source project that aims to improve the orchestration of data science.

Software Engineering

Software engineering has always been a core competency for data engineers. In the early days of contemporary data engineering (two thousand–2010), data engineers worked in low-level frameworks and wrote MapReduce jobs in C, C++, and Java. At the height of the big data age (mid-2010s), engineers began using frameworks that abstracted these low-level details.

This abstraction continues to this day. Cloud data warehouses support powerful transformations using SQL semantics. Tools like Spark are easier to use and have evolved from simple coding details to user-friendly data frameworks. Despite this abstraction, software development remains fundamental to data development. We would like to briefly discuss some general areas of software engineering that apply to the data engineering life cycle.

Core Data Processing Code

Although more abstract and easier to maintain, the core data processing code still needs to be written and appears throughout the data engineering lifecycle. Whether ingesting, transforming, or serving data, data engineers must be proficient and productive

Notes

in frameworks and languages such as Spark, SQL, or Beam. We reject the notion that SQL is not code.

It is also imperative that a data engineer understand proper code testing methods such as unit, regression, integration, end-to-end and smoke.

Development of Open-source Frameworks

Many data engineers are heavily involved in the development of open-source frameworks. They adopt these frameworks to solve specific problems in the data engineering lifecycle and then refine the framework's code to improve the tools for their use cases and contribute to the community.

In the age of big data, we have witnessed a Cambrian explosion of data processing frameworks within the Hadoop ecosystem. These tools primarily focused on transforming and deploying parts of the data engineering lifecycle. The specialization of data engineering tools has neither stopped nor diminished, but the emphasis has shifted up the abstraction scale, away from direct data processing. This new generation of open-source tools helps engineers manage, improve, link, optimize and monitor data.

For example, from 2015 to the early 2020s, airflow dominated the orchestration space. Now a new group of open-source competitors (including Prefect, Dagster and Metaflow) have emerged to address Airflow's perceived limitations and offer better metadata handling, portability, and dependency management. Where the future of orchestration is headed is unknown.

Before data engineers start developing new internal tools, they should examine the landscape of publicly available tools. Keep an eye on the total cost of ownership (TCO) and opportunity cost associated with implementing a tool. There is a good chance there is already an open-source project addressing the problem you are trying to solve and you would do well to collaborate rather than waste a lot of time for no reason.

2.1.5 Architecting the Data Platform

A good data architecture provides built-in functionality at every step of the data lifecycle and backends. We start by defining the data architecture and then discuss the components and considerations. Then we look at specific batch patterns (data stores, data lakes), streaming patterns and patterns that combine batch and streaming. We always value using the possibilities of the cloud to ensure scalability, availability, and reliability.

What is Data Architecture?

Successful data engineering is based on a rock-solid data architecture. The goal of this section is to discuss some common architectural approaches and frameworks and then expand on our persistent definition of what constitutes "good" data architecture. In fact, we will not make everyone happy. Nonetheless, we will present a pragmatic, domain-specific working definition for data architecture that we believe will work for organisations of vastly different scales, business processes and needs.

What is data architecture? Pausing to unwrap it, the subject becomes a bit unclear; Data architecture research provides many inconsistent and often outdated definitions. It is similar to when we defined data engineering: there is no consensus. In a field that is constantly changing, this is to be expected. Before defining the term, it is important to understand the context in which it is placed. Let us briefly review enterprise architecture, which provides the framework for our definition of data architecture.

Notes

Enterprise Architecture Defined

Enterprise architecture has many subsets, including business, technical, application and data sets. Therefore, many frameworks and resources are dedicated to enterprise architecture. In fact, architecture is a surprisingly controversial topic.

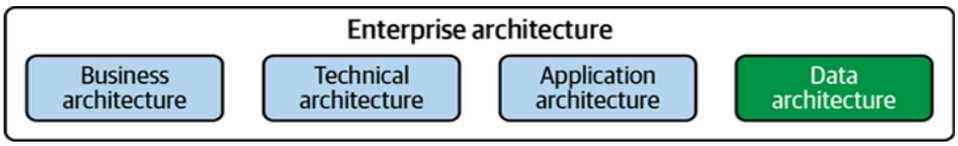


Figure. Data architecture is a subset of enterprise architecture

The term enterprise met with mixed reactions. It evokes sterile corporate offices, cascading command, and control functions/planning, stagnant corporate cultures, and empty slogans. Nevertheless, we can learn a lot here.

Before we define and describe enterprise architecture, let us discuss this term.

Let us look at how some key thought leaders define enterprise architecture: TOGAF, Gartner and EABOK.

TOGAF is the Open Group Architecture Framework, a standard of the Open Group. It is considered to be the most widely used architectural framework today. Here is the definition of TOGAF:

The term “enterprise” in the context of “enterprise architecture” can refer to an entire enterprise, encompassing all of its services, processes and information and technology infrastructure, or to a specific area within the enterprise. In both cases, the architecture spans multiple systems and multiple functional groups within the enterprise.

Gartner is a global research and advisory firm that produces research articles and reports on business trends. Among other things, he is responsible for the (notorious) Hype Cycle by Gartner. The Gartner definition is as follows:

Enterprise Architecture (EA) is a discipline for proactively and holistically leading business responses to disruptive forces by identifying and analysing the implementation of change toward the vision and desired business outcomes. EA delivers value by providing business and IT leaders with recommendations to tailor policies and projects to achieve specific business outcomes that take advantage of relevant business disruptions.

EABOK is the Enterprise Architecture Knowledge Book, a reference to enterprise architecture published by MITER Corporation.EABOK was released as an incomplete draft in 2004 and has not been updated since. Although outdated, EABOK is often mentioned in enterprise architecture descriptions; we found many of your ideas useful in authoring these modules. Here is the definition of EABOK:

Enterprise Architecture (EA) is an organisational model; An abstract representation of a company aligning strategy, operations, and technology to create a roadmap for success.

We extract some common themes in these enterprise architecture definitions: change, alignment, organisation, opportunity, problem solving and migration. Here is our definition of enterprise architecture, which we think is most relevant to today’s fast-paced data landscape:

Enterprise architecture is the design of systems to support change in the enterprise, achieved through flexible and reversible choices made through careful consideration of trade-offs.

Notes

Here we touch on some key areas that we will return to throughout the modules: flexible and reversible choices, change management and trade-off evaluation. We discuss each topic in detail in this section and then refine the definition in the last part of the module by providing several examples of data architecture.

Flexible and reversible choices are essential for two reasons. First, the world is constantly changing and it is impossible to predict the future. Reversible choices allow you to adjust course as the world changes and you gather added information. Second, there is a natural tendency toward corporate freeze as organisations grow. Introducing a culture of reversible choices helps overcome this tendency by reducing the risk associated with a decision.

Jeff Bezos is credited with the idea of the one- and two-way doors. A one-sided door is a decision that is almost impossible to reverse. For example, Amazon might have decided to sell or shut down AWS. It would be almost impossible for Amazon to rebuild a public cloud with the same market position after such an action.

On the other hand, a two-way door is an easily reversible decision: you walk through and move on if you like what you see in the room or walk out if you do not like it. Amazon might decide to require the use of DynamoDB for a new microservices database. If this policy does not work, Amazon has the option to reverse it and refactor some services to use other databases. Because the stakes for each reversible decision (two-way door) are low, organisations can quickly make more decisions, iterate, improve, and collect data.

Architects do not simply map IT processes and look vaguely into a distant, utopian future; They actively solve business problems and create new opportunities. Technical solutions do not exist for their own sake, but in support of business goals. Architects identify problems in the current state (poor data quality, scalability limits, business units losing money), define desired future states (agile data quality improvement, scalable cloud data solutions, improved business processes) and take initiatives by implementing small, concrete steps. It is worth repeating:

Technical solutions do not exist as an end in themselves, but to support corporate goals.

We found a major inspiration in the Software Architecture Fundamentals by Mark Richards and Neal Ford. They emphasise that technical compromises are inevitable and ubiquitous. Sometimes the fluid nature of software and data leads us to believe that we are free from the constraints engineers face in the cold, harsh physical world. In fact, this is partly true; Fixing a software bug is much easier than redesigning and replacing an airplane wing. However, digital systems are limited by physical limitations such as latency, reliability, density, and power consumption. Engineers also face various non-physical limitations such as: B. the characteristics of programming languages and frameworks, as well as practical limitations in managing complexity, budgets, etc. Magical thinking culminates in poor engineering.

Data engineers must consider trade-offs at every step to design an optimal system and minimize high-yielding technical debt.

Let us reiterate a key point in our definition of enterprise architecture: enterprise architecture balances flexibility and trade-offs. This is not always an easy balance and architects need to be constantly aware that the world is dynamic. Given the pace of change facing businesses, organisations, and their architecture, they cannot afford to stand still.

Notes

Data Architecture Defined

Now that you understand enterprise architecture, let us delve deeper into data architecture by creating a working definition that lays the foundation for the rest of the modules. Data architecture is a subset of enterprise architecture and inherits its characteristics: processes, strategy, change management and trade-off evaluation. Here are some data architecture definitions that influence our definition.

TOGAF defines the data architecture as follows:

A description of the structure and interaction of the organisation's major data types and sources, logical data assets, physical data assets and data management resources.

DAMA DMBOK defines the data architecture as follows:

Identify business data needs (regardless of structure) and design and maintain expert plans to meet those needs. Use expert plans to guide data integration, control data assets and align data investments with business strategy.

Taking into account the two previous definitions and our experience, we have elaborated our definition of data architecture:

Data architecture is the design of systems to support an organisation's changing data needs, achieved through flexible and reversible choices made through careful consideration of trade-offs.

How does data architecture fit into data engineering? Just as the data engineering lifecycle is a subset of the data lifecycle, the data engineering architecture is a subset of the overall data architecture. The data engineering architecture consists of the systems and frameworks that form the key sections of the data engineering lifecycle. We use data architecture synonymously with data engineering architecture.

Other aspects of data architecture to consider are operational and technical. The operational architecture encompasses the functional requirements of what needs to happen in terms of people, processes, and technology. Which business processes does the data serve, for example? How does the organisation manage data quality? What is the required latency from creating the data to making it available for the query? Technical architecture describes how data is ingested, stored, transformed, and delivered throughout the data engineering lifecycle. For example, how do you move 10 TB of data every hour from a source database to your data lake? In short: the operational architecture describes what needs to be done and the technical architecture describes in detail how this will be done.

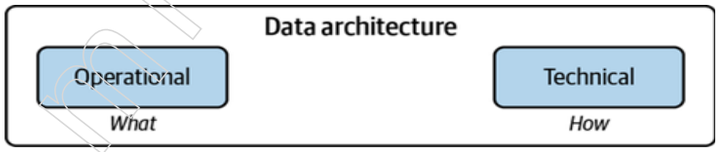


Figure. Operational and technical data architecture

Now that we have a working definition of data architecture, let us look at the elements of a "good" data architecture.

Good Data Architecture

According to Grady Booch, "Architecture represents the major design decisions that shape a system, where importance is measured by the cost of change." Data architects aim to make major decisions that lead to good architecture at a fundamental level .

Notes

What do we mean by "good" data architecture? To paraphrase an old cliché, you know it when you see it. A good data architecture meets business needs with a common and widely reusable set of building blocks while maintaining flexibility and making the appropriate trade-offs. Bad architecture rules and tries to cram a bunch of unique choices into one big ball of mud.

Agility is the foundation of good data architecture; Realize that the world is fluid. A good data architecture is flexible and easy to maintain. It evolves in response to changes in the business and modern technologies and practices that may create even more value in the future. Organisations and their data use cases are constantly evolving. The world is dynamic and the pace of data changes is accelerating. Last year's data architecture that served you well may not be enough today, let alone next year.

Poor data architecture is tightly coupled, rigid, overly centralized, or uses the wrong tools for the task, making development and change management difficult. Ideally, if the architecture is designed with reversibility in mind, changes will be less expensive.

The undercurrents of the data engineering lifecycle form the foundation of good enterprise data architecture at any stage of data maturity. Again, these backgrounds are security, data management, DataOps, data architecture, orchestration, and software development.

A good data architecture is something that lives and breathes. It is never finished. In fact, by our definition, change and evolution are central to the meaning and purpose of data architecture. Now let us look at the principles of good data architecture.

Principles of Good Data Architecture

This section provides a high-level overview of good architecture, focusing on principles—key ideas that are useful for evaluating important architectural decisions and practices. We draw inspiration for our architecture principles from a variety of sources, most notably the AWS Well-Architected Framework and Google Cloud's Five Principles for Cloud Native Architecture.

The AWS Well-Architected Framework consists of six pillars:

- Operational excellence
- Security
- Reliability
- Performance efficiency
- Cost optimization
- Sustainability

Google Cloud's Five Principles for Cloud-Native Architecture are as follows:

- Design for automation.
- Be smart with state.
- Favor managed services.
- Practice defense in depth.
- Always be architecting.

We encourage you to study both frameworks carefully, gaining valuable insights and identifying inconsistencies. With these data engineering architecture principles, we would like to expand or deepen these pillars:

Notes

1. Choose common components wisely.
2. Plan for failure.
3. Architect for scalability.
4. Architecture is leadership.
5. Always be architecting.
6. Build loosely coupled systems.
7. Make reversible decisions.
8. Prioritize security.
9. Embrace FinOps.

Principle 1: Choose Common Components Wisely

One of the primary responsibilities of a data engineer is to select common components and practices that can be widely deployed across an organisation. When architects make good choices and lead effectively, common components become a fabric that facilitates team collaboration and eliminates silos. Common components enable agility within and between teams and the sharing of knowledge and skills.

Common Components can be anything that has broad applicability within an organisation. Common components include object storage, version control systems, observability, monitoring and orchestration systems and processing engines. Common components should be accessible to anyone with an appropriate use case and teams are encouraged to rely on common components already in use rather than reinventing the wheel. Common components must support strong security and permissions to enable asset sharing between teams and prevent unauthorized access.

Cloud platforms are an ideal place to adopt common components. For example, the separation of compute and storage in cloud data systems allows users to access a tier of shared storage (most commonly object storage) using specialized tools to access and query the data needed for specific use cases.

Choosing common components is a balancing act. On the one hand, you need to focus on the needs of the entire lifecycle and data development teams, using common components that are useful for individual projects, while facilitating interoperability and collaboration. On the other hand, architects need to avoid making decisions that reduce the productivity of engineers working on domain-specific problems by forcing them to adopt unique technological solutions.

Principle 2: Plan for Failure

Modern hardware is very robust and durable. However, any hardware component will fail given enough time. To build extremely robust data systems, you must account for errors in your designs. Here are some key terms for evaluating failure scenarios: we describe them in more detail in this section and throughout the modules:

Availability

The percentage of time that an IT service or IT component is in an operational state.

Reliability

The likelihood that the system will meet defined standards in performing its intended function over a specified period of time.

Notes

Recovery time target

The maximum acceptable time for a service or system interruption. The Recovery Time Objective (RTO) is typically determined by determining the business impact of an outage. A one-day RTO might be fine for an internal reporting system. A five-minute website downtime could have a significant negative impact on an online retailer's business.

Recovery point destination

The OK status after recovery. In data systems, data is often lost in the event of a failure. In this configuration, the Recovery Point Objective (RPO) refers to the maximum acceptable data loss.

Engineers must consider acceptable reliability, availability, RTO and RPO when planning for an outage. These guide your architectural decisions when evaluating failure scenarios.

Principle 3: Architect for Scalability

Scalability in data systems involves two main capabilities. First, scalable systems can be scaled to handle copious amounts of data. We might need to set up a large cluster to train a model on a petabyte of customer data or scale a streaming ingestion system to handle a temporary spike in load. Our scalability allows us to handle temporary extreme loads. Second, scalable systems can shrink. As soon as the peak load subsides, we have to automatically reduce capacity in order to reduce costs.(This is related to Principle 9.) An elastic system can scale dynamically in response to the load, ideally in an automated way.

Some scalable systems can also be scaled to zero: they shut down completely when not in use. Once the large model training job is complete, we can delete the cluster. Many serverless systems (eg.serverless functions and online serverless analytics processing or OLAP databases) can be automatically scaled to zero.

Keep in mind that implementing inappropriate scaling strategies can lead to overly complicated systems and excessive costs. A simple relational database with a failover node might be more appropriate for an application than for a complex cluster arrangement. Measure your current utilisation, estimate peak utilisation, and estimate utilisation for the next few years to determine if your database architecture is adequate. If your startup is growing much faster than expected, that growth should also result in more resources being devoted to transforming it for scalability.

Principle 4: Architecture Is Leadership

Data Architects are responsible for technology decisions and architecture descriptions and for propagating these options through effective leadership and training. Data architects must be highly technical, but delegate most of the work of individual contributors to others. Strong leadership qualities paired with high technical competence are rare and extremely valuable. The best data architects take this duality seriously.

Note that leadership does not imply a command-and-control approach to technology. In the past, it was common for architects to choose a proprietary database technology and force each team to host their data there. We reject this approach as it can significantly hamper ongoing data projects. Cloud environments allow architects to balance common component options with the flexibility that enables innovation within projects.

Notes

Coming back to the notion of technical leadership, Martin Fowler describes a specific archetype of an ideal software architect, well embodied by his colleague Dave Rice:

In many ways, Architectus Oryzus' most important activity is mentoring the development team and raising its level so that it can tackle more complex problems. Improving the capacity of the development team gives an architect much more leverage than being the sole decision maker and therefore at risk of becoming an architectural bottleneck.

An ideal data architect shares similar characteristics. You have the technical skills of a data engineer but no longer practice data engineering on a daily basis; They mentor today's data engineers, make thoughtful technology decisions in consultation with their organisation and disseminate expertise through training and leadership. They train engineers in best practices and combine the company's technical resources to pursue common goals in both the technical and business areas.

As a data engineer, you should exercise architectural leadership and seek advice from architects. You may take on the role of architect yourself.

Principle 5: Always Be Architecting

We take this principle directly from Google Cloud's Five Principles for Cloud Native Architecture. Data architects do not just do their job of maintaining the status quo; Instead, they are constantly designing new and exciting things in response to business and technology changes. According to EABOK, an architect's job is to develop a comprehensive understanding of the reference architecture (current state), develop a target architecture and create a sequencing plan to prioritize and sequence architectural changes.

We add that modern architecture should not be based on command and control or cascade, but on collaboration and agility. The data architect maintains a target architecture and sequencing plans that change over time. The target architecture becomes a moving target that adapts in response to internal and global business and technology changes. The sequencing plan establishes the immediate priorities for delivery.

Principle 6: Build Loosely Coupled Systems

The Google DevOps Technology Architecture Guide states, "When the systems architecture is designed to allow teams to test, deploy and change systems without depending on other teams, teams need little communication to get their work done. "In other words: both the architecture and the equipment are loosely coupled."

In 2002, Bezos drafted an email to Amazon employees that became known as the Bezos API Mandate:

1. From now on all teams make their data and functions available via service interfaces.
2. The computers must communicate with each other via these interfaces.
3. No other form of interprocess communication is allowed: no direct links, no direct reading of another computer's data store, no shared memory model, no backdoors of any kind. The only allowed communication is via service interface calls over the network.
4. It does not matter what technology they use. HTTP, Corba, Pubsub, custom protocols, it does not matter.
5. All service interfaces, without exception, must be designed from the ground up to be externalizable. That is, the team must plan and design to make the interface accessible to developers in the outside world. Without exceptions.

Notes

The introduction of Bezos' API mandate is widely seen as a turning point for Amazon. The deposit of data and services behind APIs enabled loose coupling and led to AWS as we know it today. The loosely coupled Google search enabled him to expand his systems to an extraordinary extent.

For the software architecture, a loosely coupled system has the following properties:

1. Systems are divided into many small components.
2. These systems interact with other services via abstraction layers, such as a messaging bus or an API. These layers of abstraction hide and protect the internal details of the service, such as a database backend or inner classes and method calls.
3. Because of property , internal changes in one component of the system do not require changes in other parts. Code update details are hidden behind stable APIs. Each part can be developed and improved separately.
4. Due to property, there is no cascading global release cycle for the entire system. Instead, each component is updated separately as changes and improvements are made.

Note that these are technical systems. We have to think big. Let us translate these technical characteristics into organisational characteristics:

1. Many small teams design a large, complex system. Each team is tasked with designing, maintaining, and improving specific system components.
2. These teams publish the high-level details of their components to other teams via API definitions, message schemas, etc. Teams do not have to worry about other teams' components; You simply use the published API or message specifications to call these components. They iterate their part to improve its performance and skills over time. You can also release new features as they are added or request new things from other teams. Again, the latter happens without the teams having to worry about the internal technical details of the requested features. Teams work together through loosely coupled communication.
3. As a consequence of feature , each team can rapidly develop and improve its component independently of the work of other teams.
4. Specifically, Feature 3 means teams can release updates to their components with minimal downtime. Teams continuously release during regular business hours to make and test code changes.

By loosely coupling technology and human systems, your data engineering teams can collaborate more efficiently with each other and with other parts of the organisation. This principle also directly facilitates principle 7.

Principle 7: Make Reversible Decisions

The data landscape is changing rapidly. Today's trending technology or stack is tomorrow's afterthought. Public opinion changes quickly. You should aim for reversible decisions as they tend to simplify your architecture and keep it agile.

As Fowler wrote, "One of the most important tasks of an architect is to eliminate architecture by finding ways to eliminate irreversibility in software designs." What was true when Fowler wrote this in 2003 is still true today.

As noted, Bezos refers to reversible choices as "two-way doors." As he says, "If you leave and you don't like what you see on the other side, you can't go back sooner." We

Notes

can call these Type 1 choices. But most decisions are not like that, they are changeable, reversible, they are two-way doors. Try to use two-way doors whenever possible.

Given the speed of change and the decoupling/modularization of technologies in your data architecture, you should always strive to select the best solutions that work today. Also, be prepared to update or adopt best practices as the landscape evolves.

Principle 8: Prioritize Security

Every data engineer must take responsibility for the security of the systems they create and maintain. We will now focus on two main ideas: zero trust security and the shared responsibility security model. These are closely based on a cloud-native architecture.

Hardened-perimeter and zero-trust security models

To define Zero Trust security, it is helpful to first understand the traditional hard perimeter security model and its limitations, as outlined in Google Cloud's Five Principles:

Traditional architectures rely heavily on perimeter security, speaking a hardened network perimeter with "trusted things" inside and "untrusted things" outside. Unfortunately, this approach has always been vulnerable to internal attacks, but also to external threats like targeted phishing.

The 1996 film Mission Impossible is a perfect example of the hard perimeter security model and its limitations. In the film, the CIA stores overly sensitive data in a storage system in a room with extremely strict physical security. Ethan Hunt infiltrates CIA headquarters and exploits a human target to gain physical access to the storage system. Once you are in the safe space, you can exfiltrate data with relative ease.

For at least a decade, alarming media reports have alerted us to the growing threat of security breaches that exploit human targets within the organisation's reinforced security perimeters. Even when employees work on highly secure corporate networks, they remain connected to the outside world via email and mobile devices. External threats effectively become internal threats.

In a cloud-native environment, the notion of a hardened perimeter is further undermined. All assets are connected to the outside world. While Virtual Private Cloud (VPC) networks can be defined without external connectivity, the API control plane that engineers use to define these networks is still internet-facing.

The shared responsibility model

Amazon emphasises the shared responsibility model, which separates security into cloud security and cloud security. AWS is responsible for cloud security:

AWS is responsible for protecting the infrastructure that runs AWS services in the AWS cloud. AWS also makes services available to you that are safe to use.

AWS users are responsible for security in the cloud:

Your responsibility depends on the AWS service you use. You are also responsible for other factors, including the confidentiality of your data, the needs of your organisation and applicable laws and regulations.

In general, all cloud providers operate under some form of this shared responsibility model. They ensure their performance according to published specifications. However, it is the user's responsibility to design a security model for their applications and data and to leverage the power of the cloud to make that model a reality.

Notes

Data engineers as security engineers

In today's corporate world, a command-and-control approach to security is prevalent, with network and security teams managing perimeters and overall security practices. The cloud shifts this responsibility to engineers who are not explicitly involved in security functions. Because of this responsibility and a more general erosion of the security space, all data engineers should be considered security engineers.

Failure to accept these new implicit responsibilities can have profound consequences. Numerous data breaches have been caused by the simple mistake of setting up public access Amazon S3 buckets. Anyone who handles data must assume that they are responsible for protecting this data.

Principle 9: Embrace FinOps

Let us start by looking at some FinOps definitions. First, the FinOps Foundation offers:

FinOps is a cloud-based discipline of fiscal management and an evolving cultural practice that enables organisations to achieve maximum business value by helping engineering, finance, technology, and business teams collaborate on data-driven spending decisions.

Furthermore, in Cloud FinOps, J.R. Sorment and Mike Fuller provide the following definition:

The term "FinOps" refers to the emerging professional movement that advocates a collaborative working relationship between DevOps and finance, leading to data-driven, iterative management of infrastructure spend (i.e., lower economics). cloud unit) while increasing costs. the efficiency and the profitability of the cloud environment.

The cost structure of data has changed dramatically in the cloud age. In an on-premises environment, data systems are typically acquired with capital expenditures for a new system every few years in an on-premises environment. Those responsible must balance their budget with the desired computing and storage capacity. Overbuying means wasting money, while underbuying means hampering future data projects and spending a lot of human time controlling system load and data size; Under-procurement may require faster technology refresh cycles, which incurs additional costs.

In the cloud age, most data systems are usage-dependent and easily scalable. The systems can run on a cost-per-query model, a cost-per-throughput model, or some other variant of a pay-per-use model. This approach can be significantly more efficient than the investing approach. Now you can scale up for high performance and then scale down to save money. However, the pay-as-you-go approach makes spending significantly more dynamic. The new challenge for data stewards is to manage budgets, priorities, and efficiency. Cloud tools require a number of processes to manage expenses and resources. In the past, data engineers thought in terms of performance engineering: maximizing the performance of data processes on a fixed set of resources and purchasing appropriate resources for future needs. With FinOps, engineers must learn to think about the cost structures of cloud systems. For example, what is the appropriate mix of AWS Spot Instances when running a distributed cluster? What is the most suitable approach to carry out a large daily work in terms of performance and profitability? When should the company switch from a pay-per-view model to reserved capacity?

FinOps further develops the operational monitoring model to continuously monitor the expenses. Rather than simply monitoring requests and CPU usage for a web server,

Notes

FinOps could monitor running costs of serverless functions processing traffic, as well as spikes in output trigger alerts. Just as systems are designed to fail smoothly in the event of excessive traffic, organisations can consider implementing strict spending limits, with soft failure modes in response to spikes in spending.

Ops teams should also think in terms of attack costs. Just as a distributed denial of service (DDoS) attack can block access to a web server, many companies have found to their dismay that excessive downloads from S3 buckets can skyrocket expenses and threaten bankruptcy for a small startup can. By sharing data publicly, data teams can address these issues by setting payment policies for the subject or simply monitor overspending on data access spending and quickly terminate access if spending spikes to unacceptable levels.

Now that you have a thorough understanding of the principles of good data architecture, let us dig a little deeper into the main concepts you need to design and build good data architecture.

Major Architecture Concepts

Following current data trends, it seems like new types of tools and data architectures are emerging every week. In the midst of this hectic activity, we must not lose sight of the main goal of all these architectures: to transform data into something useful for later consumption.

Domains and Services

Before we delve into architectural components, let us briefly review two terms that come up frequently: domain and services. A domain is the real-world subject area that you design for. A service is a set of functionality whose purpose is to perform a task. For example, you might have a customer order processing service whose job is to process orders as they are created. The sole task of order processing is the processing of orders; Other functions such as inventory management or updating user profiles are not provided.

A domain can contain several services. For example, you might have a sales domain with three services: Orders, Billing and Products. Each service has specific tasks that support the sales area. Other domains can also share services. In this case, the accounting domain is responsible for basic accounting functions: billing, payroll, and accounts receivable (AR). Note that the accounting domain shares the billing service with the sales domain because an invoice is generated when a sale is made and accounting needs to track the invoices to ensure payment is received. Sales and accounting have their respective domains.

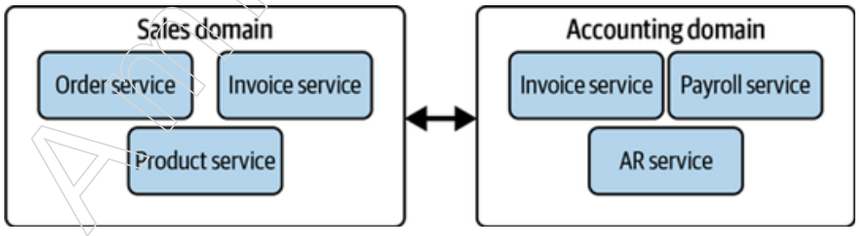


Figure. Two domains (sales and accounting) share a common service (invoices) and sales and accounting own their respective domains

When thinking about what constitutes a domain, focus on what the domain represents in the real world and work backwards. In the example above, the sales domain should represent what is happening with the sales function in your company.

Notes

When designing your sales domain, avoid copying and pasting what other companies do. Your company's sales function has unique aspects that require specific services to function the way your sales team expects.

Identify what should go in the domain. When determining what the domain should encompass and what services should be included, the best way is to simply talk to users and stakeholders, listen to them and build the services that help them do their jobs. Avoid the classic pitfall of designing in a vacuum.

Distributed Systems, Scalability and Designing for Failure

The discussion in this section relates to our previously discussed second and third principles of data engineering architecture: plan for failure and architecture for scalability. As data engineers, we are interested in four closely related characteristics of data systems (availability and reliability were mentioned above, but we repeat them here for completeness):

Scalability

Allows us to increase the capacity of a system to improve performance and handle demand. For example, we might want to scale a system to handle a high query rate or to process a large data set.

Elasticity

The ability of a scalable system to scale dynamically; A highly elastic system can automatically scale up and down based on the current workload. As demand increases, scaling is key, while scaling in a cloud environment saves money. Modern systems sometimes scale to zero, which means they can shut down automatically when idle.

Availability

The percentage of time an IT service or component is in an operable state.

Reliability

The likelihood that the system will meet defined standards in performing its intended function over a specified period of time.

How are these features related? If a system fails to meet performance requirements for a period of time, it may become unresponsive. Therefore, low reliability can lead to low availability. On the other hand, dynamic scaling helps ensure proper performance without manual intervention by engineers: elasticity improves reliability.

Scalability can be achieved in a number of ways. Does a single machine handle everything for your services and domains? A single machine can be scaled; You can increase resources (CPU, disk, memory, I/O). However, there are hard limits to the possible resources on a single machine. And what happens if this machine fails? Given enough time, some components will eventually fail. What is your plan for backup and failover? As a rule, individual machines cannot offer high availability and reliability.

We use a distributed system to achieve higher scalability overall as well as higher availability and reliability. Scale out allows you to add more machines to meet load and resource demands. Common scale-out systems have a lead node that acts as the main point of contact for workload instantiation, progression, and completion. When a workload begins, the leader node dispatches tasks to worker nodes within its system, completes the tasks and returns the results to the leader node. Typical modern distributed

Notes

architectures also include redundancy. Data is replicated so if one machine goes down, the other machines can pick up where the missing server left off; The cluster can add more machines to restore capacity.

Tight Versus Loose Coupling: Tiers, Monoliths and Microservices

When designing a data architecture, you decide how much interdependence you want to build into your different domains, services, and resources. At one end of the spectrum, you can opt for extremely centralized dependencies and workflows. Each part of a domain and service is critically dependent on all other domains and services. This pattern is called tightly coupled.

At the other end of the spectrum, there are decentralized domains and services that are not strictly interdependent, in a pattern known as loose coupling. In a loosely coupled scenario, it is easy for distributed teams to build systems whose data might not be used by their peers. Make sure you assign common standards, ownership, responsibility, and accountability to the teams that own their respective domains and services. The design of a "good" data architecture depends on the trade-offs between tight and loose coupling of domains and services.

It is worth noting that many of the ideas in this section come from software development. We will try to keep the context of the original intent and spirit of these lofty ideas, keep them independent of data and then explain some differences to consider when applying these concepts specifically to data.

Architecture Tiers

As you develop your architecture, it helps to understand the layers of the architecture. Your architecture is made up of layers (data, applications, business logic, presentation, etc.) and you need to know how to decouple those layers. Since tightly coupled modalities present obvious vulnerabilities, you should consider how to layer your architecture to ensure maximum reliability and flexibility. Let us look at the single-tier and multi-tier architecture.

Single Tier

In a single-tier architecture, your database and application are tightly coupled and reside on a single server. This server can be your laptop or a single virtual machine (VM) in the cloud. The tight coupling means that if the server, database, or application fails, the entire architecture fails. While single-tier architectures are good for prototyping and development, they are not recommended for production environments due to obvious risks of failure.

Although single-tier architectures rely on redundancy (e.g., failover replication), they have significant limitations in other respects. For example, it is often impractical (and not recommended) to run analytic queries against production application databases. Otherwise, there is a risk that the database will be overloaded and the application will no longer be available. A single-tier architecture works well for testing systems on a local machine but is not recommended for production use.

Multitier

The challenges of a tightly coupled, single-tier architecture are solved by decoupling the data and the application. A tiered architecture (also known as n-tier) consists of separate tiers: data, application, business logic, presentation, etc. These tiers are bottom-up and hierarchical, meaning that the bottom tier is not necessarily distinct from the top

Notes

tiers is dependent. ; The upper layers depend on the lower layers. The idea is to separate the data from the application and the application from the presentation.

A common multi-tier architecture is a three-tier architecture, a common client-server design. A three-tier architecture consists of data, application logic and presentation tiers. Each level is isolated from the other, allowing for a separation of concerns. With a three-tier architecture, you can use whatever technology you prefer within each tier without the need for a monolithic approach.

We have seen many single tier architectures in production. Single-tier architectures offer simplicity but also come with significant limitations. Eventually, an organisation or application outgrows this agreement; works fine until it does not work anymore. For example, in a single-tier architecture, the data and logic tiers share and compete for resources (disk, CPU and memory) in a way that a multi-tier architecture simply avoids. The resources are distributed at various levels. Data engineers should use layers to evaluate their layered architecture and the way dependencies are handled. Again, start simple and move on to additional levels as your architecture becomes more complex.

In a multi-tier architecture, when working with a distributed system, you should consider the separation of your tiers and the way resources are shared within the tiers. Distributed systems under the hood power many of the technologies you will find throughout the data engineering lifecycle. First, consider whether you want resource contention with your nodes. Otherwise, use a shared-nothing architecture: a single node handles each request, which means other nodes do not share resources such as memory, disk, or CPU with this node or other nodes. Data and resources are isolated on the node. Alternatively, multiple nodes can handle multiple requests and share resources, but at the risk of resource contention. Another consideration is whether the nodes should share the same disk and storage accessible to all nodes. This is called a shared disk architecture and is common when you need shared resources in the event of an accidental node failure.

Monoliths

The general idea of a monolith includes as much as possible under one roof; In its most extreme form, a monolith consists of a single code base running on a single machine, providing both the application logic and user interface.

Coupling within monoliths can be viewed in two ways: technical coupling and domain coupling. Technical coupling refers to the architectural levels while domain coupling refers to the way domains are coupled. A monolith exhibits varying degrees of coupling between technologies and domains. You could have an application with multiple tiers that are decoupled in a multi-tier architecture, but still share multiple domains. Or you could use a single-domain, single-tier architecture.

The tight coupling of a monolith implies a lack of modularity in its components. Replacing or upgrading components of a monolith is often an exercise in trading one problem for another. Because of the tight coupling, component reuse in the architecture is difficult or impossible. When evaluating how a monolithic architecture can be improved, a clash often ensues: one component is improved, often at the expense of unknown consequences for other areas of the monolith.

Microservices

Compared to the characteristics of a monolith (interconnected services, centralization, and tight coupling between services), microservices are the complete

Notes

opposite. Microservices architecture includes separate, decentralized, and loosely coupled services. Each service has a specific function and is decoupled from other services operating in its domain. If a service stops working temporarily, this will not affect the ability of other services to continue to function.

A common question is how to turn your monolith into many microservices. This entirely depends on how complex your monolith is and how much effort it takes to start extracting services from it. Your monolith may not be partitionable. In this case, you should start building a new parallel architecture where the services are decoupled in a microservices-friendly way. We are not proposing a full refactoring, but rather breaking down the services. The monolith did not come overnight and is both a technological and an organisational problem.

Be sure you get buy-in from stakeholders of the monolith if you plan to break it apart.

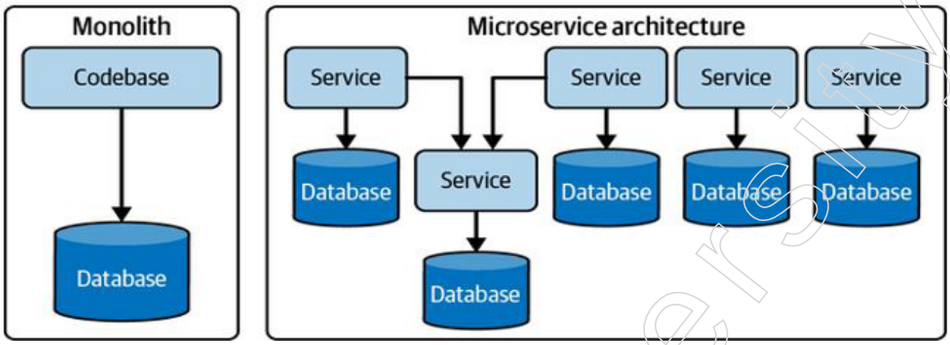


Figure. An extremely monolithic architecture runs all functionality inside a single codebase, potentially co-locating a database on the same host server

Considerations for Data Architecture

As we mentioned at the beginning of this section, the concepts of "tightly coupled" and "loosely coupled" originated in software engineering and some of these concepts date back more than 20 years. Although data architecture practices have now embraced those of software development, it is still common to find very monolithic and tightly coupled data architectures. Part of this is due to the nature of existing data technologies and the way they are integrated.

For example, data pipelines can consume data from many sources that are ingested into a central data store. The core data warehouse is inherently monolithic. A step toward a microservices equivalent of a data store is to decouple the workflow with domain-specific data pipelines that connect to the appropriate domain-specific data stores. For example, the sales data pipeline connects to the sales-specific data warehouse and the inventory and product domains follow a similar pattern.

Rather than dogmatically preaching microservices over monoliths (among other arguments), we propose that you pragmatically use loose coupling as the ideal while acknowledging the status and limitations of the data technologies you use in your data architecture. Incorporate reversible technology options that allow for modularity and flexible coupling wherever possible.

You divide the components of your architecture vertically into various levels of interest. While a layered architecture solves the technical challenges of decoupling shared resources, it does not address the complexities of domain sharing. In the spirit of a single tier vs. multi-tier architecture, you should also consider how to separate the domains of your data architecture. For example, your team of analysts could be based on

Notes

sales and inventory data. The sales and inventory domains are different and should be considered separately.

One approach to solving this problem is centralization: a single team is responsible for collecting data from all domains and reconciling it for use across the organisation. (This is a common approach in traditional data storage.) Another approach is the data network. With the data mesh, each software team is responsible for preparing their data for use by the rest of the organisation. We will talk more about the data network later in this module.

Our Advice: Monoliths are not necessarily bad and under certain conditions it might make sense to start with them. Sometimes you have to be quick and it is much easier to start with a monolith. Just be prepared to break it into smaller pieces over time; Don't get too comfortable.

User Access: Single Versus Multitenant

As a data engineer, you must make decisions about how systems are shared between multiple teams, organisations, and customers. All cloud services are multi-tenant, although this multi-tenancy occurs in multiple grains. For example, a cloud computing instance typically resides on a shared server, but the virtual machine itself provides some level of isolation. Object storage is a multi-tenant system, but cloud providers ensure security and isolation as long as customers set their permissions properly. Engineers often have to make multi-tenancy decisions on a much smaller scale. For example, several departments in a large company use the same data warehouse. Does the organisation share data from multiple major customers in the same spreadsheet?

When it comes to multi-tenancy, we need to consider two factors: performance and security. With multiple large tenants within one cloud system, will the system support consistent performance across all tenants, or will there be an issue with noisy neighbours?(That is, will high usage of one tenant affect the performance of other tenants?) For security reasons, data from different tenants must be properly isolated. If a company has multiple external customer tenants, these tenants must not know each other and technicians must prevent data leakage. Data isolation strategies vary by system. For example, it is often perfectly acceptable to use multi-tenant tables and isolate data across views. However, you must ensure that these views cannot lose data. Read the vendor or project documentation to understand appropriate strategies and risks.

Event-Driven Architecture

Your business is rarely static. Things often happen in your business, such as acquiring a new customer, placing a new customer order, or placing an order for a product or service. These are all examples of events, broadly defined as something that happened, usually a change in the state of something. For example, a customer can create a new order, or a customer can update that order later.

An event-driven workflow includes the ability to asynchronously create, update and move events in various parts of the data engineering lifecycle. This workflow can be broken down into three principal areas: event production, routing, and consumption. An event must be produced and forwarded to something that consumes it without tight dependencies between the producer, the event router, and the consumer.

An event-driven architecture uses the event-based workflow for communication between different services. The benefit of an event-based architecture is that it spreads

Notes

the status of an event across multiple services. This is useful when a service goes offline, a node in a distributed system fails, or when you want multiple consumers or services to access the same events. Anytime you have loosely coupled services, this is a candidate for an event-driven architecture. Many of the examples that we describe later in this module involve some form of event-driven architecture.

Brownfield Versus Greenfield Projects

Before designing your data architecture project, you need to know whether you are starting from nothing or designing an existing architecture from scratch. Each type of project requires weighing the pros and cons, albeit with different considerations and approaches. The projects are divided into two areas: Brownfield and Greenfield.

Brownfield Projects

Brownfield projects often require a redesign and re-engineering of an existing architecture and are constrained by past and current decisions. Because change management is an important part of architecture, you must find a way to circumvent these limitations and map out a way forward to meet your new business and technical goals. Brownfield projects require a deep understanding of legacy architecture and how various old and modern technologies interact. Too often it is easy to criticize the work and decisions of a previous team, but it is far better to engage, ask questions and understand why the decisions were made. Empathy and context go a long way in diagnosing problems with existing architecture, spotting opportunities and spotting impediments.

At some point you will have to adopt your new architecture and technologies and reject the old ones. Let us look at some popular approaches. Many teams rush headfirst into a full or major overhaul of the old architecture, often finding that it is outdated. While this approach is popular, we do not recommend it due to the risks involved and the lack of a plan. This path often leads to disaster with many irreversible and costly decisions. Your job is to make reversible decisions with high ROI.

If you can reject the old architecture, note that there are numerous ways to reject your old architecture. It is critical to demonstrate the value of the new platform by incrementally increasing its maturity to provide proof of success and then following an exit plan to shut down legacy systems.

Greenfield Projects

At the other end of the spectrum, a greenfield project allows you to start over without being constrained by the history or legacy of an earlier architecture. Greenfield projects tend to be easier than brownfield projects and many data architects and engineers find them more fun! You can try the latest and coolest architecture tools and patterns. What could be more exciting?

There are a few things to watch out for before you get too carried away. We are seeing teams with shiny object syndrome getting too cocky. They feel compelled to search for the latest and greatest technological fad without understanding how it will affect the value of the project. There is also a temptation to do resume-based development and embrace impressive modern technology without prioritizing the end goals of the project. Always put requirements before building something cool.

Whether you are working on a brownfield or greenfield project, always focus on the principles of “good” data architecture. Evaluate trade-offs, make flexible and reversible decisions, and aim for a positive ROI.

Notes

Now let us look at examples and types of architectures: some that have been established for decades (the data warehouse), some are brand new (the data lake house) and some that have come and gone quickly but still influence current architectural patterns . (Lambda architecture).

Examples and Types of Data Architecture

Since data architecture is an abstract discipline, it is helpful to argue with examples. In this section, we describe prominent examples and types of data architectures that are popular today. While this set of examples is not intended to be exhaustive, it is intended to familiarize you with some of the more common data architecture patterns and to get you thinking about the flexibility and trade-off analysis required when designing a good architecture for your use case. .

Data Warehouse

A data warehouse is a central data center used for reporting and analysis. The data in a data warehouse is typically highly formatted and structured for analytics use cases. It is one of the oldest and most established data architectures.

In 1989, Bill Inmon developed the idea of a data warehouse, which he described as “an integrated, non-volatile, time-varying, topic-oriented collection of data in support of managerial decisions”. Although the technical aspects of the data warehouse have evolved significantly, we believe this original definition is still valid today.

Historically, data warehouses have often been used by companies with significant budgets (often millions of dollars) to purchase data systems and pay internal teams for ongoing support in maintaining the data warehouse. That was expensive and labour intensive. Since then, the scalable pay-as-you-go model has made cloud data warehouses affordable even for small businesses. With a third party managing the data warehouse infrastructure, organisations can do much more with fewer people, even as their data complexity increases.

Two types of data storage architectures are worth mentioning: organisational and technical. The organisation’s data warehouse architecture organises the data associated with specific sales team structures and processes. The technical data warehouse architecture reflects the technical nature of the data warehouse, such as MPP.A company can have a data warehouse without an MPP system or operate an MPP system that is not organised as a data warehouse. However, the technical and organisational architectures existed in a virtuous circle and are often identified with each other.

The architecture of the organisational data warehouse has two main characteristics:

Separates online analytical processing (OLAP) from production databases (online transaction processing) This separation is essential for the growth of companies. By moving data to a separate physical system, production systems are relieved and analytical performance is improved.

Centralizes and organises data

Traditionally, a data warehouse retrieves data from application systems using ETL. In the extraction phase, data is extracted from the source systems. In the transformation phase, the data is cleaned and standardized and the business logic is organised and enforced in a highly modelled form. In the loading phase, the data is transferred to the target DB system of the data warehouse. The data is loaded into multiple data marts that fit the analysis needs of specific industries or companies and departments. Data

Notes

warehouse and ETL go hand-in-hand with specific business structures, including ETL and DBA development teams that implement directives from business leaders to ensure data for reporting and analysis is aligned with business processes.

The cloud data warehouse

Cloud data warehouses represent a significant evolution of on-premises data warehouse architecture and as such have led to significant changes in organisational architecture. Amazon Redshift started the cloud data storage revolution. Rather than sizing an MPP system over years and signing a multi-million-dollar contract to purchase the system, organisations had the option of running an on-demand Redshift cluster that scaled over time, as data and capacity become available. Analysis require it. grew up. They could even activate new Redshift clusters as needed to service specific workloads and quickly remove clusters when they were no longer needed.

Google BigQuery, Snowflake, and other competitors popularized the idea of separating processing power and memory. In this architecture, data is stored in object storage, allowing for unlimited storage space. This also gives users the ability to increase computing power when needed and provide ad hoc big data capabilities without the long-term cost of thousands of nodes. The Cloud Data Warehouses extend the capabilities of MPP systems to cover many big data use cases that have required a Hadoop cluster in the recent past. You can easily process petabytes of data in a single query. They typically support data structures that allow storage of tens of megabytes of raw text data per line or extremely large and complex JSON documents. As cloud data warehouses (and data lakes) mature, the line between data warehouse and data lake will continue to blur.

The impact of the new capabilities offered by cloud data warehouses is so significant that we might consider abandoning the term data warehouse altogether. Instead, these services are evolving into a new data platform with much richer capabilities than a traditional MPP system.

Data marts

A data mart is a refined subset of a warehouse designed for analysis and reporting and focused on a single sub-organisation, department, or line of business. Each department has its own data mart specifically tailored to its needs. This is in contrast to the entire data warehouse serving the broader organisation or business.

Data marts exist for two reasons. First, a data mart makes data more accessible to analysts and report developers. Second, data marts represent an additional stage of transformation beyond the initial ETL or ELT pipeline. This can significantly improve performance when your reports or analytical queries require complex data joins and aggregations, especially when the raw data is large. The transformation processes can populate the data mart with combined and aggregated data to improve performance for live queries.

Data Lake

One of the most popular architectures to emerge in the big data era is the data lake. Instead of imposing strict structural constraints on the data, why not just put all your data, structured or unstructured, in one central location? The data lake promised to be a democratizing force and allow the company to benefit from an unlimited source of data. The first-generation data lake, Data Lake 1.0, made strong contributions but overall fell short of expectations.

Notes

Data Lake 1.0 started with HDFS. As the cloud grew in popularity, these data lakes transitioned to cloud-based object storage, which offers extremely low storage costs and unlimited storage capacity. Rather than relying on a monolithic data warehouse where storage and compute are tightly coupled, the data lake allows you to store vast amounts of data of any size and type. When that data needs to be queried or transformed, you have access to almost unlimited computing power by activating a cluster on demand and you can choose your preferred data processing technology for the task at hand: MapReduce, Spark, Ray, Presto. , beehive etc.

Despite the promises and hype, Data Lake 1.0 had serious flaws. The data lake became a landfill; Terms like data swamp, dark data and WORN OUT were coined when once promising data projects failed. Data grew to unmanageable sizes with little schema management, data cataloguing and discovery tools. Also, the original concept of the data lake was read-only, which caused major problems with the introduction of regulations like GDPR that required user records to be specifically deleted.

Data processing was also a challenge. Banal data transformations such as joins caused major problems when encoded as MapReduce jobs. Later frameworks like Pig and Hive improved the data processing situation but did little to solve basic data management problems. Simple Data Manipulation Language (DML) operations common in SQL (deleting or updating rows) were cumbersome to implement and were usually accomplished by creating entirely new tables. While big data engineers exuded particular disdain for their data warehousing peers, they could point out that data warehouses immediately provided basic data management capabilities and that SQL was an efficient tool for writing complex and efficient queries and transformations.

Data Lake 1.0 also failed to deliver on another key promise of the big data movement. Open-source software in the Apache ecosystem was encouraged to avoid multi-million-dollar contracts for proprietary MPP systems. Cheap off-the-shelf hardware would replace custom solutions from vendors. In fact, big data costs have skyrocketed as the complexity of managing Hadoop clusters has forced companies to hire large teams of highly paid engineers. Organisations often chose to purchase customized and licensed versions of Hadoop from vendors to avoid the exposed wires and sharp edges of the raw Apache code base and to purchase a set of scaffolding tools to make Hadoop easier to use. Even companies that avoided managing Hadoop clusters using cloud storage had to invest heavily in talent to write MapReduce jobs.

We must be careful not to underestimate the value and power of first-generation data lakes. Many organisations have recognized significant value in data lakes, particularly large, data-centric Silicon Valley tech companies like Netflix and Facebook. These companies had the resources to create successful data practices and build their custom Hadoop-based tools and extensions. But for many organisations, data lakes have become an internal super accumulation of waste, frustration, and rising costs.

Modern Data Stack

The modern data stack is a popular analytics architecture right now, highlighting the type of abstraction that we expect to become more prevalent in the years to come. While previous data stacks relied on monolithic and expensive toolsets, the main goal of the modern data stack is to use cloud-based, ready-to-use, turnkey components to build a modular and cost-effective system. -effective data architecture. These components include data pipelines, storage, transformation, data management/control, monitoring, visualization, and exploration. The domain is still evolving and the specific tools are

Notes

changing and evolving rapidly, but the main goal will remain the same: reduce complexity and increase modularization. Note that the modern data stack idea integrates very well with the converged data platform idea from the previous section.

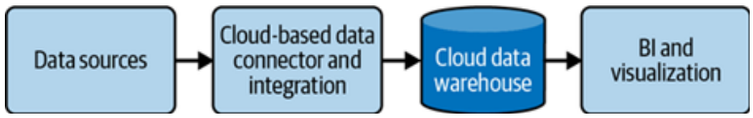


Figure. Basic components of the modern data stack

Key outcomes of the modern data stack are self-service (analytics and pipelines), agile data management and the use of open-source tools or simple proprietary tools with clear pricing structures. Community is also a central aspect of the modern data stack. Unlike products of the past, whose releases and roadmaps were hidden from users, projects and companies operating in the modern data stack space often have strong user bases and active communities that participate in development by providing Use the product early, propose features and submit pull requests to improve the code.

Regardless of where "modern" goes, we believe the key concept of plug-and-play modularity with easy-to-understand pricing and implementation is the way of the future. In analytics engineering in particular, the modern data stack is and will remain the standard choice for data architecture. Throughout the modules, the architecture we refer to incorporates parts of the modern data stack, such as plug-and-play and cloud-based modular components.

Lambda Architecture

In the "old days" (early to mid-2010s), the popularity of working with streaming exploded with the advent of Kafka as a highly scalable message queue and frameworks such as Apache Storm and Samza for real-time streaming/analysis. . These technologies enabled companies to perform new ways of analysing and modelling big data, user aggregation and classification and product recommendations. Data engineers had to figure out how to balance batch and streaming data on a single architecture. Lambda architecture was one of the first popular answers to this problem.

In a Lambda architecture, you have systems that work independently: batch, stream, and service. The source system is ideally immutable and append-only, sending data to two destinations for processing: stream and batch. In-stream processing aims to provide data with the lowest possible latency in a "velocity" layer, typically a NoSQL database. In the batch layer, the data is processed and transformed in a system as a data warehouse, creating aggregated and precomputed views of the data. The service layer provides a combined view by aggregating the query results from the two layers.

The Lambda architecture has some challenges and criticisms. Managing multiple systems with disparate code bases is as difficult as it sounds and results in error-prone systems with code and data that are extremely difficult to reconcile. Next, let us look at a reaction to the lambda architecture, the kappa architecture.

Kappa Architecture

In response to the shortcomings of the Lambda architecture, Jay Kreps proposed an alternative called the Kappa architecture. The central thesis is: Why not just use a stream processing platform as the backbone for all data processing: ingestion, storage, and delivery? This enables a true event-driven architecture. Batch and real-time processing can be seamlessly applied to the same data by reading the live event stream directly and replaying copious amounts of data for batch processing.

Notes

Although Kappa's original architectural paper was published in 2014, we have not seen widespread acceptance. There can be several reasons for this. First, streaming itself remains a mystery to many companies. It is easy to talk about but putting it into practice is harder than expected. Second, the Kappa architecture is complicated and expensive in practice. While some transmission systems can scale to copious amounts of data, they are complex and expensive; Storage and batch processing are still much more efficient and cost-effective for large historical datasets.

The Dataflow Model and Unified Batch and Streaming

Both Lambda and Kappa attempted to overcome the limitations of the Hadoop ecosystem of the 2010s by attempting to tie together complicated tools that did not naturally fit together in the first place. The core challenge of unifying batch and streaming data remained and Lambda and Kappa provided inspiration and foundation for further advances in this task.

One of the key issues in managing batch processing and sequencing is unifying multiple code paths. Although the Kappa architecture is based on a unified queuing and storage layer, you still have to deal with using different tools to collect real-time statistics or run batch aggregation jobs. Nowadays engineers try to solve this problem in diverse ways. Google made its name by developing the Dataflow model and the Apache Beam framework that implements this model.

The central idea in the data flow model is to show all data as events since the aggregation is done in different window types. Continuous real-time event streams are unlimited data. Data stacks are simply bounded streams of events and the bounds provide a natural window. Engineers can choose from multiple windows for real-time aggregation, such as push or mirror. Batch and real-time processing is done on the same system with identical code. The "charge as a special case of transmission" philosophy is now more pervasive. Various frameworks such as Flink and Spark have taken a similar approach.

Architecture for IoT

The Internet of Things (IoT) is the distributed collection of devices, also called things: computers, sensors, mobile devices, smart home devices and anything else with an internet connection. Rather than generating data through direct human input (think data entry from a keyboard), IoT data is generated by devices that regularly or continuously collect data from the environment and transmit it to a destination. IoT devices often underperform and operate in low-resource, low-bandwidth environments.

While the concept of IoT devices dates back at least a few decades, the smartphone revolution spawned a huge swarm of IoT devices overnight. Since then, numerous new IoT categories have emerged, such as smart thermostats, car entertainment systems, smart TVs, and smart speakers. The IoT has evolved from a futuristic fantasy into a vast area of data engineering. We expect the Internet of Things to become one of the predominant ways of generating and consuming data and this section is a bit more detailed than the others you have read. A cursory understanding of IoT architecture will help you understand the broader trends in data architecture. Let us take a quick look at some IoT architecture concepts.

Devices

Devices (also known as things) are the physical hardware that connects to the internet, senses the environment around them and collects and transmits data to a downstream destination. These devices can be used in consumer applications such as

Notes

a doorbell camera, a smart watch, or a thermostat. The device could be an AI-powered camera monitoring an assembly line for failing components, a GPS tracker to record vehicle locations, or a Raspberry Pi programmed to download the latest tweets and brew your coffee. Any device that can collect data from its environment is an IoT device.

Devices must at least be able to collect and transmit data. However, the device can also analyse data or apply ML to the collected data before forwarding it: edge computing or edge machine learning.

Interfacing with devices

A device is only useful if it can retrieve your data. This section covers some of the key components required to interact with outdoor IoT devices.

IoT gateway

An IoT gateway is a hub for connecting devices and securely routing devices to the appropriate destinations on the internet. While you can connect a device directly to the Internet without an IoT gateway, the gateway allows you to connect devices with exceptionally low power consumption. It acts as an intermediary for data retention and manages an internet connection to the final destination of the data.

The new low-power Wi-Fi standards are intended to make IoT gateways less critical in the future but are only now being implemented. Typically, a swarm of devices uses many IoT gateways, one in each physical location where the devices reside.

Ingestion

Ingestion starts with an IoT gateway as explained above. From there, events and metrics can flow into an event ingestion architecture.

Storage

The storage requirement depends on the latency requirements of the IoT devices in the system. For example, for remote sensing, where scientific data is collected for further analysis, stacking objects may be perfectly acceptable. However, near real-time responses can be expected from a system backend constantly analysing data in a home monitoring and automation solution.

Serving

Serving patterns are incredibly diverse. In a scientific batch application, data can be analysed using a cloud data store and then provided in a report. The data is displayed and made available in a variety of ways in a home monitoring app. The data is shortly analysed using a stream processing engine or by querying a time-series database to look for critical events such as fire, power failure or theft. The detection of an anomaly triggers alarms for the owner, the fire brigade, or another body. There is also a batch analysis component, such as a monthly home health report.

Scratching the surface of the IoT

IoT scenarios are incredibly complex and data engineers who may have spent their careers in business data are less familiar with IoT architecture and systems. We hope this introduction will encourage interested data engineers to learn more about this fascinating and rapidly evolving specialization.

Notes

Data Mesh

The data mesh is a recent response to sprawling monolithic data platforms such as centralized data lakes and data warehouses and to the “big data divide” where the landscape is divided into operational and analytical data. The data mesh attempts to reverse the challenges of a centralized data architecture by applying domain-driven design concepts (commonly used in software architectures) to data architecture. Since data networks have been attracting a lot of attention lately since, keep that in mind.

A big part of the data network is decentralization, as Zhamak Dehghani pointed out in his seminal article on the subject:

To decentralize the monolithic data platform, we need to invert the way we think about data, its location and ownership. Rather than sending domain data to a central proprietary platform or data lake, domains should host and deliver their domain records in an easy-to-use manner.

Dehghani then identified four key components of the data network:

- Domain-oriented decentralized architecture and data ownership
- Data as a product
- Self-service data infrastructure as a platform
- Federated Computational Governance

Other Data Architecture Examples

data architectures have countless other variations such as data structure, data hub, scaled architecture, metadata-first architecture, event-driven architecture, live data stack and many more. And new architectures will continue to emerge as practices become established and mature and tools are simplified and improved. We have focused on a handful of the most critical data architecture patterns that are either very well established, rapidly evolving, or both.

As a data engineer, pay attention to how new architectures can help your business. Keep up to date with new developments by developing an elevated level of awareness of developments in the data engineering ecosystem. Keep an open mind and do not get emotionally attached to just one approach. Once you identify the potential value, deepen your knowledge, and make concrete decisions. When done right, minor adjustments or major overhauls to your data architecture can have a positive business impact. The

Data architecture is not designed in a vacuum. Larger companies can still employ data architects, but those architects need to be well versed and up to date with the state of the art and data. Gone are the days of ivory tower data architecture. In the past, architecture was orthogonal to engineering. We expect this distinction to fade as data engineering and engineering in general evolve rapidly and become more agile, with the separation between engineering and the architecture narrowing.

Ideally, a data engineer works with a dedicated data architect. However, if a company is small or has a low level of data maturity, a data engineer could work as an architect. Since data architecture is the background of the data engineering lifecycle, a data engineer needs to understand “good” architecture and the diverse types of data architecture.

2.1.6 Introduction to Data Wrangling

Data Wrangling turns messy, bad data into clean, useful data. This is a batch transformation process. Data conflicts have long been a major source of annoyance

Notes

and job security for data engineers. Suppose developers receive EDI data related to transactions and invoices, a combination of structured data and text, from a partner company. The typical process for disputing this data is to first attempt to ingest it.Often the data is so flawed that extensive text pre-processing is required. Developers can capture the data as a single text field table - an entire row is captured as a single field. Then the developers start writing queries to analyse and slice the data. Over time, they discover data anomalies and edge cases. Finally, they get the data in approximate form. Only then can the downstream transformation process begin.

Data dispute resolution tools are designed to simplify significant parts of this process. These tools often scare off data engineers because they claim it is not code, which sounds clunky. We prefer to think of data dispute tools as integrated development environments (IDEs) for bad data. In practice, data engineers spend too much time analysing ugly data; Automation tools allow data engineers to focus time on more interesting tasks. Dispute resolution tools can also allow engineers to offload some of the analysis and recording work to analysts.

Graphical data dispute tools typically present a sample of data in a visual interface, with derived types, statistics including distributions, outliers, outliers, and null values. Users can then add processing steps to troubleshoot data issues. A step can contain instructions on how to deal with misspelled data, how to split a text field into multiple parts, or how to link to a lookup table.

Users can complete the steps for a complete record when the whole task is completed. The job is typically sent to a scalable data processing system such as Spark for large datasets. After the job runs, errors and unhandled exceptions are returned. The user can further refine the recipe to deal with these outliers.

We strongly encourage both novice and experienced engineers to experiment with combat tools. Big cloud providers sell their version of data management tools and there are plenty of third-party options available. Data engineers may find that these tools significantly streamline certain parts of their work. From an organisational perspective, data engineering teams should consider training specialists on data disputes when they often come from new and confusing data sources.

Data Wrangling Process

Without data disputes, there is a risk that we may overlook issues in the data that may have important consequences for future analysis. The first crucial step in data wrangling is to read data from a source file into data tools such as the Python data framework and discover its granularity.

Data Extraction and Reading Stage

Understanding the form and granularity of a table gives us an idea of what a row in a data table represents. This helps us determine if the granularity is mixed, requires aggregation, or weights are required. After looking at the granularity of your dataset, you should have answers to the following questions.

- What does a record represent? Clarity about this will help you successfully conduct a data analysis and present your results.
- Do all records in a table capture granularity at the same level? Sometimes a table contains additional summary rows at a different granularity and you want to use only the rows that have the right level of detail.

Notes

- If the data is aggregated, how was the aggregation performed? Sum and average are common types of aggregation. With averaged data, the variability of the measurements is usually lower and the correlations often appear stronger.
- What kind of aggregations could you perform on the data? Aggregations can be useful or necessary to combine one table of data with another.

Knowing the granularity of your table is a first step in cleaning up your data and will show you how to analyse the data.

Discussion techniques during the Extract and Read Data step help us to put data from a source file into a data frame and understand its structure. Once we have a data scaffold, further discussions are needed to assess and improve quality and prepare the data for analysis.

Extracted Data Analysis Stage

The data in the data frame is also subject to quality checks to detect problems in the data. To find erroneous and missing values, we can take many approaches.

- Check summary statistics, distributions, and value counts. We check the quality of your data through visualizations and summary statistics. A table showing the number of unique values in a feature can reveal unexpected encodings and imbalanced distributions when an option occurs infrequently. Percentiles can be helpful in revealing the proportion of values with unusually high (or low) values.
- Logical expressions can identify records with out-of-range values or out-of-control relationships. By simply calculating the number of records that failed quality control, you can quickly see the extent of the problem.
- Search the entire dataset for records with problematic values in a particular attribute. Sometimes an entire dataset is skewed, for example, when a comma is missing from a CSV-formatted file. Or the dataset may represent an unusual situation (e.g., ranches included in the home sales data) and you need to decide whether or not to include that in your analysis.
- Consult an external source to find out if there is a reason for the error: The biggest success factor is curiosity about your data. Look for clues that can tell you about the quality of your data. The more evidence you find, the more confident you will be in your findings. And when you spot problems, dig deeper. Try to understand and explain unusual phenomena. A good understanding of your data will help you assess whether an issue you have identified is minor and can be ignored or fixed, or whether it severely limits the usefulness of your data. This curiosity mindset is closely related to exploratory data analysis.

2.2 Overview of RDBMS and No SQL

Relational Database Management Systems (RDBMS) are the main drivers of information systems around the world. They allow a large number of users to access, create, edit, and edit data quickly and simultaneously without affecting other users. They also allow developers to write useful applications to access their resources and give administrators the skills they need to manage, protect, and optimize the company's data assets.

2.2.1 Introduction to RDBMS

RDBMS is an acronym for Relational Database Management Systems. It is an application that will allow us to create, delete and update relational databases. A

Notes

relational database is a database system that stores and retrieves data in the form of rows and columns in a table format. It is a small subset of the DBMS developed by E.F. Codd was developed. Important DBMS such as SQL, MySQL and ORACLE are based on relational DBMS concepts.

The fact that the values of each table are related to each other is the cornerstone of the relational DBMS.It is able to process substantial amounts of data and easily simulate queries.

What exactly is a table?

The data in a relational database management system (RDBMS) It is stored in database objects called tables. This table, which has multiple columns and rows, is a collection of related data items.

In a relational database, a table is the most popular and basic form of data storage. The following program demonstrates a customer table.

Table: Customers

ID	Name	Age	Address	Salary
1001	Amit	33	Gorakhpur	2000
1002	Tanvi	33	Delhi	1000
1003	Meha	35	Noida	1200
1004	Digvijay	38	Haldwani	2500
1005	Neha	42	Almora	1800
1006	Atharv	25	Greater Noida	800
1007	Priyanka	28	Delhi	2000

What exactly is a field?

Each table is divided into fields representing smaller entities. ID, Name, Age, Address and Salary are the fields in the customer table.

A field is a table column used to store specific information about each entry in the database.

What is record or a row?

Each individual element in a table is referred to as a record, often referred to as a row of data. For example, in the Customers table, there are seven records. Below is a single row of data or records from the Customers table.

1001	Amit	33	Gorakhpur	2000

What exactly is a column?

In a table, a column is a vertical element that contains all the information about a particular field. For example,

Address is a column in the customer database that represents the location description and may look like this.

Address

Gorakhpur

Notes

Delhi

Noida

Haldwani

Almora

Greater Noida

Delhi

What is the meaning of a NULL value?

In a table, a NULL value is a value in a field that is empty, meaning that a field with a NULL value has no value.

It is important to understand the difference between a NULL value and a null value or a field containing spaces. A field with a NULL value was left blank throughout the record generation process.

Relational database management systems simulate the following functions to maintain data integrity:

Entity Integrity: There must be no duplicate records in the database table.

Referential integrity: Only rows that are not required by other tables can be deleted from the tables. Otherwise, there may be discrepancies in the data.

Custom Integrity: Users set confidentiality and access rules.

Domain Integrity: Columns in database tables are contained within some structural boundaries based on default values, data types, or ranges.

Functions

- In a database file, data must be stored in tabular form, i.e., in the form of rows and columns.
- Each row in the table is called a record/tuple. The cardinality of the table is the collection of such records.
- Each column in the table is called an attribute/field. The array of the table is a collection of similar columns.
- There must be no duplicate data records in the DB table. Using a candidate key avoids data duplication. The candidate key is a set of properties that must be present in order for each record to be uniquely identified.
- The use of foreign keys links the tables together.
- Database tables are also nullable, which means that if the values of any of the items in the table are unfilled or missing, the value is converted to a NULL value that is different from null. (NOTE: The primary key is not nullable.)

Advantages

- • Ease of administration: each table can be modified independently without affecting the others.
- • Security: It is more secure as it has multiple levels of security. Access to shared data may be restricted.
- • Flexible: Data can be updated in a unique location without having to make changes to many files. Databases can

Notes

be easily expanded to accommodate additional records, increasing scalability. It also makes it easier to use SQL queries.

- ❖ Users: RDBMS can store many users in a client-side architecture.
- ❖ Allows easy storage and retrieval of substantial amounts of data.

Simple Data Handling:

- ❖ The relational architecture enables faster data retrieval.
- ❖ Redundancy or duplication of data is avoided through keys, indices, and normalization principles.
- ❖ Because RDBMS is based on ACID principles for data transactions, data consistency (Atomic Consistency Isolation Durability) is guaranteed.
- ❖ Fault Tolerance: Database replication enables concurrent access and supports system recovery in the event of a disaster such as a disaster. B. a power failure or a sudden shutdown.

Disadvantages

- ❖ Excessive cost and extensive hardware and software support: These systems require large budgets and configurations to work.
- ❖ Scalability: As more data is added, more servers, power and storage are required.
- ❖ Complexity: Copious amounts of data make it difficult to understand context and can affect performance.
- ❖ Structured Boundaries – Fields or columns in a relational database system are surrounded by multiple boundaries, which can lead to data loss.

2.2.2 Tools of RDBMS

RDBMS Tools is a collective term for tools, utilities, and wizards that you can use to perform database management tasks. Some database tools perform similar tasks, although no database tool performs all database administration tasks. The following sections explain which database tools can be used on specific operating systems and which tools are preferred for general database administration tasks. The database management tools include features that meet the needs of three diverse types of database professionals.

- Database developers want tools that help them write quality code the first time and easily maintain it.They value programming automation and collaboration tools so they can shorten development cycles without increasing risk.
- Database administrators use tools to monitor the health and performance of databases. They take on tasks ranging from diagnosing and fixing performance bottlenecks to implementing database schema changes.
- Database analysts pull data from multiple sources, integrate, cleanse, and then prepare the data for analysis. It is important for them to be able to collaborate on queries and datasets without depending on IT for access.

2.2.3 Role of RDBMS in Data Engineering

The RDBMS provides an interface between users and applications and the database, as well as administrative functions to manage data storage, access, and performance.

Notes

The relational database management system is the great-grandfather of permanent data warehouses. The computer industry in its early days used techniques that are now considered primitive data persistence.

In the 1970s, IBM scientist Edgar Codd developed the relational model, which was adopted by IBM, Oracle, Microsoft, and others. It is still widely used today and plays a key role in data engineering.

Relational databases, which make up most databases in use today (and most of these modules), were designed for transaction processing. They process the data companies need to stay in business. The data that the organisation needs is known and usually structured in a predictable and stable manner. In other words, we know what we need and how the data interacts with each other. Our needs may change over time, but the changes are gradual and, in most cases, changes to the computing system do not need to be made in a hurry.

2.2.4 Overview of NoSQL Database

A NoSQL database, which stands for “non-SQL” or “non-relational,” is a database that provides storage and retrieval of data. In addition to the tabular relationships used in relational databases, this data is modelled differently. These databases first appeared in the late 1960s but did not get the name NoSQL until the early 21st century.

NoSQL databases are increasingly used in real-time online applications and big data analytics. Not only SQL is a term that emphasises the fact that NoSQL systems can enable SQL-like query languages.

Simplicity of design, horizontal scalability across server clusters and tighter control over availability are all advantages of a NoSQL database. By default, the data structures used by NoSQL databases differ from those that use relational databases, which allows NoSQL to perform some operations faster.

The applicability of a NoSQL database is determined by the problem it is designed to answer. NoSQL database data structures are sometimes considered more flexible than relational database tables.

Many NoSQL databases make trade-offs between consistency and availability, performance, and partition tolerance. The use of low-level query languages, the lack of standardized interfaces and previous large investments in relational databases are barriers to wider adoption of NoSQL storage.

Although most NoSQL databases lack true ACID transactions (atomicity, consistency, isolation, and durability), some databases including MarkLogic, Aerospike, FairCom c-treeACE, Google Spanner (although technically a NewSQL database), Symas LMDB and OrientDB, the case made them a central part of their designs.

Most NoSQL databases support eventual consistency, which means that database updates are propagated to all nodes over time. As a result, data lookup may not return current data immediately, or data reads may be imprecise, a problem known as stale reads. Some NoSQL systems can also experience lost writes and other data corruption.

To prevent data loss, certain NoSQL systems provide ideas such as write-ahead logging. Data consistency is even more difficult to achieve when distributed transaction processing occurs across many databases. Both NoSQL and relational databases struggle with this. Even current relational databases do not allow referential integrity constraints between databases. For distributed transaction processing, few systems support the X/Open XA standards and ACID transactions.

Notes

Advantages of NoSQL

There are numerous advantages to working with NoSQL databases such as MongoDB and Cassandra. High scalability and availability are the main advantages.

High Scalability: For horizontal scaling, NoSQL databases use sharding. Sharding is the process of partitioning data and distributing it over numerous machines while maintaining the data's order. Horizontal scaling entails adding new machines to handle the data and vertical scaling is adding more resources to the present machine. Vertical scaling is difficult to implement, whereas horizontal scaling is straightforward. MongoDB, Cassandra, and other horizontal scaling databases are examples. Because of its scalability, NoSQL can handle copious amounts of data. As the amount of data grows, NoSQL scales to handle it efficiently.

High Availability: The automatic replication feature in NoSQL databases makes them universally available as the data replicates itself to a previous consistent state in case of failure.

NoSQL's Disadvantage:

NoSQL has the following disadvantages.

NoSQL databases have a very narrow focus as they are primarily designed for storage and offer extraordinarily little functionality. When it comes to transaction management, relational databases outperform NoSQL databases.

Open-Source: NoSQL is a free-to-use database. There is no reliable standard for NoSQL yet. In other words: there is a good chance that two database systems are not equal.

The goal of big data tools is to make managing substantial amounts of data as easy as possible. But it is not that easy. Data management in a NoSQL database is much more complicated than in a relational database. NoSQL, in particular, is notorious for being difficult to set up and much more difficult to manage on a day-to-day basis.

GUI mode database access tools are not flexible: GUI mode database access tools are not flexible in the market. Backup

– For some NoSQL databases like MongoDB, backup is a big mistake.MongoDB does not include a method to back up data consistently.

Large Document Size – Data is stored in JSON format in some database systems like MongoDB and CouchDB. This means the documents are quite large (due to BigData, network bandwidth and speed) and verbose key names actually hurt the document size by increasing it.

Difference Between RDBMS and NoSQL Databases

Although there are numerous differences between relational database management systems (RDBMS) and NoSQL databases, one of the most significant differences is the way the data in the database is modelled. In this section, we will walk through an example of how to represent the same data in a relational and a NoSQL database. Then we will go over some of the other key differences between relational and NoSQL databases.

Types of NoSQL database:

The following are examples of NoSQL databases and the database systems that belong into that category:

Notes

1. MongoDB is a document-based database that uses NoSQL technology.
2. Key value store: Memcached, Redis, Coherence
3. Tabular: Hbase, Big Table, Accumulo
4. Document Based: MongoDB, CouchDB, Cloudant

2.2.5 Applications of NoSQL Database

Nonrelational databases provide four main storage categories: (1) document, (2) key-value pairs, (3) column-wide and (4) chart. Let us look at these in turn.

So-called "document" databases are the most free-form non-relational offering since their documents store potentially hierarchical collections of objects with properties. Returning to our earlier example, while a relational database stores data about a single camper in a series of related tables, a document database might store all the details of that camper in a single document. Said document could contain "personal information" that includes the properties we saw in before (e.g., Name, Age and Favourite Food) while other objects e.g., Attendance, Cabin Assignments, Activities etc. You could keep other properties.

The document database format is therefore suitable for storing less structured information in a freer form. It is a great starting option for any new development project when you are not sure what the data models will look like.Non-relational databases store everything and allow schemas to easily evolve over time without requiring much structure.

As the name suggests, "key-value databases" store key-value pairs, or KVPs for short. A CIP can be any simple information dyad; That is, one element serves as a key associated with another element as a value. While the document database version could store the information for a single child in a single document, it would be much more complicated in a key-value database.

But you would be cramming a key-value NoSQL database into a use case for which it is, at best, unsuitable. Such databases are intended for simpler purposes such as caching temporary data for web sessions, collecting simple telemetry points from devices in the field, etc. If you are wondering why anyone would accept such a limited data model, the answer is: its main benefit. : stunning performance. Hashing and other algorithms allow reading and writing of simple KVP structures much faster than many alternatives.

"Column-wide stores," sometimes called "column databases," sit in an interesting place between document databases and relational databases. They are not document databases in the sense that they do not support indeterminate hierarchies of objects with properties, but they are not relational databases in the sense that they do not require all rows in a table to have the same columns. And despite their name, these databases can often be row-oriented rather than column-oriented.

A key factor in understanding the broad columnar data model is a key space, which is a collection of various quasi-table structures, typically referred to as "column families". Each column family can have any number of rows, but each row must (1) have a unique row key that distinguishes it from all others and (2) can have any number of columns of information, each typically containing a KVP and a Timestamp indicating the last update.

For example, our summer camp database might have multiple key areas for storing the same information as before, while the specific key area for personal information might contain a row for each child with columns for name, age, favourite food and so on. The challenge of storing such data in a column store is finding the right approach to splitting the data into different key spaces and choosing values that truly individualize the rows.

Notes

Finally, we come to “graph databases,” which are most easily understood in terms of their nodes and relationships. The nodes store the data values, often as small “mini documents” such as document databases or KVP “bags”, etc. The relationships connect the nodes and are thus quasi first-class citizens in the diagram data model. While a relational database brings together diverse types of data in tables that share unique IDs, a graph database links one node to another through explicit relationships of diverse types.

It would be trivially easy to store the sample data we are discussing in a chart database. Each camper’s personal data is a single node, with all related information stored in other nodes linked by appropriate relationships. In addition, it would also be easy to store relationships between children, keep track of who is friends with whom, which campers do not have a good relationship with which other campers, etc.

The structure of graph databases makes them ideal for many use cases that cannot be easily handled by other types of databases. Examples include organisational charts, social networks, network gear and routing diagrams, street, and other maps, driving directions, etc.

Popular uses of NoSQL databases are:

User Profiles – Authentication tools and LDAP are good for authentication and authorization, but data such as rewards, criminal records, promotions, phone numbers and addresses are added daily. Other databases cannot accommodate such changing data. We can use dynamic NoSQL documents to store such data in the document over time.

Product and catalogue data: Many new products are added every day in e-commerce companies or chemical companies. Every time a new product is added, it is not easy to quickly change the scheme. In these scenarios, using a NoSQL database is easier than using any other traditional database.

Metadata: We often need metadata that describes our data. In such scenarios, a graph-based database is a good option, but we can also use the document database for these applications.

Contents: MongoDB is primarily a document database. It is great for serving text and HTML documents. In addition, it offers precise control over content storage and indexing.

2.2.6 Important Commands in NoSQL

MongoDB, the main NoSQL database, is stored in the form of documents. The document is a collection of key-value pairs. The key is also called an attribute.

Documents have a dynamic schema and documents in the same collection can vary in the set of fields. We will now review MongoDB CRUD operations and commands.

The Create Operation

The create process adds the new document to the collection. If no collection exists, MongoDB creates a new collection and inserts a document into it. MongoDB provides the following methods for inserting a document into the database:

- db.collection.insertOne();
- db.collection.insertMany();

MongoDB insert operation targets unique collections. Also, Mongo preserves document-level atomicity.

Notes

The insertMany() method can insert multiple documents into the collection at once. We need to pass an array of documents to the method.

In MongoDB, each document requires an _id field that uniquely identifies the document that acts as the primary key. If a user does not include the _id field during the insert process, MongoDB automatically generates and inserts an ID for each document.

The following is the list of methods that can also be used to insert documents into the collection:

- db.collection.update();
- db.collection.updateOne();
- db.collection.updateMany();
- db.collection.findAndModify();
- db.collection.findOneAndUpdate();
- db.collection.findOneAndReplace();
- db.collection.save();
- db.collection.bulkWrite();

The Read Operation

The read operation retrieves documents or document data in the collection. To get all documents in a specific collection, pass an empty document as a filter. We need to pass the query filter parameter to apply our document selection criteria.

- db. collection.find({})

This operation will return all the documents from the collection.

2.2.7 Insight of Various Database Tools

Relational databases have been commercially available for more than three decades. Some of the more mature and popular commercial products include:

- Oracle Database from Oracle Corporation
- SQL Server from Microsoft
- DB2 Universal Database from IBM

All of these database servers do the same thing, although some are better suited to running exceptionally large or immensely powerful databases. Others are better at handling exceptionally large objects or files or XML documents etc. Also, all of these servers are fairly compatible with the latest ANSI SQL standard.

In addition to commercial database servers, there has been a great deal of activity in the open-source community over the past two decades with the aim of creating a viable alternative. Two of the most widely used open-source database servers are PostgreSQL and MySQL.

Although relational databases are still and will continue to be, widely used for some time, new database technologies have emerged to meet the needs of companies like Amazon and Google. These technologies include Hadoop, Spark, NoSQL and NewSQL, which are scalable, distributed systems typically deployed on simple server clusters.

Because organisations often store data using multiple technologies, there is a need to separate SQL from a specific database server and provide a service that can span multiple databases. For example, a report may need to collect data stored in Oracle,

Notes

Hadoop, JSON files, CSV files and Unix log files. A new generation of tools have been built to meet this type of challenge, such as Apache Drill, which is an open-source query engine that allows users to write queries that can access data stored in most any database or filesystem.

Summary

- Data Engineering Basics for Everyone provides an accessible overview of the fundamental concepts and processes involved in data engineering.
- It aims to demystify data engineering and make it understandable for a broader audience, including those without a technical background.
- Data engineering is the foundation of data-driven decision-making and analytics. It involves collecting, storing, and processing data from various sources to make it ready for analysis by data scientists and other stakeholders. These modules emphasises the importance of data quality, data integrity and data security in the data engineering process.
- Data processing and data transformation are key components of data engineering. The modules delve into data pipelines and ETL (Extract, Transform, Load) processes, which are crucial for transforming raw data into a usable and valuable form. It also introduces concepts like data lakes and data marts.
- Furthermore, data architecture is explored to understand the design and organisation of data systems. This includes discussions on data modelling, data schema and data integration.
- Overall, Data Engineering Basics for Everyone serves as a beginner-friendly guide, empowering readers with the knowledge necessary to navigate the world of data engineering and gain insights from the wealth of data available in today's digital landscape.

Glossary

- Data Engineering: The process of designing, building, and maintaining systems that collect, store, process and transform data into a usable and valuable form for analysis and decision-making.
- Data Storage: The technology and processes involved in storing and organizing data, such as relational databases, NoSQL databases, data warehouses and data lakes.
- Relational Database: A type of database that stores data in tables with predefined relationships between them, using SQL (Structured Query Language) for querying.
- NoSQL Database: A type of database that does not follow the traditional relational model and is suitable for handling unstructured or semi-structured data.
- Data Warehousing: A centralized repository that integrates data from multiple sources to support business intelligence and analytics activities.

Check Your Understanding

1. What is Data Engineering?
   a) The process of analysing data to gain insights
   b) The process of designing, building, and maintaining data systems
   c) The process of visualizing data for decision-making
   d) The process of collecting data from various sources

Notes

2. Which technology is commonly used for storing structured data with predefined relationships?
   a) NoSQL database
   b) Data warehouse
   c) Data lake
   d) Relational database
3. What is the purpose of a data warehouse?
   a) To store unstructured data
   b) To integrate data from various sources for analytics
   c) To store raw and unprocessed data
   d) To facilitate real-time data processing
4. ETL stands for:
   a) Extract, Transform, Load
   b) Extract, Transfer, Load
   c) Extract, Translate, Load
   d) Extract, Transmit, Load
5. What is the primary goal of data processing in data engineering?
   a) To collect data from various sources
   b) To store data in a data lake
   c) To transform raw data into a usable form for analysis
   d) To visualize data for reporting
6. Which term refers to a series of data processing steps that move data from its source to a destination?
   a) Data Integration
   b) Data Pipeline
   c) Data Transformation
   d) Data Storage
7. What does “Data Quality” refer to in data engineering?
   a) The accuracy and consistency of data over its entire lifecycle
   b) The size of the dataset
   c) The number of data sources used
   d) The complexity of data transformations
8. What is the primary purpose of a data lake?
   a) To store processed and structured data
   b) To store raw and unprocessed data in its native format
   c) To serve as a centralized data repository for a specific business function
   d) To facilitate real-time data analysis
9. Data governance is a set of policies and procedures to ensure:
   a) The security of data
   b) The visualization of data
   c) The responsible and ethical use of data
   d) The integration of data from various sources
10. What is the role of a Data Scientist in data engineering?
    a) To design data pipelines
    b) To collect data from various sources
    c) To build and maintain databases
    d) To analyse and interpret data for insights
11. Which term refers to the blueprint that defines the structure of a database or data table?

Notes

a) Data Governance b) Data Schema

c) Data Catalog d) Data Wrangling

12. What is the process of cleaning, transforming, and enriching raw data to make it suitable for analysis?

a) Data Catalog b) Data Governance

c) Data Visualization d) Data Wrangling

13. Which technology is suitable for handling unstructured or semi-structured data?

a) Relational database b) NoSQL database

c) Data warehouse d) Data lake

14. Data integration is the process of:

a) Collecting data from various sources

b) Transforming raw data into a usable form

c) Combining data from various sources into a unified view

d) Storing data in a data warehouse

15. Which type of database is optimized for real-time data processing and analytics?

a) Data warehouse b) Data lake

c) Relational database d) NoSQL database

16. What is the primary goal of data visualization?

a) To design data pipelines b) To store and organise data

c) To clean and transform data

d) To represent data in a visual format for better understanding

17. Which technology provides a centralized repository with metadata and information about available datasets?

a) Data Warehouse b) Data Lake

c) Data Catalog d) Data Mart

18. The acronym “ETL” stands for:

a) Extract, Transform, Load b) Extract, Transfer, Load

c) Extract, Translate, Load d) Extract, Transmit, Load

19. What is the main advantage of using a data lake?

a) Fast real-time data processing

b) Integration of data from various sources

c) Cost-effective storage of raw and unprocessed data

d) Efficient data querying using SQL

20. Data governance involves:

a) Collecting and storing data

b) Transforming raw data into a usable form

c) Ensuring the responsible and ethical use of data

Notes

d) Building data pipelines for data processing

Exercise

1. Explain source of gathering data.
2. Write a short note on introduction to data wrangling.
3. Explain the role of RDBMS in data engineering.
4. What are the important commands in NoSQL?

Learning Activities

1. Discuss Python Basics for Data Science with an example.
2. Describe Data Structures in Python of any company.

Check Your Understanding- Answers

1. b	2. d	3. b	4. a
5. c	6. b	7. a	8. b
9. c	10. d	11. b	12. d
13. b	14. c	15. d	16. d
17. c	18. a	19. c	20. c

Further Readings and Bibliography

1. "Introduction to Machine Learning" by Ethem Alpaydin
2. "Pattern Recognition and Machine Learning" by Christopher Bishop
3. "Machine Learning: A Probabilistic Perspective" by Kevin P. Murphy
4. "Machine Learning Yearning" by Andrew Ng
5. "Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow" by Aurélien Géron
6. "An Introduction to Statistical Learning" by Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani
7. "Deep Learning" by Ian Goodfellow, Yoshua Bengio and Aaron Courville
8. "Machine Learning" by Tom Mitchell, McGraw-Hill Education, 1st Edition.
9. "Artificial Intelligence: A Modern Approach" by Stuart Russell and Peter Norvig, Pearson, 3rd Edition
10. "Deep Learning" by Ian Goodfellow, Yoshua Bengio and Aaron Courville, Publication: The MIT Press, 1st Edition
11. "Natural Language Processing with Python" by Steven Bird, Ewan Klein and Edward Loper, O'Reilly Media, 1st Edition

Module - III: Analysing Data with Python

Learning Objectives

At the end of this module, you will be able to:

- Analyse importing data sets
- Understand the dataset
- Learn python package for data science
- Summarise importing and exporting data in python
- Describe basic insights from datasets
- Analyse cleaning and preparing the data
- Know identify and handle missing values
- Learn data formatting
- Describe data normalisation sets
- Analyse summarising the data frame
- Know descriptive statistics
- Learn basic of grouping
- Describe ANOVA
- Analyse correlation
- Summarise model development
- Know simple and multiple linear regression
- Learn model evaluation using visualisation
- Describe polynomial regression and pipelines
- Analyse R-squared and MSE for In-sample evaluation
- Know prediction and decision making
- Learn model evaluation
- Analyse over-fitting, under-fitting, and model selection
- Describe ridge regression
- Learn grid search
- Know model refinement

Introduction

Python has become one of the most popular standard languages and is a complete suite for data science operations. Python offers many libraries like NumPy, Pandas, SciPy, Scikit-Learn, Matplotlib, Seaborn and Plotly. These libraries provide a comprehensive data analytics ecosystem used by data scientists, data scientists and business analysts. Python also offers other features such as flexibility, ease of learning, faster development, a large active community, and the ability to work on complex numerical, scientific and research applications. All these features make it the top choice for data analysis.

The 21st century is the information age. We live in the information age, which means every aspect of our daily lives generates data. In addition, business processes,

Notes

government processes and social media posts generate massive amounts of data. This data accumulates day by day as economic, governmental, scientific, technical, health, social, climate and environmental data is continuously generated. In all these areas of the decision-making process, needs a structured, generalised, efficient, and flexible system for the analytical and scientific process to obtain information about the generated data.

In today's intelligent world, data analysis drives effective decision-making in business and government. Data analysis is the process of examining, pre-processing, exploring, describing, and visualising a specific data set. The main goal of the data analysis process is to find the information needed for decision making. Data analysis offers many approaches, tools, and techniques, all applicable to different fields such as economics, social sciences, and fundamental sciences.

3.1 Importing Data Sets

Datasets in Python are of immense importance and are used to process copious amounts of data. These records are somewhat similar to the packages in Python 3.6 and later. Python datasets consist of a dataset object, which in turn contains metadata as part of the dataset. Queries on these records can involve record objects to return the requested index based on rows and columns. When data is first loaded, the dataset object is displayed, which also contains metadata, which consists of other valuable information.

3.1.1 Understanding the Dataset

Datasets in Python are of immense importance and are used to process copious amounts of data. These records are somewhat similar to the packages in Python 3.6 and later. Python datasets consist of a dataset object, which in turn contains metadata as part of the dataset. Queries on these records can involve record objects to return the requested index based on rows and columns. When data is first loaded, the dataset object is displayed, which also contains metadata, which consists of other valuable information.

3.1.2 Python Package for Data Science

The basic Python package for data science and data analysis is:

- NumPy: This is an abbreviation for Numeric Python. It is the most powerful scientific library available in Python for handling multidimensional arrays, matrices, and methods for efficiently computing mathematics.
- SciPy: It is also a powerful scientific library for performing scientific, mathematical, and engineering operations.
- Pandas: This is a data exploration and manipulation library that provides tabular data structures such as DataFrames and various methods for analysing and manipulating the data.
- Scikit-learn stands for "Scientific Machine Learning Toolkit". It is a machine learning library that offers a variety of supervised and unsupervised algorithms such as regression, classification, dimensionality reduction, cluster analysis and anomaly detection.
- Matplotlib: This is the main data visualization library and the base library for all other Python visualization libraries. It offers 2D and 3D charts, graphs, graphs, and numbers for data mining. Works on NumPy and SciPy.

Notes

- Seaborn: Based on Matplotlib, it provides high-level plots that are easy to draw, interactive and better structured.
- Plotly: Plotly is a data visualization library. It offers high-quality interactive charts such as scatterplots, line charts, bar charts, histograms, boxplots, heatmaps and secondary charts.

3.1.3 Importing and Exporting Data in Python

Reading and providing data (often referred to as data import) is an essential first step in data analysis. The term parsing is also sometimes used to describe the loading of text data and its interpretation into tables and various data types. We will focus on input and output using pandas, although there are many tools in other libraries to help you read and write data in different formats.

Input and output fall into a few main categories: reading text files and other more efficient formats on disk, loading data from databases, and interacting with network sources such as web APIs.

Reading and Writing Data in Text Format

Pandas provides a number of functions for reading tabular data as a DataFrame object. The following list summarises some of them:

Function	Description
read_csv	Load delimited data from a file, URL, or file-like object; use comma as default delimiter
read_fwf	Read data in fixed-width column format (i.e., no delimiters)
read_clipboard	Variation of read_csv that reads data from the clipboard; useful for converting tables from web pages
read_excel	Read tabular data from an Excel XLS or XLSX file
read_hdf	Read HDF5 files written by pandas
read_html	Read all tables found in the given HTML document
read_json	Read data from a JSON (JavaScript Object Notation) string representation, file, URL, or file-like object
read_feather	Read the Feather binary file format
read_orc	Read the Apache ORC binary file format
read_parquet	Read the Apache Parquet binary file format
read_pickle	Read an object stored by pandas using the Python pickle format
read_sas	Read a SAS dataset stored in one of the SAS system's custom storage formats
read_spss	Read a data file created by SPSS
read_sql	Read the results of a SQL query (using SQLAlchemy)
read_sql_table	Read a whole SQL table (using SQLAlchemy); equivalent to using a query that selects everything in that table using read_sql
read_stata	Read a dataset from Stata file format
read_xml	Read a table of data from an XML file

Due to real-world data clutter, some data-loading functions (notably pandas.read_csv) have accumulated an extensive list of optional arguments over time. It is normal to feel overwhelmed by the number of different parameters (pandas.read_csv has around

Notes

50). There are many examples of this in the Pandas online documentation. So, if you are having trouble reading a specific file, there might be an example similar enough to help you find the right settings.

Writing Data to Text Format

Data can also be exported in a comma-delimited format. Using the DataFrame method to_csv we can write the data to a comma separated file. Of course, other separators can also be used. Missing values appear as empty strings in the output. If no other options are specified, the row and column labels are saved.

JSON Data

JSON (short for JavaScript Object Notation) has become one of the standard formats for data transfer over an HTTP request between web browsers and other applications. It is a much freer data format than a tabular text format like CSV.

JSON is almost valid Python code, apart from its null value and some other nuances (e.g., prohibition of commas at the end of lists). The basic types are objects (dictionaries), arrays (lists), strings, numbers, logical values, and null values. All keys of an object must be strings. There are several Python libraries for reading and writing JSON data. We can use JSON here because it is built into the Python standard library. To convert a JSON string in Python, we use the json.loads function. json.dumps, on the other hand, converts the Python object back to JSON.

XML and HTML: Web Scraping

- Python has many libraries for reading and writing data in the ubiquitous HTML and XML formats. Examples include lxml, Beautiful Soup and html5lib. While lxml is much faster, other libraries can handle malformed HTML or XML files better.
- Pandas has a built-in feature, Pandas.read_html, which uses all of these libraries to automatically parse tables from HTML files as DataFrame objects.
- The pandas.read_html function has several options, but by default it searches and attempts to parse all table data contained within tags. The result is a list of DataFrame objects.

Binary Data Formats

- An effortless way to store (or serialise) data in binary format is to use Python's built-in Pickle module.
- All pandas objects have a to_pickle method that saves data to disk in pickle format.
- Pickle files are normally readable only in Python. We can read any "kicked" object stored in a file directly with the built-in pickle or even more simply with pandas.read_pickle.

Reading Microsoft Excel Files

Pandas also supports reading table data stored in Excel 2003 files (and later) with both pandas.ExcelFile class or pandas.read_excel function. Internally, these tools use the xlrd and openpyxl add-ons to read old XLS files and new XLSX files, respectively.

Using HDF5 Format

HDF5 is a respected file format developed for archiving copious amounts of scientific data tables. It is available as a C library and has interfaces in many other languages

Notes

including Java, Julia, MATLAB, and Python. “HDF” in HDF5 stands for hierarchical data format. Each HDF5 file can store multiple datasets and supporting metadata. Compared to simpler formats, HDF5 supports on-the-fly compression with multiple compression modes, allowing them to archive data with repeating patterns more efficiently.HDF5 can be an excellent choice for working with datasets that do not fit in memory because it can efficiently read and write small blocks of much larger arrays.

To get started with HDF5 and pandas, the PyTables package is a prerequisite.

Interacting with Web APIs

Many websites have public APIs that provide data feeds in JSON or some other format. There are many ways to access these APIs from Python. One of the recommended methods is the requirement group.

Import/ Export operations with Databases

In a corporate environment, a lot of data may not be stored in text or Excel files.SQL-based relational databases (such as SQL Server, PostgreSQL, and MySQL) are widely used and many alternative databases are popular. Database selection is typically based on application performance, data integrity and scalability requirements.

Pandas has some features that make it easier to load SQL query results into a DataFrame. Most Python SQL drivers return a list of tuples when selecting data from a table. We pass a list of tuples to the DataFrame constructor.Project

SQLAlchemy is a popular Python SQL toolkit that examines many common differences between SQL databases. Pandas has a read_sql function that allows us to easily read data from a generic SQLAlchemy connection.

3.1.4 Basic Insights from Datasets

During data analysis and modelling, a lot of time is spent preparing the data: loading, cleaning, transforming, and organising. It is often reported that these tasks take up 80% or more of an analyst’s time. Sometimes the way data is stored in files or databases is not suitable for a particular business. Many researchers choose to process data ad hoc from one module to another, using a general programming language such as Python, Perl, R or Java, or Unix word processing tools such as sed or awk. Luckily, pandas, along with built-in Python functions, provide a flexible and fast set of high-quality tools for manipulating data into any shape you want.

Exploring Data

We explore data by performing exploratory data analysis (EDA). EDA is the most critical and important part of the data analysis process. EDA offers the following advantages:

- Provides a first look at the data and its context.
- Quickly gain insights and identify potential drivers from predictive analytics data. Find questions and questions that can be answered to make a decision.
- Assess data quality and help us create a roadmap for data cleansing and preprocessing.
- Find missing values, outliers, and the importance of features for the analysis.
- EDA uses descriptive statistics and visualization techniques for data mining.

Notes

In EDA, the first step is to read the record. We can read the data set with pandas. The Panda library offers many ways of reading data. It can read files in different formats like CSV, Excel, JSON, Parquet, HTML and Pickle. These options were discussed in the previous subtopic. After reading the data, we can examine the data. This initial exploration will help us understand the data and better understand the area. We can use the data.info() function to inspect the schema, columns, rows, data types and missing values of a table in a DataFrame.

With the descriptive function, we can display the descriptive statistics of the data. This function describes numeric objects by count, mean, standard deviation, min-max and first, second and third quartile.

3.2 Cleaning and Preparing the Data

Data analysts and scientists spend most of their time cleaning data and pre-processing unstructured datasets. Although this activity is less talked about, it is one of the most frequently performed tasks and one of the most important skills for any data professional. Mastering data cleaning skills is essential for any aspiring data analyst. Data cleansing and pre-processing is the process of identifying, updating, and removing corrupted or invalid data. The cleaning and pre-processing result in high-quality data for error-free and reliable analysis. High quality data can beat complex algorithms and outperform simple and less complex algorithms. High quality in this context means accurate, complete, and consistent data. Data cleaning is a set of actions such as handling missing values, removing outliers, coding functions, scaling, transforming, and splitting.

Filtering Data to Weed out the Noise

In the last two decades, the data volume of companies and authorities has increased due to digitization. It has also introduced more consistency, errors, and missing values. Data filtering is responsible for fixing these issues and optimising for management, reporting, and forecasting. The filtering process increases the accuracy, usefulness, completeness, consistency, and quality of data by dealing with dirty, messy, or coarse datasets. This is an especially crucial step in any type of data management as it can determine a company's competitive advantage. Data analysts must master the ability to filter data. Diverse types of data require diverse types of processing. Therefore, a systematic approach to data filtering should be chosen. Data can be filtered by columns or rows. Let us go through them one by one.

Column-wise Filtration

We can filter the columns using the filter() method. Cut []. filter() selects columns when passed as a list of columns.

We can also filter columns by truncating them. When slicing, a single column does not need a list, but if we filter multiple columns, they need to be in the list. The single-column output is a series of pandas. If we want the output to be a DataFrame, we just need to add a column name to the list.

Row-wise Filtration

We can filter the data using indexes, bins, and conditions. With indexes, we need to pass the index of the dataset, while with slicing, we need to pass the slicing range.

Conditional filtering requires some conditions to be enclosed in square brackets [ ] or square brackets ( ).For a single value we use the condition == (double equals), while for multiple values we use the isin() function and pass a list of values.

Notes

3.2.1 Identify and Handle Missing Values

Missing values are values that do not appear in the data. Missing values can occur due to human error, confidentiality issues, or a respondent's omission to provide values. This is the most widespread problem in data science and the first step in data preprocessing. Missing values affect the performance of the machine learning model. Missing values can be handled as follows:

- ❖ Delete the missing value records.
- ❖ Enter the missing value manually.
- ❖ Fill in the missing values using central measures of tendency such as mean, median and mode. The mean is used to assign a numeric feature, the median is used to assign an ordinal feature and the mode or highest value is used to assign a categorical feature.
- ❖ Enter the value using machine learning models such as regression, decision trees and ANN.

It is important to understand that in some cases missing values do not affect the data. For example, driver's license numbers, social security numbers, or other unique identifiers do not affect machine learning models because they cannot be used as a function in the model.

Dropping Missing Values

In Python, missing values can be removed using the dropna() function. dropna accepts one argument: comment. The comment can take two values: all or any. "any" removes specific rows that contain NAN or missing values, while "all" removes all rows that contain NAN or missing values.

Please Enter the Missing Value

In Python, missing values can be removed using the fillna() function. The fillna() function accepts a value that we want to fill in the missing space. We can fill in the missing values using mean, median and mode.

3.2.2 Data Formatting

Handling Outliers

Outliers are data points that are far from the most similar points. In other words, we can say that outliers are entities that stand out from the crowd. Outliers cause problems when building predictive models, e.g., B. long model training times, low accuracy, increased error variance, reduced normality, and reduced power of statistical tests.

There are two types of outliers: univariate and multivariate. One-dimensional outliers can be found in univariate distributions, while multivariate values can be found in n-dimensional spaces. We can detect and deal with outliers in the following ways:

- Boxplot: We can use a boxplot to group data points by quartiles. Group the data points between the first and third quartiles in a rectangular box. On a boxplot, outliers are also represented as individual points using the interquartile range.
- Scatterplot: A scatterplot displays points (or two variables) on a two-dimensional plot. One variable is positioned on the x-axis and the other on the y-axis.
- Z-Score: The Z-Score is a type of parametric approach to detecting outliers. A

Notes

normal distribution of the data is assumed. The outlier lies at the end of the normal curve distribution and is far from the mean:

$$Z = \frac{x-\mu}{\sigma}$$

- Interquartile Range (IQR): The IQR is a robust statistical measure of data spread. It is the difference between the third and first quartiles. These quartiles can be displayed in a boxplot. This is also known as the average spread, average 50% spread, or H-spread:

$$IQR = Q3 - Q1$$

Percentile: A percentile is a statistical measure that divides data into one hundred equal groups. Its value indicates the percentage of the population that falls below this value. For example, the 95th percentile means that 95% of people fall into this category.

Feature Encoding Techniques

Machine learning models are mathematical models that require numeric and integer values for calculations. Such models cannot work with categorical features. Therefore, we often need to convert categorical features into numeric features. The performance of the machine learning model is affected by the coding technique we use. The categorical values range from zero to N-1 categories.

Feature Scaling

In real life, most traits have different ranges, sizes, and units, such as age between 0 and 200 and salary between zero and thousands or millions. From the perspective of a data analyst or data scientist, how can we compare these characteristics when they are on different scales? Large-sized features will outweigh smaller-sized features in machine learning models. Fortunately, these problems can be solved with feature scaling or feature normalization. Function scaling brings all functions to the same size level. Not all types of algorithms require this; Some algorithms require uniquely scaled data, such as those based on Euclidean distance measures such as K-Nearest Neighbour and the K-Means Clustering algorithm.

Feature Transformation

The function transformation changes functions to have the required form. It also reduces the effect of outliers, handles skewed data, and makes the model more robust. The following list shows the diverse types of feature transformation:

The log transformation is the most commonly used mathematical transformation to convert skewed data to a normal distribution. Before applying the log transformation, ensure that all data values contain only positive values. otherwise, an exception or error message is thrown.

The squaring and cube transformation has a moderate impact on the shape of the distribution. It can be used to reduce left asymmetry.

The square and cube root transforms have a fairly strong transforming effect on the distribution shape, but it is weaker than the logarithms. It can be applied to right-skewed data.

Discretization can also be used to transform a column or numeric attribute. For example, the age of a candidate pool can be grouped into intervals like 0-10, 11-20 and so on. We can also use discretization to assign concept labels instead of intervals like young, old, and old.

Notes

If the feature is right or positive slanted or clustered at lower values we can apply the square root, cube root and logarithmic transformation, while if the feature is left or negative slanted or clustered at higher values then we can apply the cube, square, etc.

Feature Splitting

Function splitting helps data analysts and data scientists create more new functions for modelling. It enables machine learning algorithms to understand features and discover potential information for decision making; the breakdown of name characteristics into first name, middle name and surname and the breakdown of an address into house number, town, landmark, area, city, country, and zip code.

Composite functions such as string and date columns violate the principles of ordered data. Function splitting is a good option when you want to generate more functions from a compound function. To do this, we can use the components of a column. For example, we can easily get the year, month, and day of the week from a date object. These characteristics can directly affect the predictive model. There is no general rule for dividing functions into components; This depends on the properties of the function.

3.2.3 Data Normalisation Sets

Data normalization follows a specific set of rules called “normal forms”. These data normalization forms are categorised by level and each rule builds on the previous one i.e., H. You can only apply the second level of rules if your data matches the first level of rules and so on.

There are many types of data normalization forms, but here are four of the most common and widely used normal forms that can be applied to most data sets.

1. First Normal Form (1NF)

   First Normal Form, also known as 1NF, is the most basic form of data normalization. The core result of this rule is ensuring that there are no duplicate entries in a group. That means:

   Each cell must have a unique value.

   Each record must be unique.

   An example would be a table documenting a person’s name, address, gender and whether they ordered a Splunk t-shirt.

   Example data of the first normal form:

Name	Address	Gender	T-Shirt Order
Sanjay Kumar	A-37 Sector 12	Male	Large
Sarita Singh	H-1310, Sector 56	Female	Small
Sarita Singh	H-1310, Sector 56	Female	Medium
Chandan Sharma	L-101, Rail Vihar	Male	Medium

2. Second Normal Form (2NF)

   2NF is second normal form, based on the rules of first normal form. Again, the goal is to ensure that there are no duplicate entries in a record. Entries to which this data normalization rule applies must:

   ❖ Meet all 1NF requirements.

Notes

❖ Apply a primary key.

Applying a primary key means creating a separate table for subsets of data that fit across multiple rows. The data in each table can be linked using foreign key tags (numbers in this case).

When a primary key such as "customer number" is applied to our T-shirt example, subsets of data that require multiple rows (different T-shirt orders) must be placed in a new table with an appropriate foreign key.

Example of second normal form data:

Customer Number	Name	Address	Gender
1	Sanjay Kumar	A-37 Sector 12	Male
2	Sarita Singh	H-1310, Sector 56	Female
3	Chandan Sharma	L-101, Rail Vihar	Male

Customer Number	T-Shirt Order
1	Large
2	Small
2	Medium
3	Medium

3. Third Normal Form (3NF)

The third normal form data model includes the following rules:

❖ Must meet all 2NF requirements.

❖ Should only depend on the primary key (no transitive functional dependencies).

This means that when changes are made to the primary key, all affected data must be stored in a new table.

In our example, if you document a person's name, address and gender, but then change the name again, the gender may also change. Therefore, the class receives a foreign key and all data on the class is stored in a new table.

Example of third normal form data:

Customer Number	Name	Address	Gender ID
1	Sanjay Kumar	A-37 Sector 12	1
2	Sarita Singh	H-1310, Sector 56	2
3	Chandan Sharma	L-101, Rail Vihar	1

Customer Number	T-Shirt Order
1	Large
2	Small
2	Medium
3	Medium

Notes

Gender ID	Gender
1	Male
2	Female
3	Non-binary
4	Prefer not to say

4. Boyce and Codd Normal Form (3.5NF)

The Boyce Codd normal form, known as BCNF or 3.5NF, is an evolved version of the third normal form (3NF) data model. A 3.5NF is a 3NF table that has no overlapping candidate keys. This normal form includes the following rules:

- Must be in 3NF.
- X should be a super key for each functional dependency (X → Y).

This means that for a dependency X→Y, X cannot be a non-primary attribute if B is a prime attribute.

3.3 Summarising the Data Frame

Summarising a data set and applying a function to each group, be it an aggregation or a transformation, can be a critical component of a data analysis workflow. After loading, merging, and preparing a dataset, you may need to calculate group statistics or pivot tables for reporting or visualization. Pandas provides a versatile aggregation interface that allows you to segment, fragment and aggregate datasets in a natural way.

One of the reasons for the popularity of relational databases and SQL (which stands for "Structured Query Language") is the effortless way to join, filter, transform and aggregate data. However, query languages such as SQL impose certain limitations on the types of group operations that can be performed. Using the expressive power of Python and Pandas, we can perform fairly complex group operations by expressing them as Python user-defined functions that manipulate the data associated with each group:

- Split a Pandas object into parts using one or more keys (in the form of functions, arrays, or DataFrame column names).
- Calculate group summary statistics such as count, mean or standard deviation, or a user-defined function
- Apply intragroup transformations or other manipulations, e.g. B. Normalization, linear regression, classification, or subset selection
- Calculate pivot tables and crosstabs
- Perform quantile and other cluster statistical analysis

3.3.1 Descriptive Statistics

With copious amounts of data, the first step is often to generate summary statistics for the data in question. The most common summary statistics are the mean and standard deviation, which allow you to summarise the "typical" values in a dataset, but also other aggregates (sum, product, median, minimum, and maximum, quantiles, etc.).).

Measuring Central Tendency

Central tendency is the trend of values clustered around means, such as B. Mean, mode, and median values of the data. The main goal of central tendency is to calculate

Notes

the median of the observations. The central tendency drives the descriptive summary and provides quantitative information about a group of observations. It has the ability to represent a full set of observations. In the next few sections, let us look at each type of measure of central tendency in detail.

Mean

The mean is the arithmetic mean or average calculated by dividing the sum of observations by the number of observations. It is sensitive to outliers and noise, resulting in whenever unusual or unusual values are added to a group whose mean deviates from the typical central value. Suppose x1, x2, .. . , x N are N observations.

Let us calculate the median of the column “Communication Skills Score” using the pandas library as follows:

```
# Import pandas library

import pandas as pd

# Create dataframe

sample_data = {‘name’: [Jagan, Abha, ‘Ananya’, ‘Sanjay, ‘Vikas’],

‘gender’: [‘M’, ‘F’, ‘F’, ‘M’, ‘M’],

‘communication_skill_score’: [40, 45, 23, 39, 39],

‘quantitative_skill_score’: [38, 41, 42, 48, 32]}

data = pd.DataFrame(sample_data, columns = [‘name’, ‘gender’, ‘communcation_
skill_score’, ‘quantitative_skill_score’])

# find mean of communication_skill_score column

data[‘communcation_skill_score’].mean(axis=0)
```

Output:

37.2

In the preceding code block, we have created one DataFrame named data that has four columns (name, gender, communication_skill_score and quantitative_skill_score) and computed the mean using the mean(axis=0) function. Here, axis=0 represents the mean along the rows.

Mode

Mode is the element that occurs most frequently in a group of observations. The mode value appears frequently in the data and is primarily used for categorical values. If all values in a group are unique or non-repeating, then there is no mode. It is also possible that more than one value has the same frequency. In such cases, there can be several modes.

Let us calculate the mode value of the communication skill score column using the pandas library as follows:

```
# find mode of communication_skill_score column

data[‘communcation_skill_score’].mode()
```

Output:

39

Notes

In the preceding code block, we have computed the mode of the communication skill score column using the mode() function. Let us compute another central tendency measure: the median.

Median

The median is the midpoint or mean in a group of observations. It is also known as the 50th percentile. The median is less affected by outliers and noise than the mean and is therefore considered a better statistical measure for reporting.It comes close to a typical central value. Let us calculate the median of the column "Communication Skills Score" using the pandas library as follows:

```
# find median of communication_skill_score column

data['communcation_skill_score'].median()
```

Output:

39.0

In the code block above, we calculated the median of the Communication Skills Score column using the median() function. In the next section, let us understand and calculate measures of dispersion.

3.3.2 Basic of Grouping

Categorising a dataset and applying a function to each group, be it an aggregation or a transformation, can be a critical component of a data analysis workflow. After loading, merging, and preparing a dataset, you may need to calculate group statistics or pivot tables for reporting or visualization. Pandas provides a versatile aggregation interface that allows you to segment, fragment and aggregate datasets in a natural way.

Hadley Wickham, an author of many popular packages for the R programming language, coined the term split-apply-combine for describing group operations. In the first stage of the process, data contained in a pandas object, whether a Series, DataFrame, or otherwise, is split into groups based on one or more keys that you provide. The splitting is performed on a particular axis of an object. For example, a DataFrame can be grouped on its rows (axis="index") or its columns (axis="columns"). Once this is done, a function is applied to each group, producing a new value. Finally, the results of all those function applications are combined into a result object.

Each grouping key can take many forms and the keys do not have to be all of the same type:

- A list or array of values that is the same length as the axis being grouped
- A value indicating a column name in a DataFrame
- A dictionary or Series giving a correspondence between the values on the axis being grouped and the group names
- A function to be invoked on the axis index or the individual labels in the index

The object returned by groupby supports iteration, generating a sequence of 2-tuples containing the group name along with the chunk of data. By default, groupby groups on axis="index", but you can group on any of the other axes.

Indexing a GroupBy object created from a DataFrame with a column name or array of column names has the effect of column subsetting for aggregation. Especially for large datasets, it may be desirable to aggregate only a few columns.

Notes

Using Python functions is a more generic way of defining a group mapping compared with a dictionary or Series. Any function passed as a group key will be called once per index value (or once per column value if using axis="columns"), with the return values being used as the group names.

Mixing functions with arrays, dictionaries, or Series is not a problem, as everything gets converted to arrays internally.

A final convenience for hierarchically indexed datasets is the ability to aggregate using one of the levels of an axis index.

3.3.3 ANOVA

As we have seen, the central tendency represents the mean of a group of observations but does not provide the general picture of an observation. Dispersion metrics measure the variance in the observations. The most popular spread metrics are range, interquartile range (IQR), variance and standard deviation. These variability metrics assess the variability of the observations, or the variability of the observations. Let us look at each measure of dispersion in detail, like this:

- Range: The range is the difference between the maximum and minimum value of an observation. It is easy to calculate and easy to understand. Its unit is the same as the unit of observations. Let us calculate the range of communication skill scores as follows:

```
column_range=data['communcation_skill_score'].max()-data['communcation_skill_score'].min()
print(column_range)
```

Output:

```
22
```

In the code block above, we calculated the range of communication ability scores by taking the difference between the maximum and minimum values. The maximum and minimum scores were calculated using the max() and min() functions.

- IQR: IQR is the difference between the third and first quartile. It is easy to calculate and easy to understand. Its unit is the same as the unit of observations. It measures the middle 50% of the observation. It represents the area where most of the observations take place. IQR is also known as mid-spread or middle 50% or H-spread. Let us calculate the IQR of the communication skills scores as follows:

```
# First Quartile
q1 = data['communcation_skill_score'].quantile(.25)
# Third Quartile
q3 = data['communcation_skill_score'].quantile(.75)
# Inter Quartile Ratio
iqr=q3-q1
print(iqr)
```

Output:

```
1.0
```

Notes

In the code block above, we calculated the IQR of the communication skills scores by taking the difference between the first and third quartile scores. The first and third quartile values were calculated using the quantile(0.25) and quantile(0.75) functions.

- Variance: The variance measures the deviation from the mean. It is the mean of the squared difference between the observed values and the mean. The main problem with variance is its unit of measure since it squares the difference between the observations and the mean. Suppose x1, x2, . . . , x N are N observations.

  Let us compute the variance of communication skill scores, as follows:

  ```
  # Variance of communication_skill_score

  data['communcation_skill_score'].var()
  ```

  Output:

  69.2

In the preceding code block, we have computed the variance of communication skill scores using the var() function.

- Standard Deviation: This is the square root of the variance. Its unit is the same as for the original observations. This makes it easier for an analyst to estimate the exact deviation from the mean. The lowest value of the standard deviation represents the smallest distance of the observations from the mean; This means that the observations are less common. The highest standard deviation value represents a large distance between the observations and the mean; H. the observations are widely scattered. The standard deviation is represented mathematically by the Greek letter sigma (Σ). Suppose x1, x2, .. . , x N are N observations.

  Let us compute the standard deviation of communication skill scores, as follows:

  ```
  # Standard deviation of communication_skill_score

  data['communcation_skill_score'].std()
  ```

  Output:

  8.318653737234168

In the preceding code block, we have computed the standard deviation of communication skill scores using the std() function.

We can also try to describe the function to get all the summary statistics in a single command. The describe() function returns the count, mean, standard deviation, first quartile, median, third quartile and minimum and maximum values for each numeric column in the DataFrame and is illustrated in the following code block:

```
# Describe dataframe

data.describe()
```

Output:

communcation_skill_score quantitative_skill_score

count	5.000000	5.000000
mean	37.200000	40.200000
std	8.318654	5.848077

Notes

min	23.000000	32.000000
25%	39.000000	38.000000
50%	39.000000	41.000000
75%	40.000000	42.000000
max	45.000000	48.000000

In the preceding code block, we have generated a descriptive statistics summary of data using the describe() method.

Skewness and Kurtosis

Skewness measures the symmetry of a distribution. Shows how much the distribution deviates from a normal distribution. Its values can be zero, positive and negative. A zero value represents a perfectly normal form of distribution. Positive biases are shown with the tails pointing to the right, i. H. Outliers are shifted to the right and the data is stacked on the left.A negative distortion is displayed with the ends pointing to the left, i.e., H. Outliers are shifted to the left and the data is stacked to the right. Positive skewness occurs when the mean is greater than the median and the mode. Negative skewness occurs when the mean is less than the median and the mode.

```
# skewness of communication_skill_score column

data['communcation_skill_score'].skew()
```

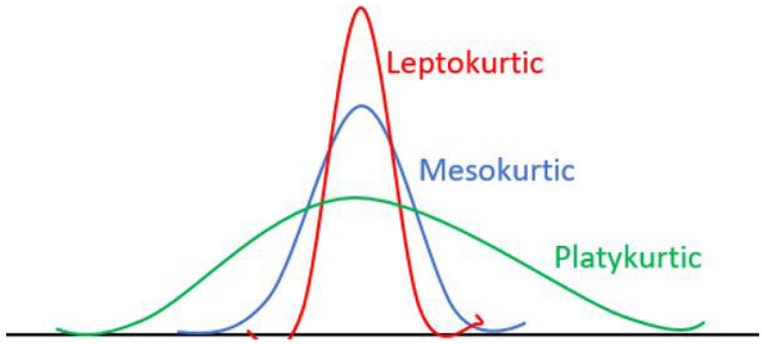
Output:

```
-1.704679180800373
```

In the code block above, we calculated the skewness of the communication skills assessment column using the skew() method. Kurtosis measures the tail (thickness of the tail) compared to a normal distribution. High kurtosis values have a strong tail, meaning there are more outliers in the observations and low kurtosis values have a light tail, meaning there are fewer outliers in the observations. There are three types of kurtosis forms: mesokurtic, platykurtic and leptokurtic. Let's define them individually as follows:

- A normal distribution without kurtosis is called a mesokurtic distribution.
- A platykurtic distribution has a negative kurtosis value and is thin-skinned compared to a normal distribution.
- A leptokurtic distribution has a kurtosis value greater than 3 and a fat tail compared to a normal distribution.

Let's look at the nature of the kurtosis shapes in the following diagram:



Notes

A histogram is an effective medium to present skewness and kurtosis. Let us compute the kurtosis of the communication skill score column, as follows:

```
# kurtosis of communication_skill_score column

data['communcation_skill_score'].kurtosis()
```

Output:

```
3.6010641852384015
```

In the preceding code block, we have computed the kurtosis of the communication skill score column using the kurtosis() method.

3.3.4 Correlation

Measuring the relationship between variables helps data analysts understand the dynamics between variables. For example, a human resources manager needs to understand the strength of the correlation between employee performance ratings and satisfaction ratings. Statistics provides two measures of covariance and correlation to understand the relationship between variables. Covariance measures the relationship between a pair of variables. It shows the degree of change in the variable, i.e., H. how changing one variable affects the other variable. Its value ranges from -infinity to +infinity. The problem with covariance is that it does not provide effective inferences because it is not normalised. We determined the relationship between communication and quantitative ability assessment using covariance on functional data.cov().

Pearson's Correlation Coefficient

Correlation shows how variables correlate with each other. Correlation provides a better understanding than covariance and is a normalised version of covariance. Correlation ranges from -1 to 1. A negative value represents an increase in one variable that causes other variables to decrease, or the variables are moving in the same direction. A positive value represents an increase in one variable leading to an increase in another variable, or a decrease in one variable leading to a decrease in another variable. A null value means that there is no relationship between the variable or that the variables are independent of each other. The correlation between the columns of the data frame is calculated using the data.corr(method ='pearson') function.

The 'method' parameter can take one of the following three parameters:

- pearson: Standard correlation coefficient
- kendall: Kendall's tau correlation coefficient
- spearman: Spearman's rank correlation coefficient

Spearman's Rank Correlation Coefficient

Spearman's rank correlation coefficient is Pearson's correlation coefficient across the ranks of the observations. It is a nonparametric measure of rank correlation. Assesses the strength of the association between two classified variables. Rank variables are ordinal numbers that are ordered in order. First we classify the observations and then calculate the rank correlation. It can be applied to both continuous and discrete ordinal variables. When the data distribution is skewed or an outlier is affected, Spearman rank correlation is used instead of Pearson correlation because there are no assumptions about the data distribution.

Notes

Kendall's Rank Correlation Coefficient

Kendall's rank correlation coefficient, or Kendall's tau coefficient, is a nonparametric statistic used to measure the association between two ordinal variables. It is a kind of rank correlation. It measures the similarity or dissimilarity between two variables. If both variables are binary, then Pearson = Spearman = Kendall's tau.

3.4 Model Development

The library you use to develop models depends on the application. Many statistical problems can be solved using simpler techniques such as ordinary least squares regression, while other problems may require more advanced machine learning methods. Luckily, Python has become one of the preferred languages for implementing analysis methods, giving you plenty of tools to explore.

A common workflow for model development is to use pandas to load and clean data before moving to a modelling library to build the model itself. An important part of the model development process in machine learning is called feature engineering.

This can describe any transformation or data analysis that extracts information from a raw dataset that might be useful in a modelling context. GroupBy and data aggregation tools are commonly used in feature engineering context.

The point of contact between pandas and other parsing libraries are usually NumPy arrays. Patsy is a Python library for describing statistical models (particularly linear models) with a string based "formula syntax" inspired by (but not exactly identical to) the formula syntax of the R and S statistical programming languages. Patsy has good support for specifying linear models in statistical models.-

Statsmodels is a Python library for customising many types of statistical models, performing statistical tests, and exploring and visualising data. statsmodels contains more "classic" frequentist statistical methods, while Bayesian methods and machine learning models can be found in other libraries.

Model types found in statsmodels include:

- Linear models, generalised linear models and robust linear models
- Linear mixed effects models
- Methods of analysis of variance (ANOVA)
- Time series processes and state space models
- Generalised method of moments

Scikit-learn is one of the most widely used and trusted general-purpose Python machine learning toolkits. Includes a wide range of standard supervised and unsupervised machine learning methods with tools for model selection and evaluation, data transformation, data loading and model persistence. These models can be used for classification, clustering, prediction, and other common tasks.

3.4.1 Simple and Multiple Linear Regression

The most common goal in statistics is to answer the question "Is the variable X (or more likely X1?)..,

Regression looks for relationships between variables. For example, you can observe several employees of a company and try to understand how their salaries

Notes

depend on their characteristics such as experience, level of education, function, place of employment, etc.

This is a regression problem where the data for each employee is an observation. It is assumed that experience, education, role, and city are independent characteristics, while salary depends on them.

Nowhere is the connection between statistics and data science stronger than in the field of prediction, specifically the prediction of an outcome (target) variable based on the values of other 'predictor' variables. This process of training a model on data where the outcome is known for subsequent application to data where the outcome is unknown is called supervised learning. Another important connection between data science and statistics is in the area of anomaly detection, where regression diagnostics, originally intended for data analysis and improving regression models, can be used to detect unusual data sets.

Simple Linear Regression

Simple linear regression provides a model of the relationship between the magnitude of one variable and that of a second; For example, if X increases, Y also increases. Or if X increases, Y decreases. Correlation is another method of measuring the relationship between two variables. The difference is that correlation measures the strength of an association between two variables, while regression quantifies the nature of the relationship.

There are diverse types of linear regression models in statistical models, from the most basic (p.(e.g., ordinary least squares) to more complex (e.g., iteratively reweighted least squares). The linear models in statsmodels have two different main interfaces: matrix-based and formula-based.Access is via this API module:

```
import statsmodels.api as sm
import statsmodels.formula.api as smf
```

A linear model is fitted with an intercept term. The sm.add_constant function can add an intercept column to an existing matrix. The sm.OLS class can fit an ordinary least squares linear regression.

The model's fit method returns a regression results object containing estimated model parameters and other diagnostics.

The summary method on results can print a model detailing diagnostic output of the model.

There are many additional tools for analysis, diagnostics, and visualization of linear model results in statsmodels that you can explore. There are also other kinds of linear models beyond ordinary least squares.

Multiple Linear Regression

When there are multiple predictors, the equation is simply extended to accommodate them:

$$Y=b_0+b_1X_1+b_2X_2+...+b_pX_p+e$$

Instead of a line, we now have a linear model—the relationship between each coefficient and its variable (feature) is linear.

All of the other concepts in simple linear regression, such as fitting by least

Notes

squares and the definition of fitted values and residuals, extend to the multiple linear regression setting.

3.4.2 Model Evaluation Using Visualisation

Data visualization is the first step in data analysis system to easily understand and communicate information. Present information and data graphically using visual elements such as tables, graphs, charts, and maps. Helps analysts understand patterns, trends, outliers, distributions, and relationships. Data visualization is an efficient way to process a large number of data sets.

Python provides several libraries for data visualization, such as Matplotlib, Seaborn and Bokeh. We focus on Matplotlib, the basic Python library for visualization.

As we know, a picture says more than a thousand words. Humans understand visual things best. Visualization helps present things to any audience and can easily explain a complex phenomenon in simple terms. Python provides a number of visualization libraries such as Matplotlib, Seaborn and Bokeh.

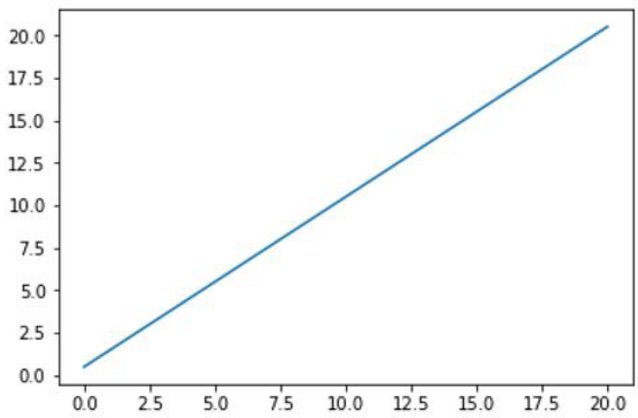
Matplotlib is the most popular Python module for data visualization. It is a base library for most advanced Python visualization modules, such as Seaborn. It offers flexible and easy-to-use built-in functions for creating figures and graphics.

To create a basic plot in Matplotlib, we need to invoke the plot() function in the matplotlib.pyplot subpackage. This function produces a two-dimensional plot for a single list or multiple lists of points with known x and y coordinates.

Let us see a small demo code for visualising the line plot:

```
# Add the essential library matplotlib
import matplotlib.pyplot as plt
import numpy as np
# create the data
a = np.linspace(0, 20)
# Draw the plot
plt.plot(a, a + 0.5, label='linear')
# Display the chart
plt.show()
```

This results in the following output:



Notes

First, we import the Matplotlib and NumPy modules. After that, we create the data with NumPy's linespace() function and plot this data with Matplotlib's plot() function. Finally, we display the figure using the show() function.

There are two basic components of a plot: the character and the axes. The figure is a container on which everything is drawn. Contains components such as plots, subplots, axes, titles, and a legend. In the next section we will focus on these components, which serve as accessories for the diagrams.

Accessories for Charts

In the matplotlib module, we can add titles and axes labels to a graph. We can add a title using plt.title() and labels using plt.xlabel() and plt.ylabel().

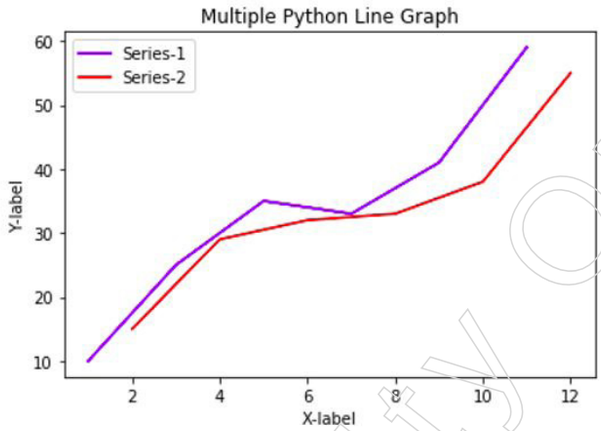
Multiple diagrams mean multiple objects, e.g., B. lines, bars, and scatters. Points from different series can be displayed in a single chart. Legends or chart series reflect the y-axis.A legend is a box that appears on the right or left side of a chart and shows what each item on the chart represents. Let us see an example where we will see how this accessory is used in our tables:

```
# Add the required libraries
import matplotlib.pyplot as plt

# Create the data
x = [1,3,5,7,9,11]
y = [10,25,35,33,41,59]

# Let's plot the data
plt.plot(x, y,label='Series-1', colour='blue')

# Create the data
x = [2,4,6,8,10,12]
y = [15,29,32,33,38,55]

# Plot the data
plt.plot(x, y, label='Series-2', colour='red')

# Add X Label on X-axis
plt.xlabel("X-label")

# Add X Label on X-axis
plt.ylabel("Y-label")

# Append the title to graph
plt.title("Multiple Python Line Graph")

# Add legend to graph
plt.legend()
```

Notes

```
# Display the plot
plt.show()
```

This results in the following output:



In the preceding graph, two lines are shown on a single graph. We have used two extra parameters – label and colour – in the plot() function. The label parameter defines the name of the series and colour defines the colour of the line graph. In the upcoming sections, we will focus on diverse types of plots. We will explore a scatter plot in the next section.

Scatter Plot

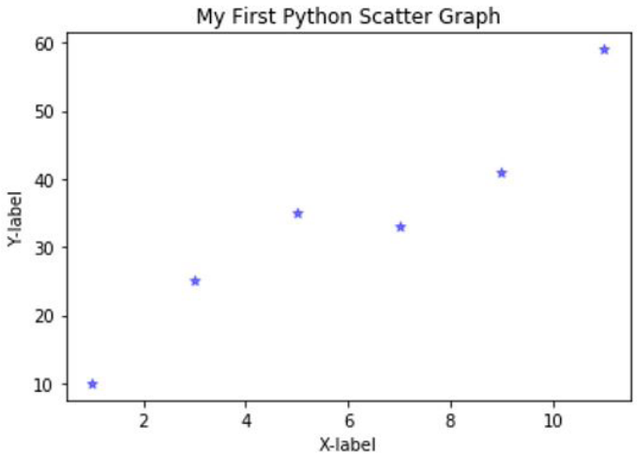
Scatter plots draw data points using Cartesian coordinates to show the values of numerical values. They also represent the relationship between two numerical values. We can create a scatter plot in Matplotlib using the scatter() function, as follows:

```
# Add the essential library matplotlib
import matplotlib.pyplot as plt

# create the data
x = [1,3,5,7,9,11]
y = [10,25,35,33,41,59]

# Draw the scatter chart
plt.scatter(x, y, c='blue', marker='*',alpha=0.5)

# Append the label on X-axis
plt.xlabel("X-label")

# Append the label on X-axis
plt.ylabel("Y-label")

# Add the title to graph
plt.title("Scatter Chart Sample")

# Display the chart
```

Notes

```
plt.show()
```

This results in the following output:



In the preceding scatter plot, the scatter() function takes x-axis and y-axis values. In our example, we are plotting two lists: x and y. We can also use optional parameters such as c for colour, alpha for the transparency of the markers, ranging between 0 and 1 and marker for the shape of the points in the scatter plot, such as *, o, or any other symbol. In the next section, we will focus on the line plot.

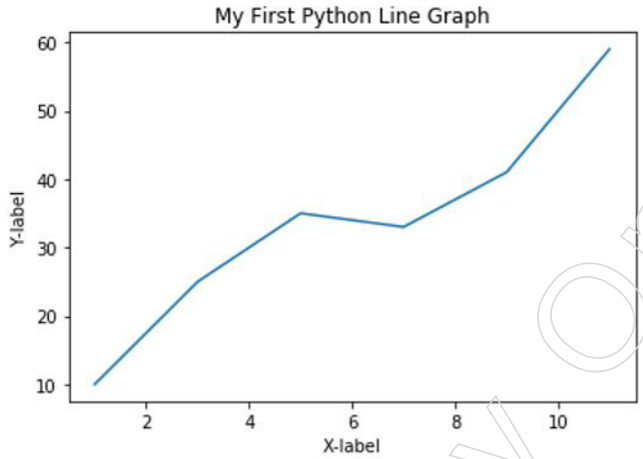
Line plot

A line plot is a chart that displays a line between two variables. It has a sequence of data points joined by a segment:

```
# Add the essential library matplotlib
import matplotlib.pyplot as plt

# create the data
x = [1,3,5,7,9,11]
y = [10,25,35,33,41,59]

# Draw the line chart
plt.plot(x, y)

# Append the label on X-axis
plt.xlabel("X-label")

# Append the label on X-axis
plt.ylabel("Y-label")

# Append the title to chart
plt.title("Line Chart Sample")

# Display the chart
plt.show()
```

Notes

This results in the following output:



In the preceding line plot program, the plot() function takes x-axis and y-axis values. In the next section, we will learn how to plot a pie chart.

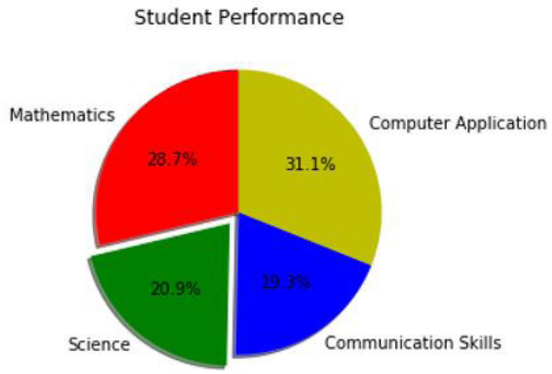
Pie plot

A pie plot is a circular graph that is split up into wedge-shaped pieces. Each piece is proportionate to the value it represents. The total value of the pie is 100 percent:

```
# Add the essential library matplotlib
import matplotlib.pyplot as plt

# create the data
subjects = [“Mathematics”,”Science”,”Communication Skills”,”Computer Application”]
scores = [85,62,57,92]

# Plot the pie plot
plt.pie(scores,
labels=subjects,
colours=[‘r’,’g’,’b’,’y’],
startangle=90,
shadow= True,
explode=(0,0.1,0,0),
autopct=’%1.1f%%’)

# Add title to graph
plt.title(“Student Performance”)

# Draw the chart
plt.show()
```

This results in the following output:

Notes



In the preceding code of the pie chart, we specified values, labels, colours, startangle, shadow, explode and autopct. In our example, values is the scores of the student in four subjects and labels is the list of subject names. We can also specify the colour list for the individual subject scores. The startangle parameter specifies the first value angle, which is 90 degrees; this means the first line is vertical.

Optionally, we can also use the shadow parameter to specify the shadow of the pie slice and the explode parameter to pull out a pie slice list of the binary value. If we want to pull out a second pie slice, then a tuple of values would be (0, 0.1, 0, 0). Let us now jump to the bar plot.

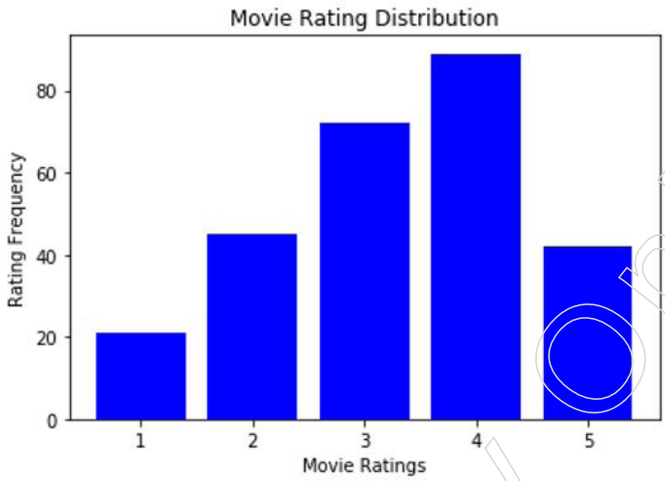
Bar plot

A bar plot is a visual tool to compare the values of various groups. It can be drawn horizontally or vertically. We can create a bar graph using the bar() function:

```
# Add the essential library matplotlib
import matplotlib.pyplot as plt

# create the data
movie_ratings = [1,2,3,4,5]
rating_counts = [21,45,72,89,42]

# Plot the data
plt.bar(movie_ratings, rating_counts, colour='blue')

# Add X Label on X-axis
plt.xlabel("Movie Ratings")

# Add X Label on X-axis
plt.ylabel("Rating Frequency")

# Add a title to graph
plt.title("Movie Rating Distribution")

# Show the plot
plt.show()
```

This results in the following output:

Notes



In the preceding bar chart program, the bar() function takes x-axis values, y-axis values and a colour. In our example, we are plotting movie ratings and their frequency. Movie ratings are on the x axis and the rating frequency is on the y axis. We can also specify the colour of the bars in the bar graph using the colour parameter. Let us see another variant of bar plot in the next subsection.

Histogram plot

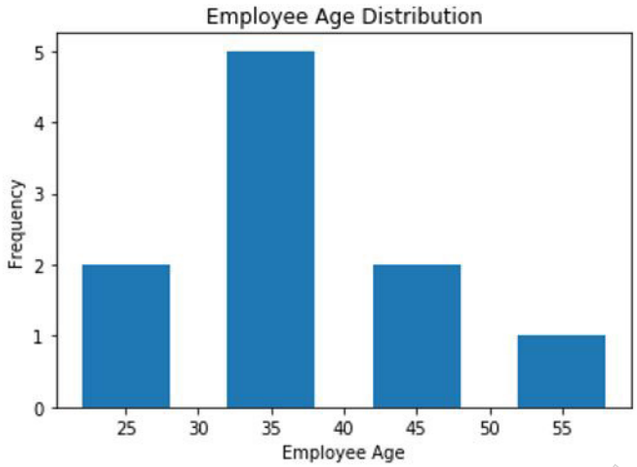
A histogram shows the distribution of a numeric variable. We create a histogram using the hist() method. It shows the probability distribution of a continuous variable. A histogram only works on a single variable while a bar graph works on two variables:

```
# Add the essential library
import matplotlib.pyplot as plt

# Create the data
employee_age = [21,28,32,34,35,35,37,42,47,55]

# Create bins for histogram
bins = [20,30,40,50,60]

# Plot the histogram
plt.hist(employee_age, bins, rwidth=0.6)

# Add X Label on X-axis
plt.xlabel("Employee Age")

# Add X Label on X-axis
plt.ylabel("Frequency")

# Add title to graph
plt.title("Employee Age Distribution")

# Show the plot
```

Notes

```
plt.show()
```

This results in the following output:



In the preceding histogram, the hist() function takes values, bins and rwidth. In our example, we are plotting the age of the employee and using a bin of 10 years. We are starting our bin from 20 to 60 with a 10-year bin size. We are using a relative bar width of 0.6, but you can choose any size for thicker and thinner width. Now it is time to jump to the bubble plot, which can handle multiple variables in a two-dimensional plot.

Bubble plot

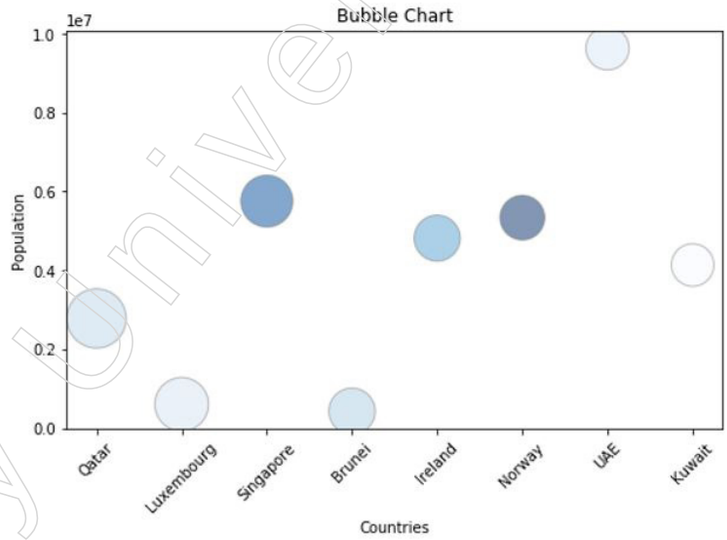
A bubble plot is a type of scatter plot. It not only draws data points using Cartesian coordinates but also creates bubbles on data points. Bubble shows the third dimension of a plot. It shows three numerical values: two values are on the x and y axes and the third one is the size of data points (or bubbles):

```
# Import the required modules
import matplotlib.pyplot as plt
import numpy as np

# Set figure size
plt.figure(figsize=(8,5))

# Create the data
countries = ['Qatar','Luxembourg','Singapore','Brunei','Ireland','Norway','UAE','Kuwait']
populations = [2781682, 604245,5757499,428963,4818690,5337962,9630959,4137312]
gdp_per_capita = [130475, 106705, 100345, 79530, 78785, 74356,69382, 67000]

# scale GDP per capita income to shoot the bubbles in the graph
scaled_gdp_per_capita = np.divide(gdp_per_capita, 80)

colours = np.random.rand(8)
# Draw the scatter diagram
```

Notes

```
plt.scatter(countries, populations, s=scaled_gdp_per_capita, c=colours, cmap="Blues",edgecolours="grey", alpha=0.5)

# Add X Label on X-axis
plt.xlabel("Countries")

# Add Y Label on X-axis
plt.ylabel("Population")

# Add title to graph
plt.title("Bubble Chart")

# rotate x label for clear visualization
plt.xticks(rotation=45)

# Show the plot
plt.show()
```

This results in the following output:



In the preceding plot, a bubble chart is created using the scatter function. Here, the important thing is the s (size) parameter of the scatter function. We assigned a third variable, scaled_gdp_per_capita, to the size parameters. In the preceding bubble plot, countries are on the x axis, the population is on the y axis and GDP per capita is shown by the size of the scatter point or bubble. We also assigned a random colour to the bubbles to make it attractive and more understandable. From the bubble size, you can easily see that Qatar has the highest GDP per capita and Kuwait has the lowest GDP per capita. In all the preceding sections, we have focused on most of the Matplotlib plots and charts. Now, we will see how we can plot the charts using the pandas module.

pandas plotting

The pandas library offers the plot() method as a wrapper around the Matplotlib library. The plot() method allows us to create plots directly on pandas DataFrames. The following plot() method parameters are used to create the plots:

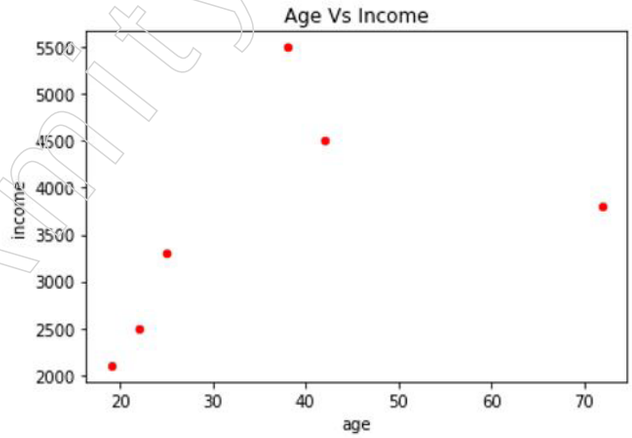
Notes

- kind: A string parameter for the type of graph, such as line, bar, barh, hist, box, KDE, pie, area, or scatter.
- figsize: This defines the size for a figure in a tuple of (width, height).
- title: This defines the title for the graph.
- grid: Boolean parameter for the axis grid line.
- legend: This defines the legend.
- xticks: This defines the sequence of x-axis ticks.
- yticks: This defines the sequence of y-axis ticks.

Let us create a scatter plot using the pandas plot() function:

```
# Import the required modules
import pandas as pd
import matplotlib.pyplot as plt

# Let's create a Dataframe
df = pd.DataFrame({
'name':['Ajay','Malala','Abhijeet','Yming','Desilva','Lisa'],
'age':[22,72,25,19,42,38],
'gender':['M','F','M','M','M','F'],
'country':['India','Pakistan','Bangladesh','China','Srilanka','UK'],
'income':[2500,3800,3300,2100,4500,5500]
})

# Create a scatter plot
df.plot(kind='scatter', x='age', y='income', colour='red', title='Age Vs Income')

# Show figure
plt.show()
```

This results in the following output:



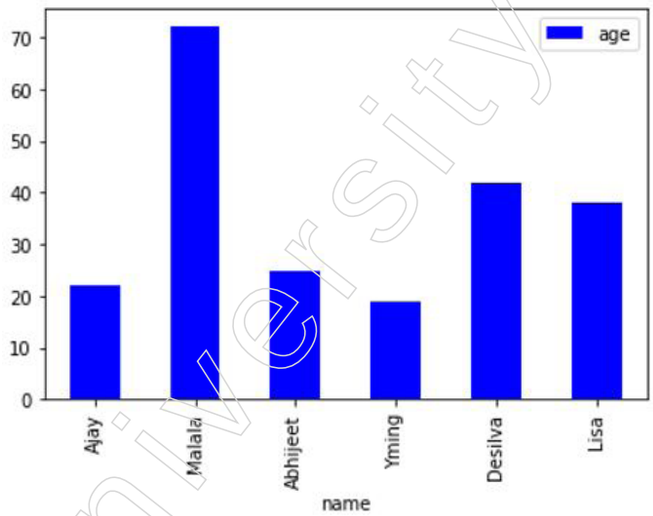
In the preceding plot, the plot() function takes kind, x, y, colour, and title values. In our example, we are plotting the scatter plot between age and income using the kind parameter as 'scatter'. The age and income columns are assigned to the x and y

Notes

parameters. The scatter point colour and the title of the plot are assigned to the colour and title parameters:

```
import matplotlib.pyplot as plt
import pandas as pd

# Create bar plot
df.plot(kind='bar',x='name', y='age', colour='blue')

# Show figure
plt.show()
```

This results in the following output:



In the preceding plot, the plot() function takes kind, x, y, colour, and title values. In our example, we are plotting the bar plot between age and income using the kind parameter as 'bar'. The name and age columns are assigned to the x and y parameters. The scatter point colour is assigned to the colour parameter.

3.4.3 Polynomial Regression and Pipelines

Polynomials are mathematical expressions with non-negative strategies. Examples of polynomial functions are linear, quadratic, cubic and quartic functions. NumPy offers the polyfit() function to generate polynomials using least squares. This function takes x-coordinate, y-coordinate and degree as parameters and returns a list of polynomial coefficients.

NumPy also offers polyval() to evaluate the polynomial at given values. This function takes coefficients of polynomials and arrays of points and returns resultant values of polynomials. Another function is linspace(), which generates a sequence of equally separated values. It takes the start, stop and the number of values between the start-stop range and returns equally separated values in the closed interval.

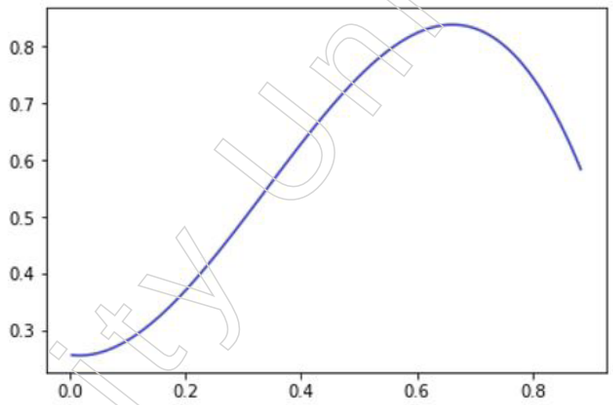
Let us see an example to generate and evaluate polynomials using NumPy, as follows:

```
# Import required libraries NumPy, polynomial and matplotlib
import numpy as np
```

Notes

```
import matplotlib.pyplot as plt

# Generate two random vectors
v1=np.random.rand(10)
v2=np.random.rand(10)

# Creates a sequence of equally separated values
sequence = np.linspace(v1.min(),v1.max(), num=len(v1)*10)

# Fit the data to polynomial fit data with 4 degrees
# of the polynomial
coefs = np.polyfit(v1, v2, 3)

# Evaluate polynomial on given sequence
polynomial_sequence = np.polyval(coefs,sequence)

# plot the polynomial curve
plt.plot(sequence, polynomial_sequence)

# Show plot
plt.show()
```

This results in the following output:



The graph shown in the preceding screenshot will change in each iteration using the program written previously. The reason for this fluctuation is the random value generation of vectors.

3.4.4 R-squared and MSE for In-Sample Evaluation

In this topic, we discuss regression evaluation measures to understand the performance level of a regression model. Model evaluation is one of the key aspects of any machine learning model building process. It helps us evaluate how our model will perform when we put it into production.

R-squared (or coefficient of determination) is a statistical model score that assesses the goodness of a regression model. Helps data analysts explain model performance

Notes

compared to the base model.Its value ranges from 0 to 1. A value close to 0 represents a poor model, while a value close to 1 represents a perfect fit. Sometimes the R squared gives a negative value. That means your model is inferior to the average base model.

- MSE: MSE is an acronym for Root Mean Square Error. It is explained as the square of the change between the original and predicted values and the average between them for all values.
- MAE: MAE is an abbreviation for Mean Absolute Error. It is explained as the absolute change between the original and predicted values and the average between them for all values.
- RMSE: RMSE is an acronym for Root Mean Square Error. It is explained as the square root of MSE.

scikit-learn provides the metric class for evaluating models. To evaluate the regression model, we have methods for R-squared, MSE, MAE and RMSE at our disposal. Each of the methods requires two inputs: the actual values of the test set and the predicted values.

The mean_absolute_error, mean_squared_error and r2_score function from sklearn. metrics provide evaluation functions for MAE, MSE and RMSE evaluations, respectively.

3.4.5 Prediction and Decision Making

No other statistical method has found more use over the years than regression, the process of establishing a relationship between multiple predictor variables and an outcome variable. The basic form is linear: each predictor variable has a coefficient that describes a linear relationship between the predictor and the outcome. More advanced forms of regression such as polynomial and spline regression allow for a non-linear relationship. In classical statistics, the emphasis is on finding a good fit to observed data to explain or describe a phenomenon. The power of this fit lies in how traditional within-sample metrics are used to evaluate the model. In contrast, the goal in data science is often to predict values for new data. Therefore, metrics based on prediction accuracy are used for out-of-sample data. Variable selection methods are used to reduce dimensionality and create more compact models.

Using Regression to Predict Taxi Fares

Imagine you work for a taxi company and one of the biggest complaints your customers have is that they do not know how much a ride will cost until it is over. Because distance is not the only variable that determines the amount of the fee. You decide to do something about it by creating a mobile app that allows customers to guess the fare when they get into a cab. To provide the intelligence for the app, he intends to use the vast amounts of fee data the company has collected over the years to develop a machine learning model.

We train a regression model to predict a fare based on time of day, day of week and pickup and drop-off locations. We use the Pandas-Corr method to find out what influence input variables such as latitude and longitude have on fare values.

If the dataset contains outliers, they can skew the results of machine learning models. We filter the dataset by removing negative fare amounts and setting appropriate fare and distance limits. We can test three different learning algorithms to determine which provides the most accurate fit and use cross-validation to measure accuracy. We will start with a linear regression model, compare the results of a

Notes

RandomForestRegressor to the same data set and see how their accuracy compares. Finally, we evaluate and compare the results of the GradientBoostingRegressor. Gradient enhancement machines use multiple decision trees, each trained to compensate for the error in the previous one's output.

When the GradientBoostingRegressor produced the highest cross-validated coefficient of determination, train it on the entire data set using the model.fit function. Finally, we use the trained model to make some predictions.

3.5 Model Evaluation

Building models is a continuous art. As we start adding and removing variables to our models, we need a way to compare models to each other and a consistent way to measure model performance. There are many ways to compare models.

3.5.1 Model Evaluation

A model's residuals compare the model's estimates to the actual values in the data. A q-q chart is a graphical technique that determines whether your data fits a reference distribution. Because many models assume the data is normally distributed, a q-q chart is one way to ensure that your data is truly normally distributed.

With all of our models, we can capture all of our coefficients and the model with which they are associated. Since it does not make much sense to look at one large column of values, we can plot our coefficients to quickly see how the models estimate the parameters compared to each other.

We can use the analysis of variance (ANOVA) method to compare them. ANOVA gives us the residual sum of squares (RSS), which is one way to measure performance (less is better).

Another way to calculate model performance is to use the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC). These methods apply a penalty for each feature added to the model (lower AIC and BIC values are better). So, we must strive to balance performance and economy.

We can perform the same calculations and model diagnostics for generalised linear models (GLMs). We can use pattern discrepancies to do model comparisons. We can also do the same calculations in a logistic regression.

Cross-validation is another technique for comparing models. One of the main benefits is that you can use new data to determine how well your model is performing. It does this by splitting your data into k parts. It keeps one of the pieces aside as the "test" set and then fits the model to the remaining k − 1 pieces, the "training" set. The fitted model is then used in the "test" and an error rate is calculated. This process is repeated until all k parts have been used as a "test set". The model's final error is an average of all models.

Cross-validation can be done in many different ways. The method just described is called "k-fold cross-validation." Alternative ways to perform cross-validation include "leave-one-out" cross-validation, where the training data consists of all data except one observation called a test set.

3.5.2 Over-Fitting, Under-Fitting and Model Selection

Suppose you are visiting a foreign country and the taxi driver cheated you. You might be tempted to say that all taxi drivers in this country are thieves. Overgeneralising is all

Notes

too common among humans and unfortunately machines can fall into the same trap if we are not careful. In machine learning, this is called overfitting: it means the model works well with the training data but does not generalise well.

Complex models such as deep neural networks can detect subtle patterns in the data. However, if the training set is noisy or too small, resulting in sampling noise, the model is likely to see patterns in the data. noise itself (as in the example of the taxi driver). Obviously, these patterns cannot be transferred to new instances. Suppose you feed your life satisfaction model with many other attributes, including non-

informative attributes like country name.In this case, a complex model can detect patterns such as the fact that all countries in the training data with a w in their name have a life satisfaction greater than the value of x: New Zealand, Norway, Sweden, and Switzerland. . How sure are you that the rule of happiness is transferrable to Rwanda or Zimbabwe? Obviously, this pattern appeared purely by chance in the training data, but the model has no way of knowing if a pattern is real or just the result of noise in the data.

It is important to use a training set that is representative of the cases to which you want to generalise. This is often harder than it sounds: if the sample is too small, there will be sampling noise (i.e., non-representative data due to chance), but even exceptionally large samples may not be representative if the sampling method is flawed. . This is called sampling bias.

Overfitting occurs when the model is too complex relative to the amount and noise of the training data. Here are possible solutions:

- ❖ Simplify the model by choosing one with fewer parameters (for example, a linear model instead of a high-degree polynomial model), reduce the number of attributes in the training data, or constrain the model.
- ❖ Collect more training data.
- ❖ Reduce noise in training data (e.g. correct data errors and remove outliers).

Restricting a model to simplify it and reduce the risk of overfitting is called regularization. The amount of regularization to be applied in learning can be controlled by a hyperparameter. A hyperparameter is a parameter of a learning algorithm (not the model). Therefore, it is not affected by the learning algorithm itself; it must be set before training and remains constant during training. If you set the regularization hyperparameter to an exceptionally large value, you get an almost flat model (a slope close to zero); The learning algorithm will certainly not overfit the training data, but it is less likely to find a satisfactory solution. Optimising hyperparameters is an important part of building a machine learning system.

As you might have guessed, underfitting is the opposite of overfitting: it occurs when your model is too simple to learn the underlying structure of the data. For example, a linear model of life satisfaction tends to fail; Reality is simply more complex than the model, so their predictions are bound to be inaccurate, even with the training examples.

These are the main options to solve this problem:

- Choose a more powerful model with more parameters.
- Adding better functions to the learning algorithm (feature engineering).
- Reduce model constraints (e.g., by reducing the regularization hyperparameter).
- The only way to find out how well a model generalises to new cases is to test it on new cases. One way to achieve this is to put your model into production and monitor

Notes

its performance. This works fine, but if your model is horribly bad your users will complain, which is not the best idea.

A better option is to split your data into two sets: the training set and the test set. As these names imply, you train your model with the training set and test it with the test set. The error rate in the new cases is called the generalization error (or out-of-sample error). Evaluating your model against the test set will give you an estimate of this error. This value tells you how well your model works in cases you have never seen before.

- When the training error is small (p.That is, your model makes few errors in the training set), but the generalization error is high, which means your model is overfitting the training data.

Evaluating a model is easy: just use a test set. Suppose you are hesitating between two types of models (e.g., a linear model and a polynomial model): how do you decide between them? One way is to train both and compare how well they generalise using the test set.

Now suppose that the linear model generalises better, but you want to apply some regularization to avoid overfitting. The question is how to choose the value of the regularization hyperparameter? One possibility is to train 100 different models with 100 different values for this hyperparameter. Suppose you find the best hyperparameter value that produces a model with the lowest generalization error. Let us assume only 5% error. They approved this model for production, but unfortunately does not work as well as expected, generating 15% errors. What just happened

The problem is that you measured the generalization error several times on the test set and adjusted the model and hyperparameters to create the best model for that particular set. This means that the model will not work as well with new data.

A common solution to this problem is the so-called fallback validation: a part of the training set is simply offered to evaluate several candidate models and select the best one. The new retained set is called the validation set (or development set or development set).More specifically, it trains multiple models with multiple hyperparameters on the reduced training set (i.e., the full training set minus the validation set) and selects the model that performs best on the validation set. After this fallback validation process, you train the best model on the entire training set (including the validation set) to get the final model. Finally, this final model is evaluated against the test set to obtain an estimate of the generalization error.

However, if the validation set is too small, the model scores will be inaccurate: you may inadvertently choose a suboptimal model. Conversely, if the validation set is too large, the remaining training set will be much smaller than the entire training set. Why is that bad? Because the final model is trained on the entire training set, it is not ideal to compare candidate models trained on a much smaller training set. It would be like picking the fastest sprinter to run in a marathon. One way to solve this problem is to perform repeated cross-validation using many small validation sets. Each model is scored once per validation set after being trained on the rest of the data. By averaging all of a model's ratings, you get a much more accurate measurement of its performance.

3.5.3 Ridge Regression

Ridge regression is a model fitting method to analyse any data suffering from multicollinearity. When the multicollinearity problem occurs, the least squares are unbiased and the variance is large, causing the predicted values to differ widely from the

Notes

actual values. The L2 regularization, also called ridge regularization, adds a penalty term to the objective function that is proportional to the square of the model coefficients. This penalty term reduces the coefficients to zero, but in contrast to L1 regularization (lasso), does not make any of the coefficients exactly equal to zero, which can also be used for feature selection. L2 regularization can help reduce overfitting and improve model stability by keeping coefficients small. Both L1 and L2 regularization are commonly used to avoid overfitting and improve the generalizability of machine learning models.

3.5.4 Grid Search

Hyperparameter tuning involves inserting the training loop into an optimization method to find the optimal set of hyperparameters for the model.

In machine learning, model training is about finding the optimal set of intersections (in the case of decision trees), weights (in the case of neural networks), or support vectors (in the case of vector machines).We call these model parameters. However, in order to perform model training and find the optimal model parameters, we often need to program a variety of things. For example, we can decide that the maximum depth of a tree is 5 (in the case of decision trees), or that the activation function is ReLU (for neural networks) or choose the set of kernels we will use (in SVM). . These parameters are called hyperparameters.

Model parameters relate to the weights and biases your model has learned. You have no direct control over the model parameters, as these depend on the training data, the model architecture, and many other factors. In other words, you cannot set the model parameters manually. Your model weights are initialised with random values and then your model optimises them as it goes through training iterations. Hyperparameters, on the other hand, refer to any parameter that you as a builder can control. This includes values such as learning rate, number of epochs, number of layers in your model and more.

Because you can manually select values for different hyperparameters, your first instinct might be to try and error to find the optimal combination of hyperparameter values. This might work for models that are trained in seconds or minutes but can quickly become expensive for larger models that require a lot of infrastructure and training time. Imagine training an image classification model that takes hours to train on the GPU. It sets some hyperparameter values to test and then waits for the results of the first training run. Based on these results, the hyperparameters are changed, the model is retrained, the results are compared to the first run and then the best hyperparameter values are determined by looking at the training run with the best metrics.

There are some problems with this approach. First, you have spent a day and many computing hours on this task. Second, there is no way of knowing if you have reached the optimal combination of hyperparameter values. You only tried two different combinations and since you changed multiple values at once, you do not know which parameter had the biggest impact on performance. Even with additional testing, this approach quickly consumes time and computational resources and may not result in the optimal hyperparameter values.

A more structured version of the trial-and-error approach described above is known as grid search. When implementing hyperparameter optimization with grid search, we choose a list of values that we want to test for each hyperparameter that we want to optimise. In the grid search, we would test each combination of the given values and then use the combination that returned the best ranking metric in our model.

Notes

This grid search approach works well for small models, but for more complex models we would want optimization for more than two hyperparameters, each with a wide range of values. Grid search leads to a combinatorial explosion: adding additional hyperparameters and values to our selection grid increases the number of combinations we need to test and the time required to test them all.

Another problem with this approach is that no logic is applied when choosing different combinations. Grid search is a brute force solution, trying all combinations of values.

scikit-learn supports an alternative to grid search called RandomisedSearchCV that implements a random search. Instead of trying every combination of hyperparameters in an array, determine how often you want to randomly sample values for each hyperparameter. To implement random search in scikit-learn, we would instantiate RandomisedSearchCV and give it a dictation similar to grid_values above, where we specify ranges instead of specific values. Random search runs faster than grid search because not every combination in its set of values is tried. However, it is highly likely that the optimal set of hyperparameters will not be among the randomly selected ones.

3.5.5 Model Refinement

Once we have a concise list of promising models, we now need to refine and adjust them. Let us look at a few ways we can do that.

Grid Search

We earlier discussed the hyperparameters option, until you find a great combination of hyperparameter values. This would be very tedious work and you may not have time to explore many combinations.

Instead, you can use Scikit-Learn's GridSearchCV class to search for you. All you need to do is tell it which hyperparameters you want it to experiment with and what values to try out and it will use cross-validation to evaluate all the possible combinations of hyperparameter values.

If we have 3-fold cross validation and two dictionaries in this param_grid, GridSearchCV will first evaluate all 3 × 3 = 9 combinations of n_clusters and max_ features hyperparameter values specified in the first dict, then it will try all 2 × 3 = 6 combinations of hyperparameter values in the second dict. So, in total the grid search will explore 9 + 6 = 15 combinations of hyperparameter values and it will train the pipeline 3 times per combination. This means there will be a grand total of 15 × 3 = 45 rounds of training! It may take a while, but when it is done you can get the best combination of parameters like this:

You can access the best estimator using grid_search.best_estimator_. If GridSearchCV is initialised with refit=True (which is the default), then once it finds the best estimator using cross-validation, it retrains it on the whole training set. This is usually a clever idea, since feeding it more data will improve its performance.

The evaluation scores are available using grid_search.cv_results_. This is a dictionary, but if you wrap it in a DataFrame you get a nice list of all the test scores for each combination of hyperparameters and for each cross-validation split, as well as the mean test score across all splits:

Notes

Randomised Search

The grid search approach is fine when you are examining few combinations, as in the example above, but RandomisedSearchCV is often preferable, especially when the hyperparameter search space is large. This class can be used in the same way as the GridSearchCV class, but instead of trying every combination, it evaluates a fixed number of combinations and chooses a random value for each hyperparameter on each iteration. This may sound surprising, but this approach has several advantages:

- If some of your hyperparameters are continuous (or discrete but with many values) and you run the random search over one thousand iterations, for example, 1000 different values will be examined for each of those hyperparameters, while the grid search alone would do the trick. Explore the few values you listed for each.
- Suppose a hyperparameter does not make much of a difference, but you do not know it yet. If you have ten values and add them to your grid search, it takes 10 times as long to train. But if you add it to a random search, it makes no difference.
- When there are six hyperparameters to be examined, each with 10 values, grid search offers no choice but to train the model millions of times, while random search can always be run for any number of iterations.

HalvingGridSearchCV hyperparameter search classes. Your goal is to use computational resources more efficiently to either train faster or explore a larger hyperparameter space.

How they work: In the first round, many hyperparameter combinations (called “candidates”) are generated using either the grid approach or the random approach. These candidates are then used to train models, which are evaluated using cross-validation as usual.

However, the resources of the formation are limited, which significantly speeds up this first round. By default, “limited resources” means that models are trained on a small portion of the training set.

However, other restrictions are also possible, e.g., B. Reducing the number of training iterations when the model has a hyperparameter to set. Once all the candidates have been judged, only the top advance to the second round, where they have more resources at their disposal for the competition. After several rounds, the final candidates are evaluated using all resources. This can save you some time tweaking the hyperparameters.

Ensemble Methods

Another way to optimise your system is to try to combine the models that work best. The pool (or “set”) often performs better than the best single model, just as random forests perform better than the individual decision trees on which they are based, especially when the individual models make vastly different errors. For example, you could train and fit a k-nearest neighbour model and then create an ensemble model that predicts only the mean of the random forest prediction and that model’s prediction.

Analysing the Best Models and Their Errors

You will often gain good insights on the problem by inspecting the best models. The sklearn.feature_selection.SelectFromModel transformer can automatically drop the least useful features for you: when you fit it, it trains a model (typically a random forest), looks at its feature_importances_ attribute and selects the most useful features. Then when you call transform(), it drops the other features.

Notes

You should also look at the specific errors your system is making and then try to understand why it's making them and what might fix the problem: add extra features or remove non-meaningful features, clean up outliers, etc.

Now is also a good time to ensure that your model performs well not just on average, but in all district categories, whether rural or urban, rich, or poor, northern, or southern, minority or not, etc. . Breaking down your validation set for each category takes a bit of work, but it is important: if your model performs poorly in an entire district category, it should not be deployed until the problem is fixed, or at least you should not be used to predictions for this category as it can do more harm than good.

Evaluate Your System on the Test Set

After tweaking your models for a while, you finally have a system that works well enough. You are ready to evaluate the final model in the test suite. There is nothing special about this process; Just get the predictors and labels from your test suite and run your final model to transform the data and make predictions. You then evaluate these predictions: In general, if you have done a lot of hyperparameter tuning, the performance will be slightly worse than what you have measured using cross-validation. This is because your system is set up to work well with the validation data, but probably not so well with unknown datasets. This is not the case in this example as the test RMSE is lower than the validation RMSE. However, if this is the case, resist the temptation to change the hyperparameters to make the numbers look good on the test set. It is unlikely that improvements will transfer to new data.

Now comes the pre-launch phase of the project: you need to present your solution (highlighting what you learned, what worked and what did not, what assumptions were made and what limitations your system has) and document everything and create nice ones Presentations with clarity. easy-to-remember visualizations and statements.

Summary

- Analysing Data with Python involves using various libraries and tools to process, manipulate, visualise, and draw insights from data sets.
- Python has become a popular choice for data analysis due to its rich ecosystem of libraries and ease of use.
- Data Collection and Import: Data analysis starts with collecting relevant data. Python offers various methods to import data from various sources, such as CSV, Excel, databases, APIs, and web scraping. Libraries like Pandas and NumPy are commonly used for data manipulation.
- Data Cleaning and Preprocessing: Raw data often contains errors, missing values, and inconsistencies. Data cleaning involves handling these issues by removing duplicates, filling in missing values and transforming data into a suitable format for analysis.
- Data Exploration: Exploring data helps in understanding its structure and characteristics. Descriptive statistics, data visualization and basic plotting using libraries like Matplotlib and Seaborn provide insights into the data distribution, correlations, and outliers.
- Data Transformation and Feature Engineering: Feature engineering involves creating new features or transforming existing ones to improve the performance of machine learning models. Techniques like scaling, normalization and one-hot encoding are commonly applied.

Notes

Glossary

- Python: A high-level, versatile programming language widely used in data analysis, data science and various other domains.
- Data Analysis: The process of inspecting, cleaning, transforming, and modelling data to uncover useful information, patterns, and insights.
- NumPy: A Python library for numerical computing that provides support for arrays, matrices, and mathematical functions.
- pandas: An open-source Python library for data manipulation and analysis, offering data structures like DataFrames and Series.
- Matplotlib: A popular Python library for creating static, interactive and publication-quality visualizations.
- Seaborn: A data visualization library based on Matplotlib, providing a higher-level interface for creating attractive statistical graphics.

Check Your Understanding

1. What is the primary purpose of data analysis?
   a) Data visualization
   b) Extracting insights and patterns from data
   c) Data collection
   d) Data preprocessing
2. Which Python library is commonly used for numerical computing?
   a) pandas
   b) Matplotlib
   c) NumPy
   d) Seaborn
3. Which Python library is used for data manipulation and analysis, providing DataFrames and Series?
   a) NumPy
   b) pandas
   c) Matplotlib
   d) Seaborn
4. What is Matplotlib used for in data analysis?
   a) Data cleaning
   b) Data preprocessing
   c) Data visualization
   d) Machine learning
5. What is the primary purpose of data cleaning in data analysis?
   a) To extract patterns and insights from data
   b) To prepare data for visualization
   c) To identify errors and inconsistencies in the dataset
   d) To apply machine learning algorithms to data
6. What does EDA stand for in data analysis?
   a) Exploratory Data Analysis
   b) External Data Aggregation
   c) Efficient Data Analysis
   d) Exclusive Data Access
7. Which Python library provides a higher-level interface for creating statistical graphics?
   a) NumPy
   b) pandas
   c) Matplotlib
   d) Seaborn

Notes

8. What does Machine Learning involve?
   a) Analysing and visualising data
   b) Preprocessing and cleaning data
   c) Developing algorithms for computers to learn from data
   d) Creating interactive visualizations
9. Which Python library is widely used for machine learning tasks?
   a) NumPy
   b) pandas
   c) Matplotlib
   d) Scikit-Learn
10. What is the process of preparing data for analysis, including cleaning and feature engineering, called?
   a) Data visualization
   b) Data analysis
   c) Data preprocessing
   d) Data manipulation
11. Which technique is used for analysing and modelling data with temporal or sequential order?
   a) Time Series Analysis
   b) Regression Analysis
   c) Hypothesis Testing
   d) Clustering
12. What is unstructured data?
   a) Data that lacks any structure or organisation
   b) Data that is stored in a data warehouse
   c) Data with predefined relationships between tables
   d) Data that follows a predefined data model
13. What does data ethics promote in data analysis practices?
   a) Responsible, fair, and transparent data analysis
   b) Data collection from various sources
   c) Data visualization and reporting
   d) Data integration and aggregation
14. Which field combines scientific methods, statistics, and programming to extract insights from data?
   a) Data cleaning
   b) Data preprocessing
   c) Data visualization
   d) Data science
15. Which Python library is used for creating static, interactive and publication-quality visualizations?
   a) NumPy
   b) pandas
   c) Matplotlib
   d) Seaborn
16. What is the purpose of data visualization in data analysis?
   a) To prepare data for analysis
   b) To identify errors in the dataset
   c) To represent data in a visual format for better understanding
   d) To develop machine learning algorithms
17. What does “EDA” stand for in data analysis?
   a) Efficient Data Analysis
   b) Exclusive Data Access
   c) Exploratory Data Analysis
   d) External Data Aggregation

Notes

18. Which library provides data structures like DataFrames and Series in Python?
    a) NumPy b) pandas
    c) Matplotlib d) Seaborn
19. Which process involves filtering, aggregating, and transforming data to gain specific insights?
    a) Data visualization b) Data preprocessing
    c) Data manipulation d) Data cleaning
20. Which library is used for creating attractive statistical graphics in Python?
    a) NumPy b) pandas
    c) Matplotlib d) Seaborn

Exercise

1. What is Importing Data Sets?
2. Explain Python Package for Data Science.
3. What is Importing and Exporting Data in Python?
4. Describe Cleaning and Preparing the Data.
5. Explain Identify and Handle Missing Values.
6. What is Data Formatting?
7. What is ANOVA?
8. Explain Model Development.
9. What is Model Evaluation?
10. Describe Grid Search.

Learning Activities

1. Discuss Importing Data Sets with an example.
2. Describe Model Development of any company.

Check Your Understanding- Answers

1. b	2. c	3. b	4. c
5. c	6. a	7. d	8. c
9. d	10. c	11. a	12. a
13. a	14. d	15. c	16. c
17. c	18. b	19. c	20. d

Further Readings and Bibliography

1. “Introduction to Machine Learning” by Ethem Alpaydin
2. “Pattern Recognition and Machine Learning” by Christopher Bishop
3. “Machine Learning: A Probabilistic Perspective” by Kevin P. Murphy
4. “Machine Learning Yearning” by Andrew Ng
5. “Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow” by Aurélien Géron

Module - IV: Introduction to Data Analysis Process

Learning Objectives

At the end of this module, you will be able to:

- Learn data analysis and its process
- Know important terms in data analysis
- Describe conceptualising data analysis as a process
- Summarise managing the data analysis process
- Analyse procedures for data analysis
- Learn sources for data gathering
- Know data analysis using Matplotlib
- Describe file formats in data analysis

Introduction

Collecting and storing data for analysis is a very human activity. Systems for tracking grain stocks, taxes, and population date back thousands of years and the roots of statistics go back hundreds of years. Related disciplines, including statistical process control, operations research, and cybernetics, exploded in the 20th century.

4.1 Overview of Data Analysis and Its Process

Many different names are used to describe the data science discipline, such as business intelligence (BI), analytics, data science and decision science and practitioners have different job titles. Data analysis is also done by marketers, product managers, business analysts and many others. Often the terms “data analyst” and “data scientist” are used interchangeably to refer to someone who works with data to derive meaningful insights from data.

Data analysis in the modern sense enabled and is associated with the history of information technology. It is shaped by research and commercialization trends and contains information about researchers and large companies. Data analysis combines computing power with traditional statistical techniques. Data analysis is part data discovery, part data interpretation and part data reporting. The goal of data analysis is mostly to improve decision making by humans and increasingly also by machines through automation.

Notes

4.1.1 Introduction to Data Analysis

The 21st century is the information age. We live in the information age, which means every aspect of our daily lives generates data. In addition, business processes, government processes and social media posts also generate massive amounts of data. This data accumulates day by day as economic, governmental, scientific, technical, health, social, climate and environmental data is continuously generated. In all these areas of decision-making, we need a structured, generalised, efficient, and flexible analytical and scientific system so that we can gain insight into the data generated.

In today's intelligent world, data analysis drives effective decision-making in business and government. Data analysis is the process of examining, pre-processing, exploring, describing, and visualising a specific data set. The main goal of the data analysis process is to find the information needed for decision making. Data analysis offers many approaches, tools, and techniques, all applicable to different fields such as economics, social sciences, and fundamental sciences.

Comparing Data Analysis and Data Science

Data analysis is the process of examining data to discover patterns that help us make business decisions. It is one of the subfields of data science. Data analysis methods and tools are widely used in various business areas by business analysts, data scientists and researchers. Their main goal is to improve productivity and profits. Data analytics searches and extracts data from various sources, performs exploratory data analysis, visualises the data, prepares reports and presents them to business decision makers.

On the other hand, data science is an interdisciplinary field that uses a scientific approach to gain insights from structured and unstructured data. Data science is a combination of all terms including data science, data mining, machine learning and other related fields. Data science is not limited to just exploratory data analysis and is used to develop predictive models and algorithms such as stock prices, weather forecasts, diseases, fraud, and recommendations such as movie, book, and music recommendations. A data analyst collects, filters, processes, and applies the statistical concepts needed to capture patterns, trends, and insights from data and to prepare reports for decision making. The primary focus of a data scientist is to help organisations solve business problems using discovered patterns and trends. The data analyst also assesses data quality and addresses data collection issues. A data scientist should have knowledge of writing SQL queries, finding patterns and using visualization and reporting tools from Microsoft Power BI, IBM Cognos, Tableau, QlikView, Oracle BI, etc. Data analysts are more technical and mathematical than data analysts. Data analysts are research and education oriented while data analysts are more application oriented. Data scientists are expected to predict future events, while data scientists derive meaningful insights from data.Data scientists develop their own questions, while data scientists find answers to the questions asked. Finally, data scientists focus on what will happen next, while data scientists focus on what has happened so far.

Important Terms in Data Analysis

- –’’’’–’—• CSV – A comma separated values (CSV) file is a comma-delimited text file in which information is separated by commas and is common in spreadsheet applications.
- Data Analytics – Science that deals with the analysis of raw data in order to draw conclusions from it.

Notes

- Data architecture – defines the organisation's information flow for physical and logical data assets and governs how these are controlled. The goal of data architecture is to translate business requirements into data and system requirements and to manage data and its flow through the enterprise.
- Acquisition – the process of gathering information and then converting it into data that can be read by a computer.
- Directory of data: A detailed list of all of your organisation's data assets. Use metadata to help data professionals quickly find, access, and interpret the most relevant data for any analysis or business purpose.
- Cleansing – the process of preparing data for analysis by altering or removing invalid, corrupted, malformed, duplicated, irrelevant, or incomplete data in a data set.
- Data Enrichment – The process of enriching, adding, augmenting, and enriching the collected data with relevant data from third parties.
- Data Governance – A system for defining the people, processes and technologies required to manage, organise, and protect corporate data assets.
- Data integrity: The integrity of enterprise data and its ability to support effective business objectives.
- Data hygiene: Continuous processes ensure that the data is clean and ready for use.
- Data import: process of transferring data from external sources to another application or database.
- Data mining – the process of transferring data from multiple sources to a central database, usually a data warehouse, where it can be viewed and analysed. This can be done in a real-time stream or in batches.
- Data integrity – is the overall accuracy, consistency, and reliability of data throughout its life cycle.
- Data Intelligence – the process of analysing various forms of data to improve a company's services or investments.• Data mapping: The process of matching data fields from one or more source files with data fields from another source.
- Data Masking – is a privacy technique that copies a record but masks sensitive data. Also known as data obfuscation.
- Data migration: the process of moving internal data between different types of file formats, databases, or storage systems.
- Data Mining – The process of discovering anomalies, patterns, and correlations in large amounts of data to solve problems through data analysis.
- Data modelling – the process of visualising and representing data elements and the relationships between them.
- Data munging – the process of preparing for data transformation and cleaning up large data sets before analysis.
- Data Replication: The process of storing data in multiple locations to improve data availability, reliability, redundancy, and availability.
- Data Quaiity – A measure of the reliability of a data set to meet the organisation's specific needs, based on factors such as accuracy, completeness, consistency, reliability, and timeliness.
- Data Science – A multidisciplinary approach to deriving actionable insights from the vast and ever-expanding amounts of data collected and created by today's organisations.

Notes

- Data security – the practice of protecting data from unauthorised access, theft, or damage throughout its lifecycle.
- Data transfer – Any information transferred or shared between systems or organisations.
- Data Validation: Ensure data accuracy and quality against established rules before data is used, imported, or otherwise processed.
- Data warehouse – a repository of structured and filtered data that has already been processed for a specific purpose. Examples include BigQuery, Redshift and Snowflake.
- Data wrangling – The process of restructuring, cleaning, and enriching raw data into a desired format for easier access and analysis.
- Database – an organised collection of information or structured data, usually stored electronically in a computer system for ease of access, management and updating. Examples include MySQL, PostgreSQL, Microsoft SQL Server, MongoDB, Oracle Database and Redis.
- Database schema – is a set of metadata that describes the relationships between objects and information in the database. It is the blueprint or architecture for how our data will look like.
- Data Set: An ordered collection of discrete but related items that can be viewed and processed individually or as a whole.
- Dummy Data – Dummy data that has the same content and layout as the real data in the test environment.
- ETL – stands for Extract, Transform and Load. ETL collects and processes data from different sources in a single data warehouse, which makes analysis much easier.
- JSON – JavaScript Object Notation (JSON) is a human-readable text data interchange format for storing and transmitting data.
- Master Data Management (MDM): Technology, tools, and processes to ensure the consistency, uniformity, and accuracy of an organisation's data on customers, products, suppliers, and other business partners.
- NoSQL – a non-relational database that stores and retrieves data without first defining its structure – an alternative to a more rigid relational database model. Instead of storing data in rows and columns like in a traditional database, a NoSQL database stores each item individually with a unique key.
- Raw data – a collection of information obtained from various sources but not processed, cleaned, or analysed.
- Relational database (RDBMS) – a type of database in which data is stored in the form of tables and tables are linked based on defined relationships. The most common means of data access for RDBMS is SQL.
- Data flow: the continuous flow of data generated from various sources to its destination for processing and analysis in near real time.
- TSV – Tab Separated Values – Files are used to store raw data and are often used by spreadsheets to exchange data between databases.
- Unstructured data: Data sets (typically large amounts of files) that are not arranged in a predefined data model or schema.

Notes

- XML (Extensible Markup Language) – a simple and extremely flexible markup language for storing and transporting data. Describes the text of a digital document.

4.1.3 Conceptualising Data Analysis as a Process

Data analysis is about looking at data, deriving key insights and drawing conclusions. The main goal of this process is to collect, filter, clean, transform, explore, annotate, visualise, and inform this data to generate insights that facilitate decision-making. In general, the data analysis process consists of the following phases:

- ❖ Data Collection: Collect and compile data from multiple sources.
- ❖ Data preprocessing: Filter, cleanse and convert data into the required format.
- ❖ Analyse and find insights: Explore, annotate and visualise data and find insights and insights.
- ❖ Interpret insights: Understand insights and determine the impact of each variable on the system.
- ❖ Storytelling: Present your achievements in the form of a story so that laypeople can understand them.

Data analysis is a range of ways to disaggregate data, assess trends over time and compare one industry or activity to another. It can also include many ways of data visualization so that trends and relationships are intuitively recognizable at a glance. Data analysis is about asking questions about what happened, what is happening and what will happen (predictive analytics). Analysis is typically the data processing, question-answering and decision-making phase of the overall business intelligence process.

4.1.4 Managing the Data Analysis Process

While every organisation has its own data needs and goals, there are key steps that remain consistent across all organisations and their data analysis processes:

- Set Goals: Set goals for your data science teams to develop a measurable way to determine if the organisation is making progress toward its goals. Identify performance metrics or indicators in a timely manner
- Identify business levers: Identify goals, metrics, and levers in the early stages of data analysis projects to ensure scope and focus of data analysis; This means that the company must be willing to make changes to improve its metrics and meet its goals
- Data Collection – Collect as much data as possible from different sources to build better models and get more useful information
- Data cleansing: improve data quality to get the right results and avoid wrong conclusions; Automate the process but assign staff to oversee data cleansing and ensure accuracy
- Expand your data science team: Include in your science team people with advanced

Notes

degrees in statistics who focus on data modelling and forecasting, as well as infrastructure engineers, developers and ETL professionals; Then give your team the rich data analysis platforms they need to automate data collection and analysis

- Refine and Iterate: Refine your data analysis model so you can iterate over the process to produce accurate forecasts, meet goals, monitor consistently, and generate reports.

4.1.5 Procedures for Data Analysis: Quantitative

The procedure for quantitative data analysis includes the following steps:

- Data preparation: The first step in data analysis is data preparation, the purpose of which is to transform the raw data into something useful and readable. It includes four phases:
- Data Validation: The purpose of data validation is to determine, as far as possible, whether the data has been collected according to pre-established standards and without bias.
- Data change: Large data sets usually contain errors. To ensure there are no such errors, the analyst must perform background data checks, check for outliers, and manipulate the raw data to identify and remove any data points that could affect the accuracy of the analysis results.
- Data Coding: refers to the grouping and categorisation of data.

After these steps, the data is ready for analysis. The two most commonly used methods for analysing quantitative data are descriptive statistics and inferential statistics.

Descriptive Statistics

In general, descriptive statistics (also called descriptive analysis) is the first level of analysis. Help scientists summarise data and find patterns. Some commonly used descriptive statistics are:

- Mean: numerical average of a set of values.
- Median: midpoint of a set of numerical values.
- Mode: most common value among a set of values.
- Percentage: used to express how a value or group of respondents within the data relates to a larger group of respondents.
- Frequency: the number of times a value is found.
- Range: the highest and lowest value in a set of values.

Descriptive statistics give absolute numbers. However, they do not explain the rationale or justification for these numbers. Because descriptive analysis is often used to analyse single variables, it is often referred to as univariate analysis. Inferential analysis shows the relationship between multiple variables using correlation, regression, or analysis of variance.

Correlation

The interaction between two variables is described by the correlation between them and can be expressed by various statistical measures. When high values in one variable are often associated with high or low values in another variable, that variable is said to be strongly correlated. Although there are many different statistical measures that express the strength of this relationship, there is one known as the Pearson coefficient.

Notes

Regression

Another way to show the relationship between two variables is with regression analysis. This technique can be used to describe relationships between more than two variables, assuming that one of the variables is dependent on the other variables or is predicted from the other variables.The target variable is called the dependent variable and the other variables are called the independent variables. The calculations in this model assume that all independent variables are uncorrelated and the correlation technique described earlier in this module can be used to test this condition.

There are many models that represent relationships between variables, so called general linear models. The simplest of these is the linear egression model, which relates the dependent variable to a linear combination of the parameters of the independent variable. The regression model calculations consist of finding these constant parameters and analysing how well they fit the data.

Often, the dependent and independent variable data must be transformed to meet the model requirements and the analyst must then return to the data manipulation tasks.

4.1.6 Procedures for Data Analysis: Qualitative

In contrast to quantitative data, qualitative data are descriptive and expressed in language rather than numerical values. Qualitative data analysis describes information and cannot be measured or counted. Refers to words or terms used to describe certain characteristics or characteristics.

To answer the question «Why?» » Would you use qualitative data or what?». It is often used to study open-ended research that allows participants (or clients) to express their true feelings and actions without guidance. In-depth interviews, focus groups, or observation are popular methods of data collection.

Analysing qualitative data works a little differently than analysing quantitative data, mainly because qualitative data consists of words, observations, images and even symbols. It is almost impossible to derive absolute meaning from such dates; It is therefore mainly used for exploratory research. While in quantitative research there is a clear distinction between the data preparation phase and the data analysis phase, in qualitative research the analysis often begins as soon as the data are available.

The analysis and processing take place in parallel and include the following process steps:

1. Become familiar with the data: Because most qualitative data is just words, a researcher should first read the data several times to become familiar with it and look for underlying observations or patterns. This also includes the transcription of data.
2. Review of Research Goal: Here the researcher reviews the research goal and identifies questions that can be answered with the collected data.
3. Framework development: Also called coding or indexing, the researcher identifies and assigns codes to ideas, concepts, behaviours, or general expressions. For example, encoding age, gender, socioeconomic status and even concepts like a yes or no answer to a question. Coding is useful for structuring and labelling data.
4. Identify Patterns and Connections: Once your data has been coded, you can begin your research by identifying topics, looking for general answers to questions, identifying data or patterns that might answer your research questions and finding areas to explore further can.

Notes

There are different methods for analysing qualitative data. The most commonly used data analysis meth-ods include:

- Content Analysis: This is one of the most common methods of analysing qualitative data. It is used to analyse documented information in the form of text, media, and even physical objects. When to use this method depends on the research questions. Content analysis is typically used to analyse respondent responses.
- Narrative analysis: This method is used to analyse the content of various sources, such as interviews with respondents, field observations or surveys. The focus is on using people's shared stories and experiences to answer research questions.
- Discourse analysis: Like narrative analysis, discourse analysis is used to analyse interactions with people. However, the focus is on analysing the social context in which the communication between the researcher and the respondent took place. The discourse analysis also examines the everyday environment of the respondent and uses this information in the analysis.
- Grounded Theory: refers to the use of qualitative data to explain why a particular phenomenon occurred. For this purpose, many similar cases are examined in different contexts and causal explanations are derived from the data. Scientists can change explanations or create new ones by examining more cases until they arrive at an explanation that works for all cases.

Qualitative data analysis works a little differently than quantitative data, mainly because qualitative data consists of words, observations, images and even symbols. It is almost impossible to derive absolute meaning from such dates; Therefore, it is used for exploratory analysis. While in quantitative analysis there is a clear distinction between the data preparation phase and the data analysis phase, for qualitative research analysis often begins as soon as the data is available.

4.1.7 Sources for Data Gathering: Machine and File Data Sources

Informally, "data" can be thought of as a collection of symbols representing a set of measurements or observations about an event or occurrence. Other meanings can include lists of «facts» or «statistics,» although any collection of words, documents, web pages and emails can also be considered data. Some of this data is designed and collected intentionally, as in scientific research, but other data can be considered "accidental" – nobody intentionally conceived Twitter as a formal data collection system, but today it has grown into a rich mine of useful data Knowledge. about the political mood and even a source of information about public health and epidemics.

Machine Data Sources

The machine data source is created on the client machine, whether it is a computer, phone, IoT or other device. It is available to users currently logged into the system and cannot be shared with other computers. They can be divided into user data sources (available only to a specific user) and system data sources (available to all system users).

Examples of machine data sources include network traffic logs, system and application logs, sensor outputs, event data from IoT devices, database query results and more.

Users must use the data source name as a shortcut to connect to data.

Notes

File Data Sources

The data sources for files are not associated with any specific machine, application, system, or user. They can be shared between devices. These data sources are usually stored in separate text files. They do not have a data source name (DSN) like machine data sources. These data sources include spreadsheets, text documents, PDFs, images and audio and video files.

A file is a sequence of bytes that is typically stored on disk. Applications often write data to files. Files can save locale, events, logs, images, and audio.

In addition, files are a universal medium for exchanging data. While data engineers wish they could retrieve data programmatically, much of the world still sends and receives files. For example, if you receive your data from a government agency, you will download it as an Excel or CSV file or receive it by email.

The main types of source file formats you will encounter as a data engineer (files created manually or by a source system process) are Excel, CSV, TXT, JSON, and XML. These files have their own peculiarities and can be structured (Excel, CSV), semi-structured (JSON, XML, CSV) or unstructured (TXT, CSV). Note that a file format that does not enforce a specific content structure can be used for both structured and unstructured data.

Some common data collection sources are described below. Depending on the source design and data type, many of these sources can generate structured, semi-structured, or even unstructured data:

APIs

APIs (Application Programming Interfaces) are the standard way of exchanging data between systems. In theory, APIs make data mining easier for data engineers. In practice, many APIs still present engineers with significant data complexity. Even with the emergence of numerous services, platforms and services that automate API data retrieval, data engineers often have to invest a lot of energy in maintaining custom API connections. We will cover the APIs in more detail later in this module.

Application Databases (OLTP Systems)

The application database saves the state of the application. A common example is a database that stores bank account balances. As customers complete transactions and pay, the app updates their bank account balance.

In general, an application database is an online transaction processing system (OLTP), a database that reads and writes single records at high speed. OLTP systems are often referred to as transactional databases, but that does not necessarily mean the system supports atomic transactions.

More generally, OLTP databases support low latency and high concurrency. An RDBMS database can select or update a row in less than a millisecond (not counting network latency) and handle thousands of reads and writes per second. A document database cluster can handle even higher document approval rates, but at the cost of potential inconsistencies. Some graph databases can also support transactional use cases.

Online Analytical Processing System

Unlike an OLTP system, an online analytical processing (OLAP) system is designed to handle large analytical queries and is inefficient when searching for individual records.

Notes

For example, modern database columns are optimised for analysing copious amounts of data and ignore indexes to improve scalability and analytical performance. Each query typically analyses a minimal block of data, often 100 MB or more. Attempting to fetch thousands of individual items per second on such a system would bring it to its knees if not combined with a caching layer designed for this use case.

Logs

The log records information about events occurring on systems. For example, the log can capture traffic and usage patterns on a web server. The desktop operating system "Windows, macOS, Linux" logs events such as system start and application start or crashes. Logs are a rich source of data that is potentially valuable for further data analysis, machine learning and automation. Here are some known log sources:

- Operating systems
- Applications
- Servers
- Databases
- Containers
- Networks
- IoT devices

All logs track events and event metadata.

Third-Party Data Sources

The consumerization of technology means that every company today is a technology company. This means that these companies and increasingly government agencies, want to make their data available to their customers and users either as part of their services or as part of a separate subscription. For example, NASA publishes various data related to its research initiatives. Facebook shares data with companies that advertise on its platform.

Why do companies want to share their data? Data sticks and the steering wheel is built by allowing users to integrate and extend their apps within user apps. Higher user adoption and usage means more data, which means users can integrate more data into their applications and data systems. The side effect is that there are now almost endless third-party data sources.

Direct access to third-party data is usually via APIs, data exchange on a cloud platform or data upload.APIs often provide deep integrations that allow clients to retrieve and transmit data. For example, many CRMs offer APIs that their users can integrate with their systems and applications. We see a common workflow to pull data from the CRM, combine CRM data with a customer scoring model and then send that data to the CRM using reverse ETL so sales reps can reach more qualified leads.

4.1.8 Data Analysis Using Matplotlib: Part - I

For 2D array charts, Matplotlib is an excellent Python visualisation library. Matplotlib is a multi-platform data visualisation package based on NumPy arrays that is designed to work with the entire SciPy stack. John Hunter was the first to launch it in 2002.

One of the most important benefits of visualisation is that it allows us to see large amounts of data in easily understood images. Line, bar, scatter, histogram, and more graphs are available in Matplotlib.

Notes

In Windows, Linux and macOS distributions, Matplotlib and most of its dependencies are accessible as wheel packages. Use the following command to install the matplotlib package:

```
python -mpip install -U matplotlib
```

Importing matplotlib :

```
from matplotlib import pyplot as plt
```

or

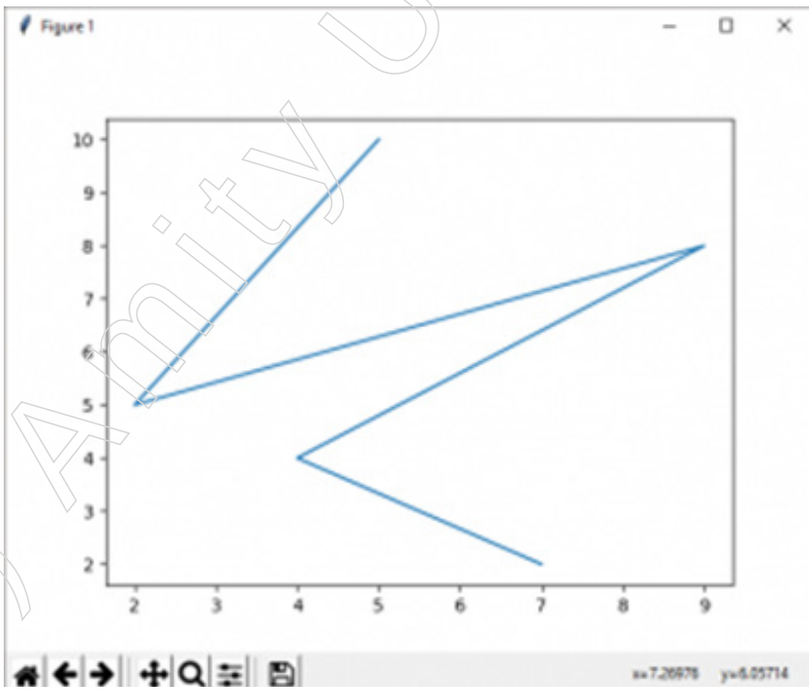
```
import matplotlib.pyplot as plt
```

Basic plots in Matplotlib :

Matplotlib has a large number of plots. Plots aid in the understanding of trends, patterns, and relationships. They're often used to make decisions based on numerical data. Some of the sample plots are discussed in this section.

```
# importing matplotlib module
from matplotlib import pyplot as plt
# x-axis values
x = [5, 2, 9, 4, 7]
# Y-axis values
y = [10, 5, 8, 4, 2]
# Function to plot
plt.plot(x,y)
# function to show the plot
plt.show()
```

Output:

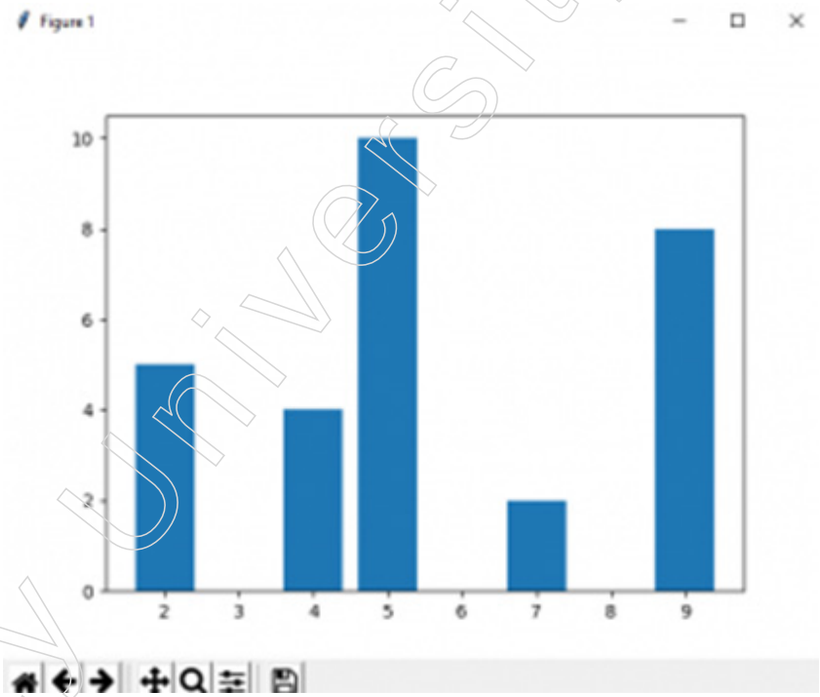


Notes

Bar plot:

```
# importing matplotlib module
from matplotlib import pyplot as plt
# x-axis values
x = [5, 2, 9, 4, 7]
# Y-axis values
y = [10, 5, 8, 4, 2]
# Function to plot the bar
plt.bar(x,y)
# function to show the plot
plt.show()
```

Output

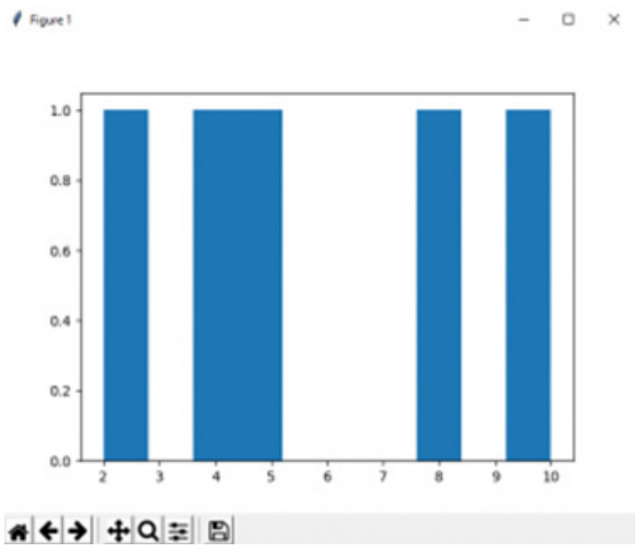


Histogram

```
# importing matplotlib module
from matplotlib import pyplot as plt
# Y-axis values
y = [10, 5, 8, 4, 2]
# Function to plot histogram
plt.hist(y)
# Function to show the plot
plt.show()
```

Output:

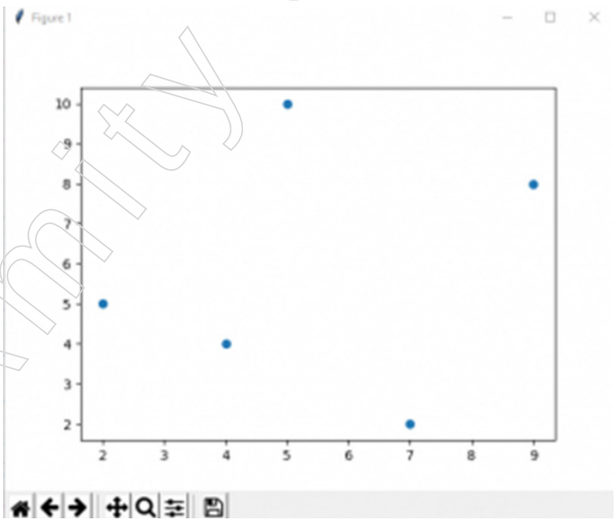
Notes



Scatter Plot:

```
# importing matplotlib module
from matplotlib import pyplot as plt
# x-axis values
x = [5, 2, 9, 4, 7]
# Y-axis values
y = [10, 5, 8, 4, 2]
# Function to plot scatter
plt.scatter(x, y)
# function to show the plot
plt.show()
```

Output:



Single-Line Plots

A single-line plot is one in which there is only one visualisation in a figure that employs the plot() function. You'll see a couple different approaches to draw a single-line

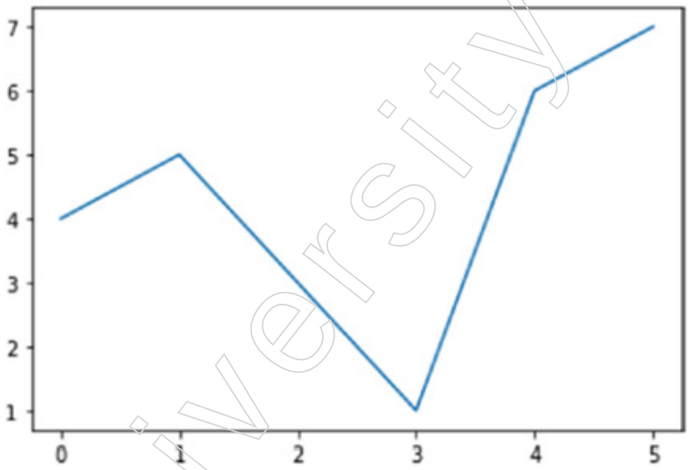
Notes

plot in this section. We've already used the plot() function to create single-line plots. Let's take a closer look at this topic with a few more excellent examples.

Make a new notebook for this chapter's demonstrations. You can also visualise the plots with Python lists, as shown below:

```
%matplotlib inline
import matplotlib.pyplot as plt
x = [4, 5, 3, 1, 6, 7]
plt.plot(x)
plt.show()
```

output

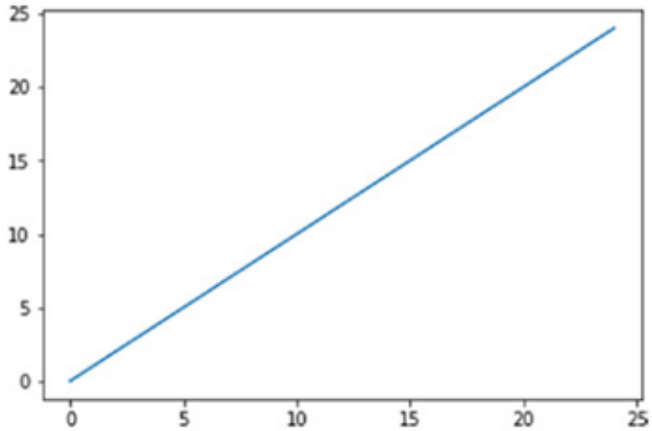


The values of the y-axis are assumed in this situation.

Here's another single-line graph that makes use of a Ndarray:

```
import numpy as np
x = np.arange(25)
plt.plot(x)
plt.show()
```

Output

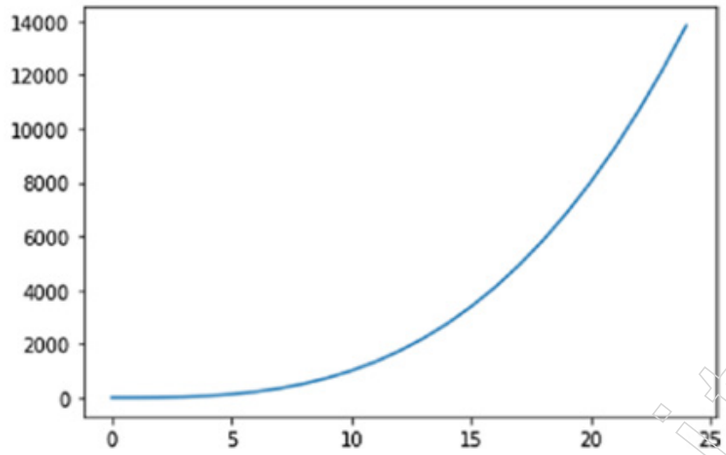


Let's have a look at the quadratic graph y = f(x) = x^3 +1. The following is the code:

Notes

```
plt.plot(x, [(y**3 + 1) for y in x])
plt.show()
```

output:



The following is an easy approach to write the same code:

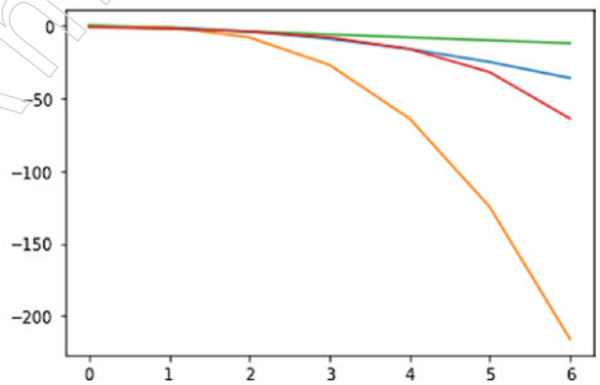
```
plt.plot(x, x**3 + 1)
plt.show()
```

Multiline Plots

It's possible to see numerous charts in a single output. Let's look at how to combine many curves into a single visualisation. A basic example is as follows:

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(7)
plt.plot(x, -x**2)
plt.plot(x, -x**3)
plt.plot(x, -2*x)
plt.plot(x, -2**x)
plt.show()
```

output



Notes

The following is an easy approach to write the same code:

```
plt.plot(x, -x**2, x, -x**3,
 x, -2*x, x, -2**x)
plt.show()
```

The output will be identical to the previous output.

4.1.9 Data Analysis Using Matplotlib: Part – II

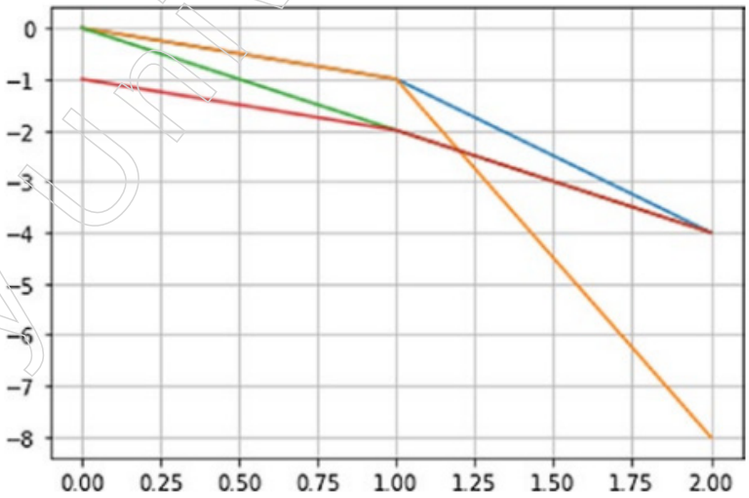
Grid, Axes and Labels

You'll now learn how to use a grid in your visualisations. The plt.grid (True) statement can be used to do this. You'll also learn how to alter axes' boundaries. But first, you'll learn how to save a visualisation to your hard drive as a picture. Take a look at the code below:

```
x = np.arange(3)
plt.plot(x, -x**2, x, -x**3, x, -2*x, x, -2**x)
plt.grid(True)
plt.savefig('test.png')
plt.show()
```

The command plt.savefig('test.png') saves the image in the Jupyter Notebook file's current directory.

Output

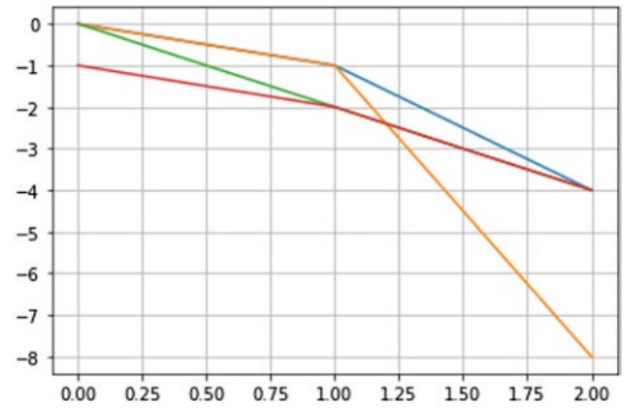


As you can see, the axes' bounds are set by default as follows:

```
x = np.arange(3)
plt.plot(x, -x**2, x, -x**3, x, -2*x, x, -2**x)
plt.grid(True)
print(plt.axis())
plt.show()
```

Notes

```
(-0.1, 2.1, -8.4, 0.4)
```



You can also change the values of the axes by doing the following:

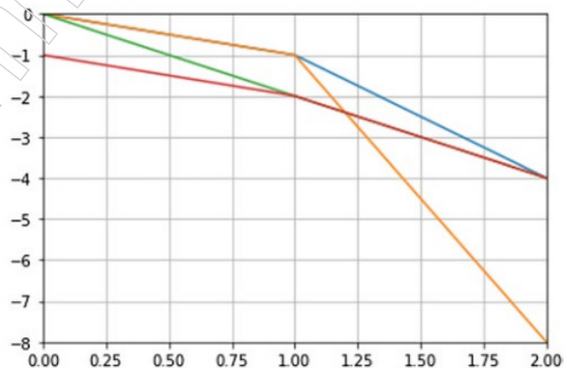
```
x = np.arange(3)
plt.plot(x, -x**2, x, -x**3, x, -2*x, x, -2**x)
plt.grid(True)
plt.axis([0, 2, -8, 0])
print(plt.axis())
plt.show()
```

The axes' values are set using the plt.axis([0, 2, -8, 0]) statement. The first pair, (0, 2), represents the x-axis limits, whereas the second pair, (-8, 0), represents the y-axis limits. Using the functions xlim() and ylim(), you may write the following code with a different syntax:

```
x = np.arange(3)
plt.plot(x, -x**2, x, -x**3, x, -2*x, x, -2**x)
plt.grid(True)
plt.xlim([0, 2])
plt.ylim([-8, 0])
print(plt.axis())
plt.show()
```

output:

```
(0.0, 2.0, -8.0, 0.0)
```

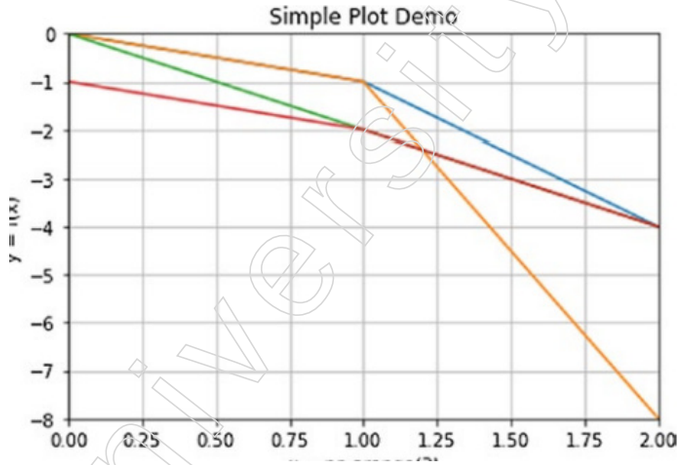


Notes

The title and labels for the axes can be added as follows:

```
x = np.arange(3)
plt.plot(x, -x**2, x, -x**3, x, -2*x, x, -2**x)
plt.grid(True)
plt.xlabel('x = np.arange(3)')
plt.xlim([0, 2])
plt.ylabel('y = f(x)')
plt.ylim([-8, 0])
plt.title('Simple Plot Demo')
plt.show()
```

This generates output with the labels and title visible.

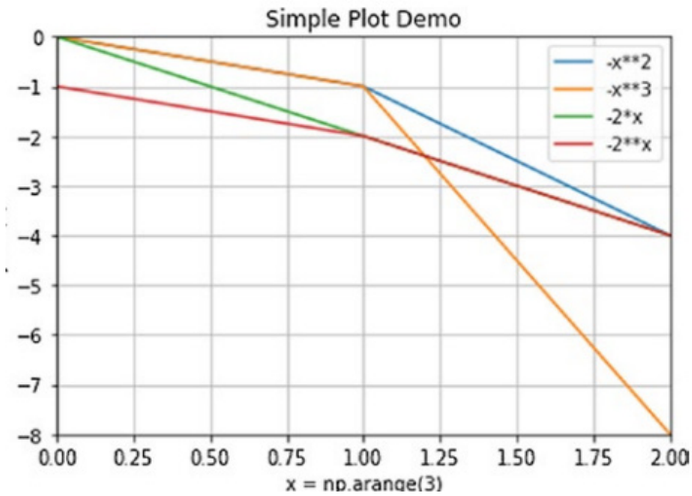


You can construct a legend by passing an argument for the parameter label to the plot() function and then calling the function legend() as follows:

```
x = np.arange(3)
plt.plot(x, -x**2, label='-x**2')
plt.plot(x, -x**3, label='-x**3')
plt.plot(x, -2*x, label='-2*x')
plt.plot(x, -2**x, label='-2**x')
plt.legend()
plt.grid(True)
plt.xlabel('x = np.arange(3)')
plt.xlim([0, 2])
plt.ylabel('y = f(x)')
plt.ylim([-8, 0])
plt.title('Simple Plot Demo')
plt.show()
```

This function generates output with curve legends,

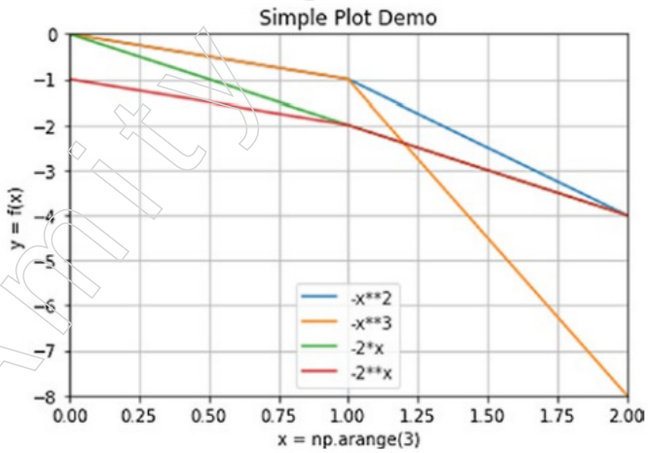
Notes



Rather than sending the legend string as a parameter to plot(), you can pass a list of strings to legend() as follows:

```
x = np.arange(3)
plt.plot(x, -x**2, x, -x**3, x, -2*x, x, -2**x)
plt.legend(['-x**2', '-x**3', '-2*x', '-2**x'])
plt.grid(True)
plt.xlabel('x = np.arange(3)')
plt.xlim([0, 2])
plt.ylabel('y = f(x)')
plt.ylim([-8, 0])
plt.title('Simple Plot Demo')
plt.show()
```

Output:




```
x = np.arange(3)
plt.plot(x, -x**2, x, -x**3, x, -2*x, x, -2**x)
plt.legend(['-x**2', '-x**3', '-2*x', '-2**x'],
 loc='lower center')
```

Notes

```
plt.grid(True)
plt.xlabel('x = np.arange(3)')
plt.xlim([0, 2])
plt.ylabel('y = f(x)')
plt.ylim([-8, 0])
plt.title('Simple Plot Demo')
plt.savefig('test.png')
plt.show()
```

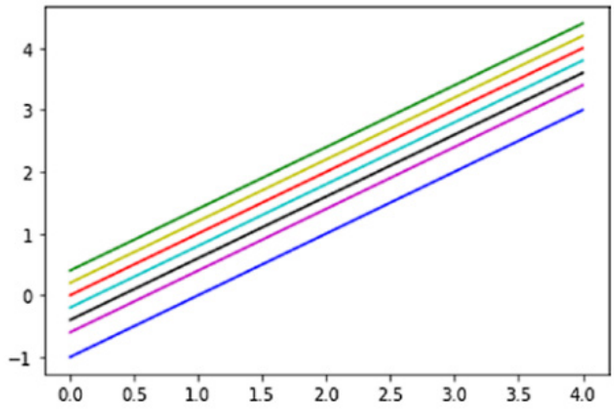
Colors, Styles and Markers

You've seen how Matplotlib assigns colours, styles, and markers to multiline plots automatically up until now. You've seen some instances of how to personalise them. Now, in this section, you'll discover exactly how to customise them.

Let's begin with the colours. The following code lists all of Matplotlib's primary colours (we won't be changing styles or markers in this example):

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(5)
y = x
plt.plot(x, y+0.4, 'g')
plt.plot(x, y+0.2, 'y')
plt.plot(x, y, 'r')
plt.plot(x, y-0.2, 'c')
plt.plot(x, y-0.4, 'k')
plt.plot(x, y-0.6, 'm')
plt.plot(x, y-0.8, 'w')
plt.plot(x, y-1, 'b')
plt.show()
```

Output



Notes

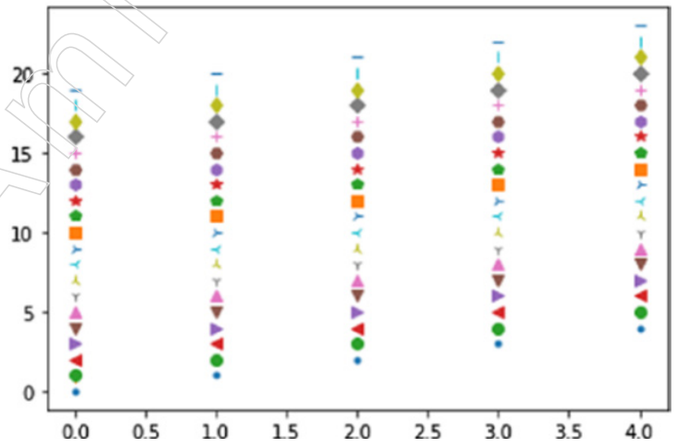
The preceding code can also be written as follows:

```
plt.plot(x, y+0.4, 'g', x, y+0.2, 'y', x, y, 'r', x, y-0.2, 'c', x, y-0.4,
'k', x, y-0.6, 'm', x, y-0.8, 'w', x, y-1, 'b')
plt.show()
```

The output will be identical to that shown above.

You can even alter the marks by doing so:

```
plt.plot(x, y, '.')
plt.plot(x, y+0.5, ',')
plt.plot(x, y+1, 'o')
plt.plot(x, y+2, '<')
plt.plot(x, y+3, '>')
plt.plot(x, y+4, 'v')
plt.plot(x, y+5, '^')
plt.plot(x, y+6, '1')
plt.plot(x, y+7, '2')
plt.plot(x, y+8, '3')
plt.plot(x, y+9, '4')
plt.plot(x, y+10, 's')
plt.plot(x, y+11, 'p')
plt.plot(x, y+12, '*')
plt.plot(x, y+13, 'h')
plt.plot(x, y+14, 'H')
plt.plot(x, y+15, '+')
plt.plot(x, y+16, 'D')
plt.plot(x, y+17, 'd')
plt.plot(x, y+18, '|')
plt.plot(x, y+19, '_')
plt.show()
```

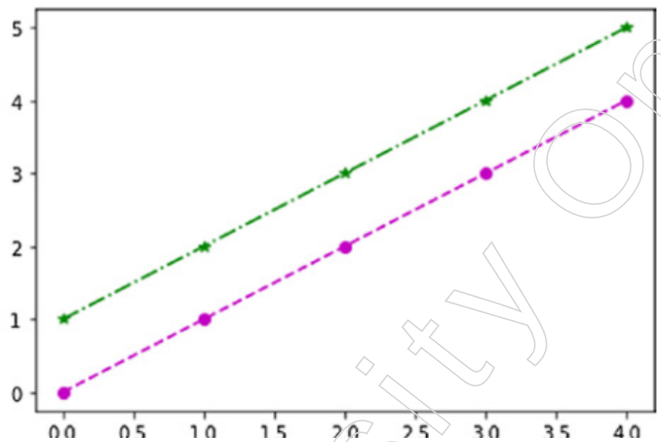


You can personalise the visualisation by combining all three techniques (for colours,

Notes

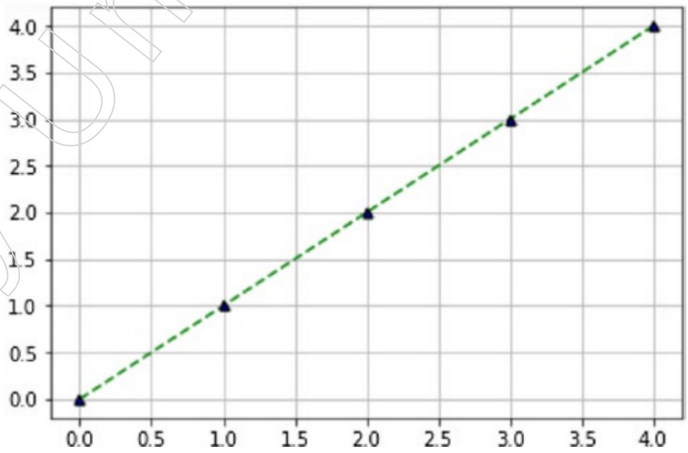
markers, and line styles) as follows:

```
plt.plot(x, y, 'mo--')
plt.plot(x, y+1 , 'g*-.')
plt.show()
```



These are the basic customizations that Matplotlib allows you to make. Everything can be customised in great detail. Here's an example of code:

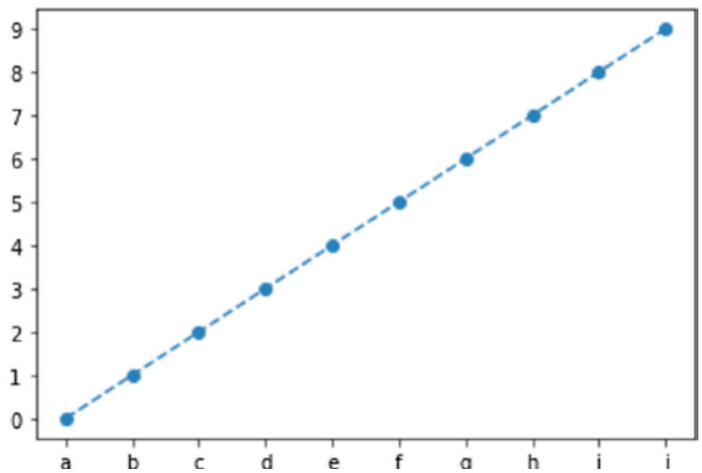
```
plt.plot(x, y, color='g', linestyle='--', linewidth=1.5,
 marker='^', markerfacecolor='b', markeredgecolor='k',
markeredgewidth=1.5, markersize=5)
plt.grid(True)
plt.show()
```



You may also change the values on the x- and y-axes by doing the following:

```
x = y = np.arange(10)
plt.plot(x, y, 'o--')
plt.xticks(range(len(x)), ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'])
plt.yticks(range(0, 10, 1))
plt.show()
```

Notes



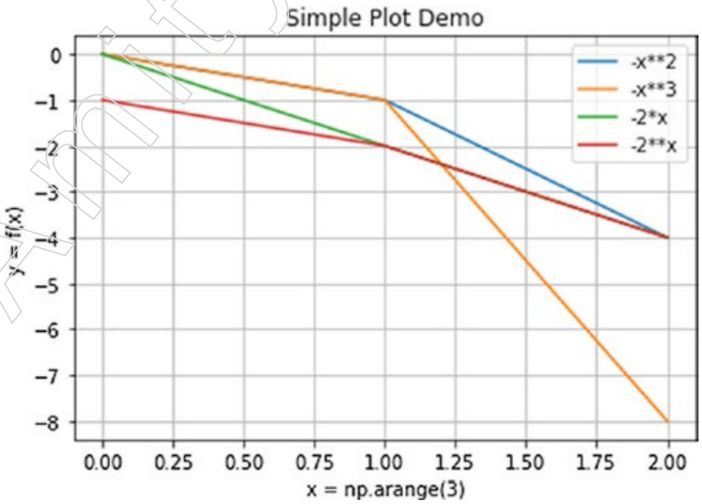
Object-Oriented Plotting

Object-oriented charts are possible to make. Let's redo one of our previous code samples in the following manner:

```
fig, ax = plt.subplots()
ax.plot(x, -x**2, label='-x**2')
ax.plot(x, -x**3, label='-x**3')
ax.plot(x, -2*x, label='-2*x')
ax.plot(x, -2**x, label='-2**x')
ax.set_xlabel('x = np.arange(3)')
ax.set_ylabel('y = f(x)')
ax.set_title('Simple Plot Demo')
ax.legend()
ax.grid(True)
plt.show()
```

It's worth noting that we're plotting and setting labels and a title with the axis object.

Output



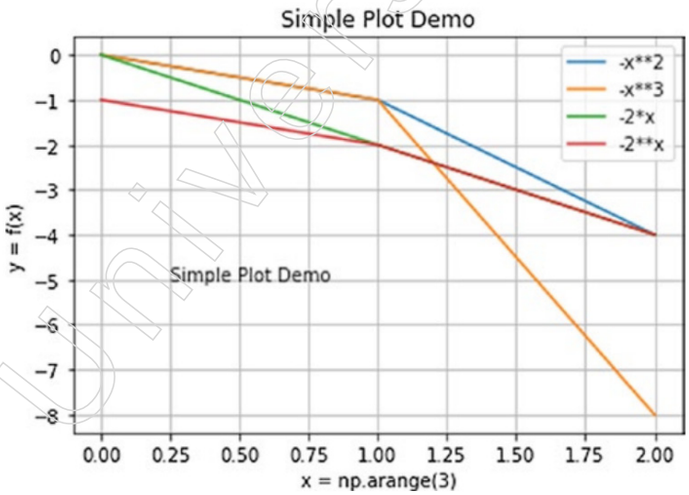
You can also use the functions ax.text() and plt.text() to add text (). The coordinates

Notes

and the text to be displayed are passed to the routines. As an example, consider the following:

```
fig, ax = plt.subplots()
ax.plot(x, -x**2, label='-x**2')
ax.plot(x, -x**3, label='-x**3')
ax.plot(x, -2*x, label='-2*x')
ax.plot(x, -2**x, label='-2**x')
ax.set_xlabel('x = np.arange(3)')
ax.set_ylabel('y = f(x)')
ax.set_title('Simple Plot Demo')
ax.legend()
ax.grid(True)
ax.text(0.25, -5, "Simple Plot Demo")
plt.show()
```

Output



Subplots

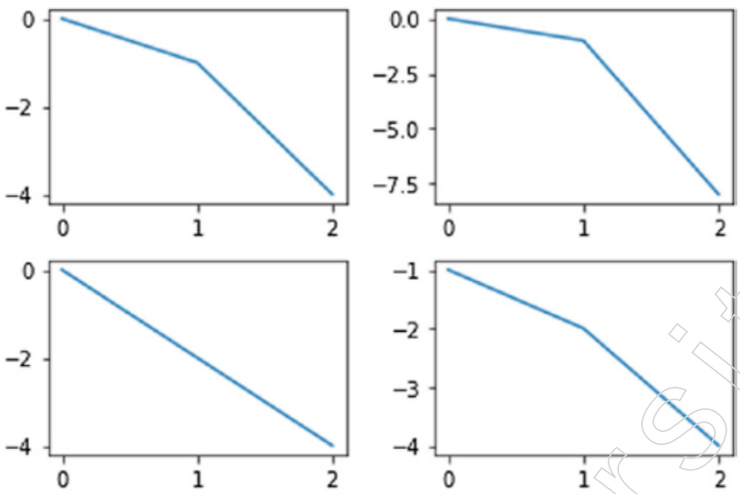
Multiple graphs can be displayed in the same output. Subplotting is the term for this method. Subplots can have unique titles, labels, and other characteristics. A grid is used to construct subplots. The first subplot is located in the upper left corner. The placements of the remaining subplots are relative to the first. As an example, consider the following:

```
x = np.arange(3)
plt.subplots_adjust(wspace=0.3,
hspace=0.3)
plt.subplot(2, 2, 1)
plt.plot(x, -x**2)
plt.subplot(2, 2, 2)
plt.plot(x, -x**3)
plt.subplot(2, 2, 3)
```

Notes

```
plt.plot(x, -2*x)
plt.subplot(2, 2, 4)
plt.plot(x, -2**x)
plt.show()
```

Output



The following is an object-oriented version of the same code:

```
fig, axs = plt.subplots(2, 2)
plt.subplots_adjust(wspace=0.3,
hspace=0.3)
axs[0, 0].plot(x, -x**2)
axs[0, 1].plot(x, -x**3)
axs[1, 0].plot(x, -2*x)
axs[1, 1].plot(x, -2**x)
plt.show()
```

4.1.10 File Formats in Data Analysis

The file format describes how data is stored on a computer hard drive or other storage device. Data types and data structure determine where and how records are stored. Your computer is filled with the most common data source formats, Excel spreadsheets, but you may not have directly used the world's largest data storage format: databases. Data files such as spreadsheets are extremely popular due to the flexibility they offer in data storage. The same flexibility of input makes it much more difficult to use data found in spreadsheets as a source for further analysis.

In most spreadsheets, there are two main forms of data: columnar data and pivot tables. The columnar form of the data is stored in structured columns, just like a query to a database where the data is stored. For many people, analysing data in spreadsheets is a data analysis technique they were taught in school or early in their careers. This led to a lot of data being stored in spreadsheets, both as columns and pivot tables.

There are many file formats in which useful data sources can be stored. While you have used spreadsheets and CSV files throughout your career, spatial files may be new

Notes

to you, but when used, they can add a new dimension to your understanding of your organisation. Other file types, such as PDFs, can contain useful data sets, but extracting data from them for the tool being analysed is not always consistent and straightforward. Since there are too many file types that data resides in, let us focus on the most common or useful ones.

Spreadsheets

The most common data file types used in your organisation are spreadsheets, typically Excel, Google Sheets, or equivalent data. These files can contain manually entered data, pivot tables, or data extracted from databases, to name a few sources. Spreadsheets were the first place you tried to prep data and start analysing. Spreadsheets are resources that cannot be ignored when working with data. With the flexibility of data entry and ease of editing, spreadsheets are a common place for your colleagues to model predictions or use their knowledge of a topic to derive meaning from the data.

However, the ability to insert data values, perform calculations, or delete data points means that caution should be exercised when using a spreadsheet as a data source. It is extremely easy to make mistakes such as B. wrong values or wrong calculations to wrong data values in worksheet cells. While many data professionals argue that spreadsheets are not a mission-critical data storage platform, they still play a key role in storing large datasets for most organisations, as noted in The Shape of some data. The flexibility that spreadsheets offer allows them to serve the needs of many people. Cells can store any type of data; the data does not need to be structured and files can be updated simply by overwriting the values that already exist in the file.

The flexibility of spreadsheets allows you to create many data-driven products from them.

Most business intelligence (BI) tools should store data in clear columns with a single header. When shared across an organisation, spreadsheets are often formatted to include more descriptive titles and other comments above or alongside data tables that you can use when analysing your data. This means spreadsheets need to be reformatted by removing headers and extra text before analysis with BI tools.

Delimited format (CSV/ TSV)

Delimited formats use a specific character to separate data values. Typically, these separators are commas (comma-separated values, or CSV for short), tabs (tab separated values, or TSV for short), spaces, or colons. These formats are suitable for storing data with a tabular structure. Each line of the file represents a record separated by newlines (\n or \r\n). Within a line, the record information is separated by a comma (,) for CSV, or a tab (\t) for TSV and so on. The first line of these files often contains the names of the table columns/properties.

CSV files can be opened in many types of programs used for spreadsheets. The main difference is in the structure of data storage. Each row in the CSV file must contain a single record, but each data field is separated by a comma instead of columns of data.

People sometimes confuse CSV and TSV files with spreadsheets. This is partly because most spreadsheet programs (e.g., Microsoft Excel) automatically display the CSV file as a table in your workbook. In the background, Excel checks the format and encoding of the file and converts it internally. However, Excel files have a different format than CSV and TSV files.

Notes

Fixed-width Format

Fixed-width format (FWF) does not use delimiters to separate data values. Instead, the values for a particular field appear in exactly the same place on each line. In general, using a filename extension like .csv, .tsv and .txt to specify the format of the file content. Filenames ending in .csv must contain comma-separated values, for example .tsv with tab separated values and .txt is usually plain text with no particular format. However, these extension names are just suggestions. Even if the file has a .csv extension, the actual content may not be formatted correctly. It is recommended to check the contents of the file to ensure the file format is correct.

Hierarchical Formats (XML/ JSON/ HTML)

Hierarchical formats store data in a nested form. For example, JavaScript Object Notation (JSON), commonly used for communication between web servers, contains key-value pairs and arrays that can be nested like a Python dictionary. Extensible Markup Language (XML) and HyperText Markup Language (HTML) are other popular document storage formats on the web.

Loosely Formatted Text

Network logs, instrument readings and program logs typically contain plain text data. There are templates that depend on the source tool but are not in a simple comma-delimited format. That is what we mean by weak formatting.

Another example is a single entry from the wireless device registry. The device reports timestamp, device ID, location and received signal strength from other devices.

Databases

Many factors that hinder the use of spreadsheets can be corrected using databases. Databases are software solutions for collecting, processing, storing, and producing data. Databases are uploaded to data servers specifically designed for the tasks they perform. Servers are still computers, but they have much more memory and processing power than a laptop or desktop computer. Databases are therefore designed for specific purposes that are important to you when working on data projects and considering what storage options to choose.

Traditionally, many databases use a programming language called Structured Query Language (SQL) to allow you to work with the data stored in them. SQL is a general-purpose language that allows you to create, update, query and delete database tables. SQL is well known to most analysts, but often not taught to non-data experts. This means that accessing data stored in databases can be difficult for many people.-

Spatial

If you are working with data that includes details such as location, distance, or area occupied by businesses or people, you are using geofiles, which contain information for mapping data points when performing geospatial analysis. Geospatial files may not be the same across all industries, but when you come across them, there are a few key differences to consider. Geospatial files contain data about points (specific locations), lines (paths), or polygons (areas on maps). These files are not supplied as a single file, but as a directory or folder of files. You will need specialised geospatial software or business intelligence tools to open up the spatial formats, although more business intelligence tools make it easier to perform this type of analysis alongside more traditional charts.

Notes

Summary

- The data analysis process is a systematic approach to examine, clean, transform and interpret data to extract valuable insights and make informed decisions.
- It is a critical step in various fields, including business, science, finance, and social sciences. The process involves several key steps, each contributing to a comprehensive understanding of the data and its underlying patterns.
- Data Collection: The first step is to gather relevant data from various sources. This may involve surveys, sensors, databases, web scraping, or other data collection methods.
- Data Cleaning: Raw data is often messy and may contain errors, missing values, or inconsistencies. Data cleaning involves identifying and rectifying these issues to ensure data accuracy and reliability.
- Data Exploration: In this stage, analysts use descriptive statistics and visualizations to get an initial understanding of the data's distribution, central tendencies, and relationships between variables.
- Data Preprocessing: To prepare the data for analysis, preprocessing techniques like normalization, scaling and feature engineering may be applied.
- Data Analysis: During this step, various statistical methods, machine learning algorithms and data mining techniques are used to extract insights, identify patterns, and draw conclusions from the data.
- Data Visualization: Visual representations, such as charts, graphs, and plots, are created to communicate the findings effectively and aid in understanding complex relationships in the data.
- Interpretation and Inference: The insights gained from data analysis are interpreted to draw meaningful conclusions and make data-driven decisions.
- Validation and Testing: It is essential to validate the analysis and test the results to ensure they are accurate and reliable.
- Reporting and Presentation: The last step involves presenting the findings in a clear and concise manner, often in the form of reports, dashboards, or presentations.
- The data analysis process is iterative and may require revisiting earlier steps to refine the analysis or explore new questions.
- It involves a combination of domain knowledge, statistical expertise, and programming skills to uncover valuable insights and support evidence-based decision-making.

Glossary

- Data Analysis: The process of examining, cleaning, transforming and interpreting data to extract valuable insights and make informed decisions.
- Data Collection: The initial step of gathering relevant data from various sources, such as databases, surveys, sensors, or web scraping.
- Data Cleaning: The process of identifying and correcting errors, inconsistencies, and missing values in the raw data to ensure accuracy and reliability.
- Data Exploration: The initial examination of the data to gain an understanding of its distribution, central tendencies, and relationships between variables.
- Descriptive Statistics: Summary statistics that provide a concise representation of data, such as mean, median, mode, standard deviation, and range.

Notes

- Data Preprocessing: The preparation of data for analysis, including normalization, scaling, and feature engineering.
- Statistical Methods: Techniques used to analyse data and draw conclusions, including hypothesis testing, regression analysis and ANOVA.
- Machine Learning Algorithms: Algorithms that allow computers to learn from data and make predictions or decisions without being explicitly programmed.
- Data Mining: The process of discovering patterns, relationships, or anomalies in large datasets using automated methods.
- Data Visualization: The use of visual representations, such as charts, graphs, and plots, to communicate data insights effectively.
- Interpretation and Inference: The process of drawing meaningful conclusions and making inferences from the data analysis results.
- Validation and Testing: The process of verifying the accuracy and reliability of the data analysis and results.
- Iterative Process: A repetitive approach to data analysis, where earlier steps may be revisited and refined to improve the analysis or explore new questions.
- Domain Knowledge: Understanding of the specific field or subject matter to interpret data in a meaningful context.
- Data-driven Decision-making: Making informed decisions based on data analysis and evidence rather than intuition or gut feelings.
- Data Report: A comprehensive document presenting the data analysis process, findings, and recommendations.
- Dashboards: Interactive visual displays of data insights that provide real-time monitoring and tracking of key performance indicators.
- Exploratory Data Analysis (EDA): An initial examination of the data to identify patterns, trends, and outliers.
- Data Manipulation: The process of filtering, aggregating, and transforming data to gain specific insights or prepare it for analysis.
- Data-Driven Insights: Actionable and valuable information extracted from data analysis to support decision-making and problem-solving.

Check Your Understanding

1. What is the primary goal of data analysis?
   a) Data visualization
   b) Data preprocessing
   c) Extracting valuable insights from data
   d) Data collection
2. What is the first step in the data analysis process?
   a) Data cleaning
   b) Data visualization
   c) Data collection
   d) Data preprocessing
3. What does data cleaning involve?
   a) Extracting valuable insights from data
   b) Transforming data for analysis
   c) Identifying and correcting errors in data
   d) Presenting data in visualizations

Notes

4. What is the purpose of data preprocessing in data analysis?
   a) Collecting data from various sources b) Transforming data for visualization
   c) Preparing data for analysis d) Extracting insights from data
5. What is exploratory data analysis (EDA) in the data analysis process?
   a) Extracting insights from data b) Cleaning and preprocessing data
   c) Summarising data using descriptive statistics
   d) Initial examination of data to understand its characteristics
6. Which step involves creating visual representations like charts and graphs to communicate data insights effectively?
   a) Data cleaning b) Data preprocessing
   c) Data visualization d) Data analysis
7. Which technique is used to analyse data and draw conclusions based on probability and hypothesis testing?
   a) Machine learning b) Data preprocessing
   c) Exploratory data analysis d) Statistical analysis
8. What is the process of verifying the accuracy and reliability of data analysis results?
   a) Data preprocessing b) Data validation
   c) Data visualization d) Data collection
9. What does domain knowledge refer to in the data analysis process?
   a) Expertise in data visualization techniques
   b) Understanding of specific subject matter or field
   c) Knowledge of machine learning algorithms
   d) Programming skills in Python
10. What does data-driven decision-making mean?
   a) Making decisions based on intuition and gut feelings
   b) Making decisions based on expert opinions
   c) Making decisions based on data analysis and evidence
   d) Making decisions based on random chance
11. What is the iterative process in data analysis?
   a) A repetitive approach to data visualization
   b) A continuous cycle of collecting data from various sources
   c) A repetitive approach to data preprocessing
   d) A cycle of revisiting and refining earlier steps in the analysis
12. What is the primary purpose of data mining in data analysis?
   a) Extracting valuable insights from data
   b) Preparing data for visualization
   c) Discovering patterns, relationships, or anomalies in large datasets
   d) Validating data analysis results

Notes

13. Which library is commonly used for data visualization in Python?
    a) NumPy b) pandas
    c) Matplotlib d) Scikit-Learn
14. What is the process of presenting the data analysis results in a clear and concise manner?
    a) Data visualization b) Data preprocessing
    c) Data reporting d) Data interpretation
15. What is the main purpose of exploratory data analysis (EDA)?
    a) To perform hypothesis testing b) To preprocess and clean data
    c) To understand the distribution and characteristics of data
    d) To create visualizations
16. Which step in the data analysis process involves transforming data for analysis and modelling?
    a) Data preprocessing b) Data collection
    c) Data cleaning d) Data visualization
17. What does “EDA” stand for in data analysis?
    a) Exploratory Data Aggregation b) External Data Analysis
    c) Exploratory Data Analysis d) Extract Data Aggregation
18. What is the purpose of statistical methods in data analysis?
    a) To collect data from various sources b) To transform data for analysis
    c) To identify and correct errors in data d) To analyse data and draw conclusions
19. Which term refers to the use of visual representations, such as charts and graphs, to communicate data insights?
    a) Data mining b) Data preprocessing
    c) Data visualization d) Data interpretation
20. What is the process of preparing data for analysis by normalising, scaling, and handling missing values called?
    a) Data cleaning b) Data visualization
    c) Data preprocessing d) Data interpretation

Exercise

1. Explain Overview of Data Analysis and Its Process.
2. What is Data Analysis?
3. Explain Important Terms in Data Analysis.
4. What is Conceptualising Data Analysis as a Process?
5. Describe Managing the Data Analysis Process.
6. Explain the Procedures for Data Analysis: Quantitative.
7. What is Procedures for Data Analysis: Qualitative?
8. Explain Sources for Data Gathering: Machine and File Data Sources.

Notes

9. What is Data Analysis Using Matplotlib: Part – I?
10. Explain Data Analysis Using Matplotlib: Part – II.

Learning Activities

1. Discuss Data Analysis and Its Process with an example.
2. Describe Data Analysis Process of any company.

Check Your Understanding- Answers

1. c	2. c
3. c	4. c
5. d	6. c
7. d	8. b
9. b	10. c
11. d	12. c
13. c	14. c
15. c	16. a
17. c	18. d
19. c	20. c

Further Readings and Bibliography

1. "Introduction to Machine Learning" by Ethem Alpaydin
2. "Pattern Recognition and Machine Learning" by Christopher Bishop
3. "Machine Learning: A Probabilistic Perspective" by Kevin P. Murphy
4. "Machine Learning Yearning" by Andrew Ng
5. "Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow" by Aurélien Géron
6. "An Introduction to Statistical Learning" by Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani
7. "Deep Learning" by Ian Goodfellow, Yoshua Bengio and Aaron Courville
8. "Machine Learning" by Tom Mitchell, McGraw-Hill Education, 1st Edition.
9. "Artificial Intelligence: A Modern Approach" by Stuart Russell and Peter Norvig, Pearson, 3rd Edition
10. "Deep Learning" by Ian Goodfellow, Yoshua Bengio and Aaron Courville, Publication: The MIT Press, 1st Edition
11. "Natural Language Processing with Python" by Steven Bird, Ewan Klein and Edward Loper, O'Reilly Media, 1st Edition

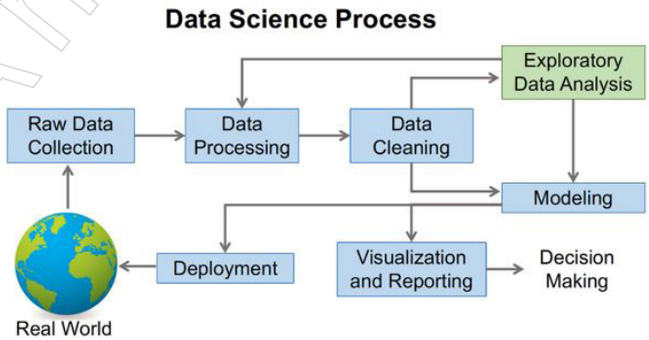
Module - V: Exploratory Data Analysis

Notes

Learning Objectives

At the end of this module, you will be able to:

- Learn what is data analytics
- Know modern data ecosystem
- Describe key players in the data ecosystem
- Learn defining data analysis
- Summarise viewpoints: what is data analytics
- Analyse the data analyst role
- Learn responsibilities of a data analyst
- Know a day in the life of a data analyst
- Describe the data ecosystem and languages for data professionals
- Learn the data analyst ecosystem
- Know types of data
- Understanding different types of file formats
- Learn sources of data
- Know languages for data professionals
- Describe understanding data repositories and big data platforms
- Learn about data repositories
- Know data marts, data lakes, ETL and data pipelines
- Learn foundations of big data
- Describe big data processing tools
- Summarise gathering data
- Analyse identifying data for analysis
- Learn data sources
- Know how to gather and import data
- Describe data wrangling
- Learn what is data wrangling and tools for data wrangling
- Know data cleaning
- Learn communicating data analysis findings and incomplete section



Notes

Introduction

The cleaning, transformation, and modelling of data to identify useful information for business decisions is called data analysis. The purpose of data analysis is to derive actionable insights from data and make decisions based on that knowledge. An important example of data analysis is that when we make a decision in our daily life, we ask ourselves what happened the last time we made that decision, or what would have happened if we had made it. It is nothing but looking back in time or looking into the future and making decisions based on our insights. We do this by collecting memories of the past or fantasising about the future. So much for data analysis. Data analysis is what an analyst does today for business purposes.

5.1 What is Data Analytics?

The science of examining raw data to draw conclusions is called data analysis. Many data analysis approaches and processes have been transformed into mechanical processes and algorithms that process raw data and are intended for human consumption.

Data analysis is an umbrella term that refers to various data analysis techniques. Data science techniques can be applied to any type of data to generate insights that can be used for improvement. Data analysis techniques can be used to uncover trends and metrics that might otherwise get lost in the sea of data. This data can then be used to improve the overall performance of an organisation or system by optimising processes.

For example, manufacturing companies often log uptime, downtime and work queues for various machines and then analyse the data to better plan workloads and keep machines at peak performance. Data analysis can do more than just identify production bottlenecks. Data analytics are used by gambling companies to create player incentive programs that keep most of them in the game. Much of the same data analysis is used by gambling companies. Content to encourage you to click, browse, or rearrange your content to achieve a different look or click.

Data analysis is essential as it helps companies improve their performance. Companies can help reduce costs by developing more efficient business processes and storing large amounts of data and integrating it into their business strategy. Data analytics can also help a company make better business decisions and evaluate customer patterns and satisfaction, which can lead to the development of new and better products and services.

5.1.1 Modern Data Ecosystem

Organisations want a unified view across the enterprise that enables them to govern and govern data consistently. This perspective brings together technical, business, and operational metadata from across the data supply chain, from origins to transformations and refinements, staged and derived data, models and rollups, reports, and data applications.

Today, tracking data to provide a holistic view is mostly a manual process. Industry experts are interviewed and the results are recorded in systems and spreadsheets that may be updated. And it can only provide a snapshot of static metadata at a time. These systems are rarely associated with data storage and pipeline processes. The new developments are specialised in this area: Some look at the data lake, others at the origin of the pipeline, still others look at the content through semantic models and profiling.

Notes

However, each tool is limited to its specific functionality and operating environment. There is a gap between where business rules are developed and written and how they are actually enforced.

Our society is moving from the industrial age to the digital age. As society goes digital, the role of data is becoming increasingly important. Sensors are used to collect data, then store, process, analyse and visualise it before being used by entities (human and/or digital) to obtain information and/or make informed decisions.

Today's "data-intensive" applications include business analytics, statistics-driven AI, digital twins and more. For example, data is needed to train statistics-driven AI and create digital twins and to enable companies to evaluate their performance in real time and learn how to improve their operations. To highlight the potential value of data, the industry uses terms like "thrive on data". In the meantime, we have all become aware of the opportunities, but also possible advantages and disadvantages, of collecting and using data on a large scale, for example from Google, Facebook, and other companies.

"Data-intensive" applications require a diverse set of data resources to "power" them. For example, raw observations from various sensors/informants, processed and/or enhanced artifacts in terms of, for example, predictive models, intent representations (e.g., designs, plans, projects, etc.), specifications (source code, work practices, etc.), or standards (regulations, rules, guidelines, etc.).In addition, such applications require data from many different sources, resulting in a transfer of ownership of the data, or at least the right to use that data.

As data has become an increasingly important resource base, systems for collecting, storing, processing, analysing and visualising data have become complex systems involving multiple actors with different interests. Therefore, we believe these complex systems should be referred to as "data ecosystems," which we define as "the entire complex network of social, physical and digital actors (i.e., Actor Web) that provide, own, sell, buy, trade and manipulate." , store and use the data."

These data ecosystems must address technological issues such as reliability, performance, interoperability, semantics, etc. as well as social issues such as data value, privacy, trust, ownership, ethics, risk, etc. For example, they can obviously play a role in how far the data can be related (human behaviour).

Also, because the data corresponds to "something" in a social, economic, or physical setting, the quality of this correlation needs to be considered. At the same time, some actors may have an interest in manipulating the data and thus falsifying this correspondence. The data also raises the question of ownership. Some companies may view the data as strategic, leading them to try to control or sell access to others. Regarding the personal data market, for example, an interesting point of view on this subject is presented.

A data ecosystem can also be viewed as a "data management company" or network company whose core business is "data management", where "data management" refers to all data activities (collection, exchange, handling, storage, use, etc.).). Such a "data management organisation" will almost always be part of a larger company focused on "normal" products and services. The use of enterprise modelling methods can certainly contribute to the development of data ecosystems such as "data management companies". Thus, the above issues are directly applicable, but imply the need for deeper documentation of data ownership, data lineage, data value (for specific stakeholders), access controls, data rules, etc.

Notes

5.1.2 Key Players in the Data Ecosystem

The companies that are paving the way for the future are those that use data to find opportunities and use that knowledge to differentiate themselves. Business leaders have realised that data is key to gaining a competitive edge, whether it's studying financial transaction patterns to detect fraud, using recommendation engines to increase conversions, exploring social media posts to gather customer feedback, or customising their offerings based on the analysis of customer behaviour.

To extract value from data, you need a diverse mix of skills and people with different roles. This video discusses the importance of data engineers, data analysts, data scientists, business analysts and business analysts or BI analysts in helping organisations leverage massive amounts of data and turn it into actionable insights.

Data Engineer: It all starts with the data engineer. Data engineers create and manage data architectures and make data available for business operations and analysis. Within the data ecosystem, data engineers collect, integrate, and organise data from various sources.

Data repositories are used for cleaning, converting, and preparing data as well as for storing and managing it. They have made data available in formats and systems that can be consumed by various business applications and by stakeholders such as data analysts and data scientists. A data engineer should have strong programming skills, a thorough understanding of technology systems and architectures and a thorough understanding of relational and nonrelational databases and data storage.

Data engineers are tasked with spotting patterns in big data and designing algorithms to make raw data more useful to businesses. This IT position requires a variety of technical skills, including in-depth knowledge of SQL database design and various programming languages. However, data engineers need to be able to communicate across departments to understand what business leaders are trying to accomplish with the organisation's massive data sets.

Data engineers are often tasked with developing algorithms to make raw data more accessible. But to do that, they must first understand the business or customer goals. When working with data, alignment with business goals is crucial, especially for companies working with large and complex datasets and databases.

Data Analyst: In a nutshell, a data analyst examines and cleanses data to uncover insights, uncover connections, spot patterns, and apply statistical approaches to it so businesses can make decisions. Analyse and explore data and visualise data to interpret and communicate data insights.

Analysts answer questions like, "In general, is the user experience when searching our site good or bad?" o What is the public opinion about our rebranding initiatives? Is there a connection between selling one product and selling another? Data analysts should be comfortable with spreadsheets, writing queries and using statistical tools to create charts and dashboards. Data scientists in today's world also need to know how to code. You must also have excellent analytical and storytelling skills. Today, let's take a look at the roles of data scientists in this ecosystem.

Data Scientist: Data scientists analyse data to generate meaningful insights and build predictive models using machine learning or deep learning models trained on historical data. Data analysts answer questions like, "How many new social media followers can one expect to get in the next month?" or "How many of my customers will one lose to the competition in the next quarter? "or "Is this financial transaction unusual

Notes

for this client?" Math, statistics and basic knowledge of programming languages, databases and data modelling are skills required of data scientists. You must also have domain knowledge.

Data Scientists are the new generation of data scientists with the technical skills to solve complex problems and the curiosity to know what problems need solving. They are both mathematicians, computer scientists and trend watchers. In addition, they are in high demand and well paid as they work in both business and IT. Who wouldn't want to be part of this elite group?

I, too, am a reflection of the present. Data scientists weren't on many people's radars a decade ago, but their rise shows how companies are using big data today. This unmanageable tangle of unstructured data can no longer be ignored. It's a virtual gold mine that can help you increase your income as long as someone searches for and uncovers business information that no one else has thought of. The data analyst comes into play.

Business Analysts or Business Intelligence Analysts: Business analysts use the work of data analysts and data analysts to study potential business implications and the actions they should take or recommend. The only difference is that BI Analysts do the same. Her focus is on the market forces and external factors that define her industry. They provide intelligent business solutions by organising and monitoring data related to various business activities and interpreting that data to generate insights and actions that help organisations improve their performance.

5.1.3 Defining Data Analysis

The process of collecting, cleaning, analysing, and extracting data and interpreting and reporting the results is called data analysis. Through data analysis, we discover patterns in data and correlations between different data points. Ideas and conclusions are drawn from these patterns and relationships. Data analytics help companies understand their historical performance and make future decisions. Companies can validate the action plan by analysing the data before committing. Saving time and resources while delivering better performance is a win-win situation. We will look at four different types of data analysis, each with its own purpose and place in the data analysis process. By summarising historical data and presenting the findings to stakeholders, descriptive analysis can help answer questions about what happened over time. It contributes to the understanding of past events. For example, a cash flow analysis or a historical performance analysis based on the organisation's KPIs. This question can be answered with the help of a diagnostic analysis. What caused this? Use descriptive analysis information to further investigate the root cause of the result.

For example, a sudden increase in website traffic for no apparent reason, or an increase in sales in an area where marketing has not changed. Predictive analytics can help you understand what's coming next. Future performance is predicted based on historical data and trends. Businesses use predictive analytics in a variety of areas, including risk assessment and revenue forecasting. Importantly, the goal of predictive analytics is to predict what might happen in the future, not predict what will happen in the future. All predictions are based on probability. What should be done about it? Prescriptive analytics can help answer this question.

The probability of different scenarios can be calculated by evaluating decisions and past events. This is the number by which the action plan is selected. Self-driving cars are an example of prescriptive analytics. They assess their surroundings before making

Notes

decisions about speed, lane changes and route, among other things. Alternatively, airlines can automatically adjust fares based on customer demand. Gasoline prices, weather conditions and traffic on connected roads are factors to consider. Let's look at some of the key steps in any data analysis procedure.

Understand the problem and the desired outcome: Understanding the problem to be solved and the desired outcome is the first step in data analysis. Before you begin the analysis, you must first determine where you are and where you want to go.

Define an easy-to-understand metric: Choosing what to measure is part of this step in the process. For example, how much of product X is sold in a given location and how this is measured. Once you know what you're measuring and how you're going to measure it, determine what data you'll need, what data sources you'll collect it from and what tools you'll work with during the quarter or festival.

Data Cleansing : Once the data has been collected, the next step is to fix any data quality issues that might affect the accuracy of the analysis. This is a crucial step because only clean data can guarantee the validity of the study. You look for missing or incomplete values as well as outliers in your data. For example, a set of customer demographics with a value of 150 in the age field is an outlier. You also normalise data from different sources. Data exploration and analysis Once the data is cleaned, it can be extracted and analysed from different angles.

You may need to manipulate your data in various ways to understand trends, establish correlations and find patterns and variations.

Interpretation of Results: After analysing the data, it is time to interpret the results. In addition, a complementary study can be performed, which can be an iterative cycle. When interpreting your results, ask yourself whether your analysis stands up to challenge and whether there are any caveats or conditions under which it might not be true.

Presentation of Results The ultimate goal of any analysis is to influence the decision-making process. Just as important as the data analysis process itself is the ability to explain and present the results clearly and effectively. You can present your data in a variety of ways, including reports, dashboards, graphs, charts, maps, and case studies.

Descriptive Analysis

Descriptive analysis is the first type of data analysis. This is the basis of any data analysis. It is the most basic and widespread use of data in business today. Descriptive analyses answer the question "What happened?" by presenting historical data in the form of a dashboard.

The most common use of descriptive analytics in business is KPI tracking (KPI).KPIs describe a company's performance using a set of metrics. Here are some examples of business applications of descriptive analysis:

- KPI dashboards
- Monthly income reports
- Overview of prospects

Diagnostic Analysis

After answering "What happened," the next step is to delve in and ask, "Why did this happen?" This is where diagnostic testing comes in. Diagnostic Analysis addresses descriptive analysis to uncover the root causes of performance. This type of analysis

Notes

is used by businesses because it creates multiple connections between data and recognises patterns of activity.

The generation of accurate information is an important part of the diagnostic analysis. If new issues arise, chances are you've already read up on the situation. Because you already have the data, you don't have to redo the work and all issues are linked together.

Diagnostic Analysis offers many professional applications including:

- Shipping company is investigating the cause of slow deliveries in a specific region.
- SaaS startup investigates whether marketing tactics led to more evidence.

Predictive Analytics

Predictive Analytics aims to answer the question "What could happen?". This type of analysis helps predict future outcomes based on past data.

This form of analysis is an advance over descriptive and diagnostic tests. Predictive Analytics uses the information we collect to make meaningful predictions about what will happen next. These studies are based on statistical models that require the use of additional technology and work to make predictions. It's also important to remember that the prediction is just a guess; The accuracy of predictions depends on accurate and high-quality data.

While descriptive and diagnostic analytics are ubiquitous in the business world, many organisations are showing early signs of trouble with predictive analytics. Some companies don't have the manpower to integrate predictive analytics everywhere. Others are unwilling or unable to invest in cross departmental analytics teams or build existing teams.

Predictive Analytics has many business applications including:

- Risk assessment
- Sales forecast
- Use customer segmentation to determine which prospects are most likely to convert
- Predictive analytics in customer success teams

Prescriptive Analytics

The latter type of data analysis is the most desirable, but few companies are actually willing to do it. Prescriptive analytics is a state-of-the-art data analysis method that uses the insights from all previous research to decide the best course of action in a given situation or choice.

Prescriptive analytics uses the latest data management technologies and techniques. This represents a significant organisational effort and organisations need to ensure they are willing to commit the time and resources required.

An example of prescriptive analytics is artificial intelligence (AI). AI systems consume vast amounts of data to constantly learn and make intelligent judgments. Well-designed AI systems are able to communicate and even respond to these decisions.

AI enables daily implementation and optimisation of business processes without human intervention.

Notes

Prescriptive analytics and AI are now being used by most major data-driven organisations (Apple, Facebook, Netflix, and others) to improve decision-making. The transition to predictive and prescriptive analytics can be challenging for some organisations. More businesses will move into the data space as technology improves and more professionals become data savvy.

5.1.4 Viewpoints: What is Data Analytics?

Data analysis is an umbrella term that refers to various data analysis techniques. Data science techniques can be applied to any type of data to generate insights that can be used for improvement. Data analysis techniques can be used to uncover trends and metrics that might otherwise get lost in the sea of data. This data can then be used to improve the overall performance of an organisation or system by optimising processes. For example, manufacturing companies often log uptime, downtime and work queues for various machines and then analyse the data to better plan workloads and keep machines at peak performance. data analysis can do much more than just identify production bottlenecks. Data analytics are used by gambling companies to create player incentive programs that keep most of them in the game. Much of the same data analysis is used by gambling companies. Content to encourage you to click, browse, or rearrange your content to achieve a different look or click.

Data analysis is essential as it helps companies improve their performance. Companies can help reduce costs by developing more efficient business processes and storing large amounts of data and integrating it into their business strategy. Data analytics can also help a company make better business decisions and evaluate customer patterns and satisfaction, which can lead to the development of new and better products and services. Different data scientists define data analysis differently.

Data analysis consists of collecting data and then analysing them to confirm various hypotheses. Data analytics also means telling the story of your data. Use data to communicate situations in the world clearly and directly to those around you. Data analysis is the process of making judgments based on available information. You watch the news every morning, like every morning. The weather report informs you about the daily temperature and whether it will rain. It may affect what you wear and what activities you can participate in. Data analytics is not an abstract concept; It's something we do every day.

However, it now has a technical name and people are being paid to do it on an ever-increasing scale. But it's really not that difficult. In other words, you have a problem and you need to fact-check your hypothesis. This is where data analysis comes into play. The process begins by defining the problem. You then need to develop your hypothesis. To do this, information must be collected, cleaned, evaluated, and then communicated to the most important stakeholders.

Any data set that you can use to look up information, anything that helps you understand what's going on, is considered data analysis. Always be aware of the financial situation of a CPA. Always look at the data to see where people have been, where they are now and where they are going. This information allows one to see further into the future and predict nearly the future of every company in which they have had an interest. Data analysis is the process of collecting, cleaning, analysing, presenting and finally sharing data and analytics to better represent what is happening in your business and your data so you can make better decisions.

Notes

Data analytics is the process, or rather the phenomenon, of taking data from a relevant demographic, such as your customers or your social media audience, breaking it down into subsets and using that data to make decisions about the products you want or to offer services or, in the digital environment in which we find ourselves, to make decisions about what specific content you want to publish in order to make it attractive to your target audience.

5.2 The Data Analyst Role

A data analyst collects and organises information about sales performance, market research, logistics, linguistics, and other behaviours. They bring technical expertise to ensure the high quality and accuracy of the data, then analyse it, create it, and present it in a way that helps people, businesses and organisations make better decisions.

A data analyst analyses datasets to find solutions to a company's customer problems. This information is also provided by the data analyst to management and other interested parties. These people work in a variety of industries, including business, banking, criminal justice, science, medicine, and government.

A data analyst is someone who has the knowledge and skills to transform raw data into information and insights that can be used to make business decisions.

5.2.1 Responsibilities of a Data Analyst

While the role of a data scientist varies depending on the type of organisation and the amount of data-driven practices implemented, some responsibilities are common in today's organisations. For example, retrieve data from primary and secondary data sources. Develop queries to extract data from databases and other data collection systems. In preparation for data analysis, data is filtered, cleaned, normalised and reorganised. Statistical tools are used to interpret the data sets. Using statistical techniques to identify patterns and relationships in data. Identifying patterns in large data sets and analysing trends is a difficult task.

Create reports and charts that clearly communicate trends and patterns. Create appropriate documentation to identify and demonstrate the processes of the data analysis process. Let's look at some of the talents a data scientist needs for these roles. The data analysis process requires a combination of technical, functional and non-technical skills.

Let's start with some of the technical skills you'll need as a data scientist. Here are some:

- Using spreadsheets like Microsoft Excel or Google Sheets will be an added benefit. Expertise in statistical visualisation and analysis applications such as IBM Cognos, IBM SPSS, Oracle Visual Analyser, Microsoft Power BI, SAS and Tableau.
- At least one programming language is required, e.g. B. R, Python and under certain circumstances C++, Java and MATLAB.
- Knowledge of SQL and the ability to work with data in relational and NoSQL databases are required.
- Data centers, data warehouses, data lakes and data pipelines are examples of data repositories where data can be accessed and extracted. Hadoop, Hive and Spark are examples of big data processing tools.

Notes

We'll now take a look at some of the technical skills you'll need as a data scientist. Here are some:

- Knowledge of statistics will help you analyse data, validate research and spot logical fallacies and errors.
- Analytical skills to aid in research, data interpretation, theorising, and forecasting. Problem solving skills because the goal of any data analysis is to solve problems definitively. Because the data analysis process actually starts with a clear formulation of the problem and the desired outcome, exploratory skills are crucial to the discovery phase – that is, understanding the problem from the perspective of different stakeholders and users.
- Data visualisation capabilities to help you identify strategies and tools to effectively communicate your findings based on your audience, data type, context and research objective.
- Project management skills are required to manage the initiative process, people, interdependencies and schedules. This brings us to your overall skills as a data analyst.

Data analysis is both a science and an art. While you may have excellent technical and functional knowledge, soft skills are one of the most important factors in your success.

- This includes your ability to collaborate with business and cross-functional teams, communicate effectively to report and present results, create an engaging and compelling story and gain support and recognition for your work. The main focus of data analysis is curiosity.
- As you work, you will encounter patterns, phenomena and anomalies that may lead you down a different path. An intelligent analyst is able to allow new questions to develop and to challenge one's own beliefs and theories.
- You'll also hear data scientists talk about intuition as a critical skill. Note that intuition here refers to the ability to predict the future based on past experience and pattern recognition.

5.2.2 Viewpoints: Qualities and Skills to be a Data Analyst

In this topic, we discuss the qualities and skills required to work as a data scientist. Traits and Skills of a Naturally Interested Data Analyst. Someone who works diligently and enjoys working with computers. A curious person will seek answers even when there are no questions and is not afraid to explore and seek in areas that have not previously been considered. Is it more important to pay attention to details or to look for reasons? You walk into a room and you just count the people, how is the room set up, pay attention to the little details and then you like computers because the technology is moving so fast? Something or a skill you learned today may not be applicable in two or three years. Therefore, you must be able to learn new software and acquire new skills as the market or industry evolves.

Is a combination of soft skills and soft skills: Python, SQL, R, Tableau and Power BI are examples of soft skills, while soft skills or interpersonal skills relate to whether you know what data you use, what tool you use and how to share data with key stakeholders. In addition, these skills required strong business acumen and presentation skills.

One must be very thorough, one must love numbers, one must love information and one must be willing to examine it thoroughly, not just superficially. So, in our industry we

Notes

can't just exchange a bank statement for cash; We must study and analyse it. Is the seal in good condition?

In today's world there are many scams, misunderstandings, and people who want to steal your information and use it dishonestly. Therefore, an experienced data analyst should be able to compare last year's data with this year's data to see if everything looks right. Do you need her eye and attitude not to take things at face value?

To be a data analyst, you need a variety of attributes and talents, which we divide into two categories: soft skills and technical skills.

Soft Skills: The most important soft skills for a data scientist are to be very curious, to ask lots of good questions, to be very considerate, to listen carefully and to understand both your and your peers' views and what they think is important most need, you are a data scientist you are enthusiastic and always want to learn because analytics is a rapidly evolving field and you need to be constantly learning and reading to stay current.

Technical Skills: Being a data analyst requires a wide range of technical skills.

SQL is the most important technical skill every new data scientist should learn. It is by far the most popular and you need to understand SQL if you are retrieving data from a database. And there's nothing like a data scientist who knows SQL inside and out.

Sometimes people try really advanced technologies before they master the basics of SQL.

Python and R, the two most commonly used programming languages for data analysis, are always beneficial. As a new data analyst, you don't need to be good at both, or even one or the other. However, if you start excelling in both fields, it will greatly benefit your career.

Another important technical skill for a data analyst is mastery of at least one data visualisation technology and knowledge of the general principles of data visualisation. The skills of data scientists today are far more dynamic than ever. Therefore, data scientists need to understand the problem they want to solve with data.

Using SQL, they were able to extract the required data and structure from the data lake. There will be many different tables and they will have to figure out how to put them together, then retrieve the data, clean it, manipulate it, manipulate it, and evaluate it to derive ecological insights. Present these results concisely and clearly with effective visualisations and dashboards, i.e., create a good story from your data.

5.2.3 A Day in the Life of a Data Analyst

A day data scientist can cover a wide range of tasks from collecting data from various sources, writing queries to retrieve data from data repositories, sifting through rows of data to generate insights, creating reports and dashboards, and interacting with stakeholders to gather information and present results. just to name a few. And of course, an important point: cleaning and preparing the data so that the results have a solid basis, which, by the way, is an important part of what any data analyst would do in their job.

But if one had to describe one type of day to you, it would be the day one scours the data for ideas. This is the aspect of the job that excites one the most. Let us take an illustration of a day in the life of a data scientist. A smart grid technology solutions company based in India, is hiring a data scientist. The company is a winner of the Beacon Awards for Smart Energy and Smart Cities solutions. Through their actionable

Notes

information platform, data analysts deliver integrated solutions for the operations centers of energy companies and smart cities.

Their client, an energy company in southern India, has seen an increase in complaints about overbilling. The fact that these fears are so widespread suggests there is more to them than just random events. So, the data analyst was asked to check the complaints and billing details to see if he could identify anything. The data analyst starts by assessing his current situation. Complaint data, subscriber data and billing data are some of the obvious places he will investigate. This allows him to compile a list of questions or make initial guesses before diving into the details of the data.

For example, consider the following usage pattern among customers who have reported this issue: Is there one usage area that is more likely to be overloaded than others? The complaints focus on the following areas: Are the complaints concentrated in certain parts of the city?

Frequency and frequency of complaints related to individual subscribers: Are the same subscribers constantly complaining about excessive bills? If so, how often does this happen repeatedly? Once a subscriber becomes overloaded, does it happen every month thereafter, or is it sporadic, if any, to recurring events? The data analyst identifies datasets which he isolates and analyses to confirm or rejects his hypotheses once he has clarified his initial hypotheses and questions. The data analyst takes the yearly, quarterly, and monthly averages of the complainant's billing amounts and looks for one area where the number of complaints is decreasing more than others.

The data analysts then checks the complainants' location information to see if there is a link between the surcharges and postcodes. Here he observes what at times appears to be a series of criticisms. It seemed like it had the potential to become something. Rather than move on to the third hypothesis, the data analyst decided to look at the data. He then get the date from the login data.

Over 95% of complaints concern their subscribers for more than seven years, although not everyone who has been with us for that long has been affected by this complaint. So far, they have seen grouping by region as well as broad grouping of complaints by date accessed. Then the data analyst gets the make and serial numbers from the meters.

All serial numbers come from the same batch of meters from the same manufacturer. The reserves resulted from the regions in which these branches were located and which were characterised by a high concentration of these branches. The data analyst is confident that he can now share these findings with interested parties. He will also disclose the sources of the data and the steps he took to reach this conclusion, as it always gives more credibility to the conclusions. This project can end at this point or come back. Maybe the same problems with small differences, or completely different complaints for which one needs to find solutions.

5.2.4 Viewpoints: Applications of Data Analytics

There are now data analysis applications available worldwide. Someone had to research and identify what information they wanted to reveal in each ad they viewed, both about the customer and the company. So, know four out of ten dentists, otherwise you will find calorie values or allergic reactions to certain foods, all of which need to be tested. It's not something to think about in isolation; It's something we do every day. People with diabetes who monitor their blood sugar are also analysed, hence its

Notes

common uses. The beauty of analytics these days is that you can use them in a variety of situations. Data and analytics can help every industry, industry, and function of a company. Generate reports in a predefined, standardised format, whether you're researching sales funnels or month-end financial data.

Everything in today's world is based on data. It doesn't matter if it's social media or big business. The term "data" refers to any type of information. Every organisation, every institution has to keep a set of data that it has recorded, collected, and stored over a long period of time. This data is collected, stored, and evaluated in order to improve and evaluate the development of the company. Data analysis or data analytics is a broad field that is most important to understand nowadays.

The term "data analysis" refers to the process of analysing data to reach specific conclusions needed to achieve business goals. It involves organising a large amount of unstructured data and using statistical tools to extract the necessary information from it. All of this requires the creation of charts, graphs, and other visual aids. Data analysis is used in virtually every aspect of human life, not just in manufacturing companies and industries.

Data Analysis in Finance We are seeing an increasing use of alternative data analysis in the financial world. For example, we can combine traditional financial research with sentiment analysis of tweets and breaking news to make better investment decisions. In addition to monitoring changes in industrial activity, satellite imagery data can be used. In addition, geolocation data can be used to track in-store traffic and predict sales.

Application of Analytics in Different Fields

1. Transportation: Data analytics can help improve transportation systems and the information that surrounds them. The predictive analytics strategy helps identify transport issues such as data traffic or network congestion. It helps to synchronise huge amounts of data and use it in the development and implementation of plans and strategies to create alternative routes, minimise congestion and thus reduce the number of accidents and random events. Data analytics can also be used to improve the shopper's travel experience by gleaning insights from social media. It also helps travel agencies customise their packages and enhance the personalised travel experience based on the information collected.
2. Logistics and delivery: Data analytics are used by several logistics companies such as DHL, FedEx, and others to manage their global operations. With the help of analysis apps, they can use GPS trackers to determine optimal shipping routes, estimate delivery times and track the status of shipped goods in real time. Data analysis has made online shopping more accessible and popular.
3. Internet search engine or Internet search results: When you search for something on the internet, search engines like Yahoo, Bing, Duckduckgo and Google use a collection of data to return results. When you use the search button, search engines use data analysis algorithms to give you the best search results in a short amount of time. Data analysis is used to collect various data that arise when searching for information.

   The requested data is treated as a keyword and all relevant information is presented in a logical and easy to understand order. For example, when you search for a product on Amazon, it keeps popping up on your social media profiles or providing details about the product to entice you to buy.
4. Production: Through methods such as predictive analytics, regression analysis,

Notes

budgeting, etc., data analysis helps the manufacturing industry to support all operations. The device can calculate the quantity of products to be created based on data collected and analysed through demand sampling and can increase operational efficiency and profitability in many other establishments.

5. Security: Security Analytics is an online security management method that focuses on examining data to provide proactive security measures. Data analysts provide the company with the highest level of security. No company can predict the future, especially when it comes to security threats. However, by deploying security research devices that can deconstruct security events, you can spot a threat before it happens and have a chance to damage your system and the root problem.
6. Education: In the current situation, data analysis applications in education are the main data analysts. It is most commonly used in adaptive learning, new inventions, adaptive content, and other similar applications. It is about assessing, collecting, researching, and detailing data about students and their individual situation in order to better understand and improve the learning process and the environment in which it takes place.
7. Healthcare: Data analytics in healthcare can be used to aggregate huge amounts of data in a matter of seconds to find treatment options or solutions to various ailments. Not only does this provide accurate solutions based on the recorded data, it can also provide accurate answers to specific patient concerns.
8. Military: Military data analytics solutions combine many specialised and application-specific use cases. It allows commanders and engineers to make connections between data analytics and areas like augmented reality and psychological research that advance military organisations around the world.
9. Insurance: Many data analyses are carried out as part of insurance procedures. Different types of data such as Data such as actuarial data and claims data help insurance companies determine the risk associated with insuring an individual. Analysis software can be used to identify risky claims, which can then be submitted to authorities for further investigation.
10. digital advertising: The application of data science has also changed the face of digital advertising. Data analytics and data algorithms are used in a variety of advertising media, such as digital billboards and web banners across the city.
11. Fraud and Risk Detection: The original use of data analysis may have been to detect fraud. Since they already had a large amount of customer data, they used data analysis. To determine the probability of default, data analysis was used to examine current spending trends and consumer profiles. Ultimately, this has led to a reduction in fraud and risk.
12. Travel: By evaluating social media and mobile/blog data, data analysis applications can be used to improve traveller shopping experiences. Businesses can leverage up-to-date browse-to-buy conversion rate data to create customised offers and packages based on customer preferences and desires.
13. Communications, media, and entertainment: Organisations in this space examine customer data and behavioural data simultaneously, develop content for different audiences, distribute content and analyse content performance. With the help of data analysis, customer data is collected and used and their usage habits in social media are analysed.
14. Energy and utility: In areas such as smart grid management, power distribution, energy

Notes

optimisation and building automation for other services, many energy management companies use data analysis software.

5.3 The Data Ecosystem and Languages for Data Professionals

The programming languages, packages, algorithms, cloud computing services and general infrastructure that a company uses to collect, store, analyse and consume data are referred to as the data ecosystem. No two companies use the same information in the same way. Therefore, every company has its own data ecosystem. In certain circumstances, these ecosystems may overlap, particularly when data is accessed or downloaded from a public source or when third-party providers (e.g., cloud storage providers) are used.

Components of a Data Ecosystem

1. Sensing: The process of finding the data sources for your project is called discovery. It assesses the quality of the data to determine if it is useful.
2. Collection:
   - ❖ Data should be collected once a credible data source has been identified.
   - ❖ Data collection can be manual or automatic. On the other hand, the manual collection of large amounts of data is usually not possible. For this reason, data analysts develop software to automate the data collection process using computer languages.
   - ❖ For example, you could develop code that extracts useful information from a web site (the aptly named web scraper).To extract specific information directly from a database or to interact with a web service, you can create and develop an application programming interface or API.
3. Wrangling: The term “data processing” refers to a set of techniques to transform raw data into a more usable format. This can include combining multiple datasets, finding, and filling data gaps, deleting unnecessary or inaccurate data and “cleaning” and organising the data for future analysis, depending on the quality of the data.

   Data processing can be manual or automatic, as can data collection. Manual methods can be effective if the data set is small enough. The amount of data in most big data projects is too large and requires automation.
4. Analysis: Raw data can be evaluated after it has been processed and converted into a usable format. These analyses can be diagnostic, descriptive, predictive, or prescriptive depending on the unique challenge your data project is trying to solve. Although each of these types of analysis is different, they use the same techniques and tools.

   Typically, your research begins with some form of automation, especially when dealing with large datasets. Data analysts use their knowledge to gain further insights after completing automated processes.
5. Storage: Data should be securely stored and accessible throughout the data lifecycle. The specific storage media are determined by your company’s data management processes.

   In this phase, the following key elements of the data ecosystem are used:

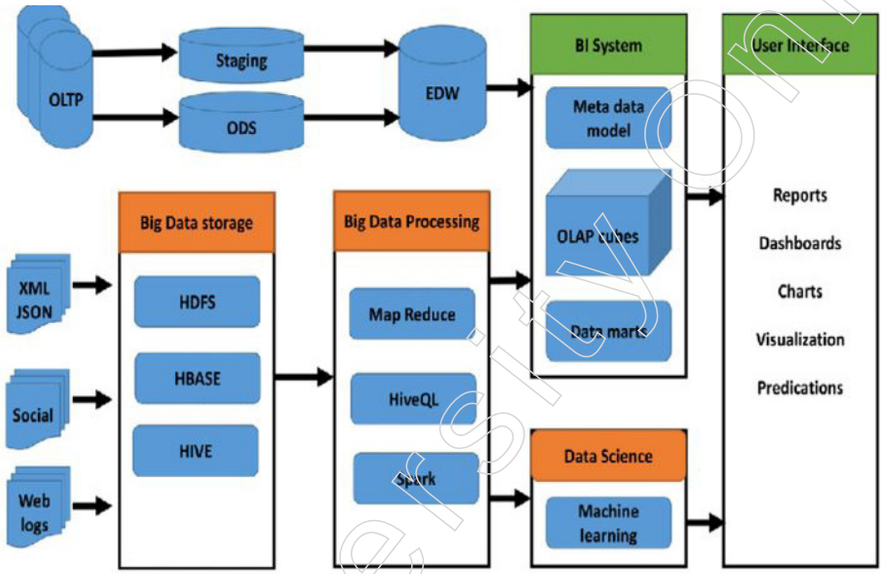
Cloud-based storage solutions: These allow a company to store data off-site and access it from anywhere.

Notes

On-site servers: These provide businesses more control over how their data is stored and used.

Other storage media: Hard drives, USB devices, CD-ROMs and floppy discs are examples of these.

5.3.1 Overview of the Data Analyst Ecosystem



The infrastructure, software, tools, frameworks, and processes used to collect, clean, analyse, mine and visualise data are part of the data scientist ecosystem. We will provide a basic overview of the ecosystem in this topic before delving into each of these topics in the following topics.

Let's start with the data: data can be categorised as structured, semi-structured or unstructured depending on how its structure is defined. Structured data has a strict format and can be neatly sorted into rows and columns. Such information can be found in databases and spreadsheets.

For example. Semi-structured data is a combination of data with consistent characteristics and data that is not strictly structured. Let's take email for example. An e-mail contains structured data such as sender and recipient names as well as unstructured data such as the e-mail body.

Unstructured Data: Complex data, mostly qualitative information that cannot be reduced to rows and columns. For example, photos, videos, text files, PDF files and documents from social networks. Data type affects the types of data repositories that can be used to collect and store data and the tools that can be used to search or analyse it.

Data comes from a variety of data sources, including relational and nonrelational databases, APIs, web services, data feeds, social platforms and tracking devices and is available in a variety of file formats.

Data Repository: This collective term includes databases, data warehouses, data warehouses, data lakes and big data repositories. Data type, format and sources affect the data repositories that can be used to collect, store, clean, analyse and extract data for analysis. For example, if you work with big data, you need a big data warehouse to store and analyse massive amounts of fast data, as well as platforms to perform complex big data analysis in real time. Languages such as query languages, programming languages

Notes

and shell and scripting languages are part of the ecosystem. They are key components in a data scientist's work environment, from accessing and manipulating data using SQL to designing data applications in Python to writing shell scripts for repetitive operational tasks. The Data Analysts ecosystem includes tools, platforms, and automated processes for all phases of the analysis process. It is a large and rich ecosystem that includes tools for collecting, extracting, transforming, and loading data into data repositories, as well as tools for data ingestion, data cleansing, exploration, analysis, and data visualisation.

5.3.2 Types of Data

Data is unstructured information that turns into something useful. Facts, observations, perceptions, numbers, letters, symbols, and images are examples of data that can be interpreted to create meaning. Data can be classified in different ways, including according to its structure.

Structured, semi-structured and unstructured data are three types of data.

Structured Data: Structured data has a well-defined structure or conforms to a specific data model, can be stored in well-defined schemas like databases and can in many cases be represented as a table with rows and columns. Structured data is a collection of objective facts and figures that can be collected, exported, archived, and organised in a database.

SQL databases and online transaction processing (OLTP) systems focus on business transactions, spreadsheets such as Excel spreadsheets and Google spreadsheets, online forms, sensors such as global positioning (or GPS) and radio frequency identification tags (or RFID) and networks and All possible sources of structured data are web server logs.

Structured data is often stored in relational or SQL databases. Standard data analysis methods and tools can also be used to explore structured data.

Semi-structured: Semi-structured data has certain organisational characteristics but does not have a stable or rigid schema. Semi-structured data cannot be stored in databases in the same way as structured data. Contains tags and elements, also called metadata, used to classify, and organise data.

Email, XML and other markup languages, binary executables, TCP/IP packets and ZIP files are just a few examples of semi-structured data sources. Data from multiple sources are combined. XML and JSON are commonly used to store and communicate semi-structured data as they allow users to define tags and attributes to store the data in a hierarchical format.

Unstructured Data: Unstructured data is clearly structured information and therefore cannot be grouped into rows and columns in a traditional relational database. It follows no particular format, order, meaning, or rule. Unstructured data can come from many different sources and be used for various business and analytical purposes.

Websites, social media feeds, images in various file formats (like JPEG, GIF and PNG), video and audio files, documents and PDFs, PowerPoint presentations, media logs and surveys are just a few examples of unstructured data sources.

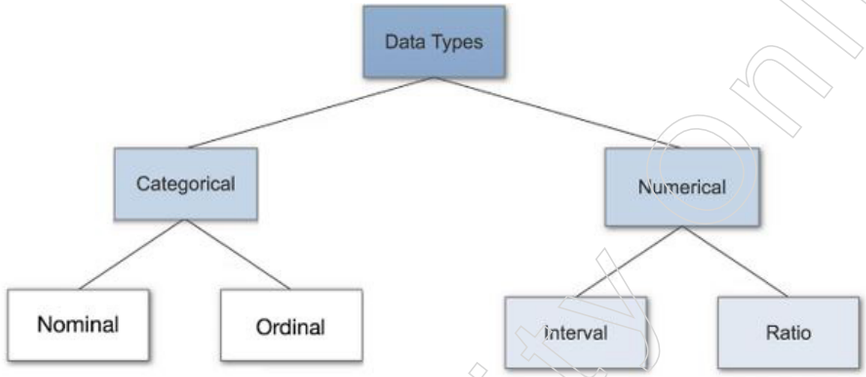
Unstructured data can be stored as files and documents (e.g. a Word document) for manual analysis or in NoSQL databases with proprietary data analysis tools.

In summary, structured data is information that is well organised in formats that can be stored in databases and that are accessible to common data analysis methods

Notes

and tools. Semi-structured data is data that is somewhat organised and relies on meta tags for grouping and prioritisation. while semi-structured data is data that is reasonably organised and relies on meta tags for grouping and hierarchy.

Unstructured data is information that is not typically grouped into rows and columns in any particular way.



In machine learning, data types are typically categorized into two main groups: categorical data and numerical data. These data types are important for feature engineering and determining which algorithms and techniques are suitable for a given dataset. Here's a breakdown of these two main types of data in machine learning:

Categorical Data:

Categorical data represents discrete, non-numeric values that belong to specific categories or classes.

Common examples of categorical data include:

- Binary: Data with two categories, such as yes/no or true/false.
- Nominal: Data with multiple categories that have no inherent order or ranking, like colours or types of animals.
- Ordinal: Data with multiple categories that have a meaningful order or ranking, like education levels (e.g., high school, college, Ph.D.).

Machine learning algorithms often require encoding categorical data into numerical format using techniques like one-hot encoding or label encoding.

Numerical Data:

Numerical data consists of continuous or discrete numeric values.

It can be further categorized into two subtypes:

- Continuous: Data that can take any real value within a given range. Examples include height, weight, or temperature.
- Discrete: Data that can only take specific, distinct values within a range, often integers. Examples include the number of siblings, the count of items in a shopping cart, or the number of bedrooms in a house.

Numerical data can be directly used in many machine learning algorithms without the need for encoding.

Additionally, it's essential to consider time series data as a separate type, especially when dealing with sequences of data points over time. Time series data includes

Notes

observations that are indexed by time or some other continuous variable, such as stock prices, temperature measurements, or sensor readings.

Understanding the types of data in your dataset is crucial for selecting the appropriate preprocessing techniques and machine learning algorithms to apply to your problem. Different algorithms are more suitable for specific data types, and preprocessing helps transform the data into a format that can be used effectively for model training and prediction.

5.3.3 Understanding Different Types of File Formats

As a data scientist, you work with many types and formats of data files. It is very important to understand the basic structure of file formats as well as their advantages and disadvantages. This knowledge will help you choose the optimal format for your data and performance needs.

In this topic we discuss some of the most common file formats: Extensible Markup Language or XML, Portable Document Format or PDF, JavaScript Object Notation or JSON, Delimited Text File Formats, Microsoft Excel Open XML Spreadsheet or XLSX.

Delimited text files are text files used to store data as text with values separated by a delimiter in each line or line. A separator is a sequence of one or more letters that defines a boundary between independent entities or values.

Any character can be used to divide the values; however, the comma, tab, colon, vertical bar, and space are the most frequent delimiters.

Comma-Separated Values (or CSVs):

The most common file types in this category are Comma Separated Values (CSV) and Tab Separated Values (TSV). The delimiter in CSV files is a comma while the delimiter in TSV files is a tab. TSV is an alternative to CSV format when literal commas are present in text data and cannot be used as delimiters. Tab stops are rare in body text. Each record in the text file is represented by a line or horizontal line with a series of values separated by a separator.

The top row serves as a column header, with each column containing a different data type. For example, one column can be of type Date while another can be of type String or Integer. Files are delimited using a standard format to provide a simple information schema and to allow field values of any length. Almost all existing applications can handle it. Delimiters are a method of defining boundaries in a data stream.

Microsoft Excel Open XML Spreadsheet or XLSX is a type of spreadsheet file created from Microsoft Excel Open XML. It is an XML-based file format created by Microsoft. In case of. An XLSX file, also called a workbook, can contain multiple worksheets. Each worksheet is divided into rows and columns, with the cell at the intersection of the two. Data is stored in each cell. XLSX uses an open file format, which means it's compatible with a wide variety of other programs.

Can use and save all Excel functions and is also one of the safest file formats because it cannot save malicious code. Extensible Markup Language or XML is a markup language for encoding data with defined rules.

humans and robots can read XML file format. It is a self-describing language for transmitting data over the Internet. In some ways, XML is similar to HTML, but there are some differences. For example, take A. Predefined tags, which are not used in XML.

Notes

HTML is one of them. Since XML is platform and programming language independent, it facilitates the exchange of data between systems.

Portable Document Format (PDF) was developed by Adobe to deliver documents independent of applications, hardware, and operating systems so that they can be viewed on any device. This format is commonly used in legal and financial documents, as well as for filling out data in forms.

JSON or JavaScript Object Notation is an open, text-based standard for transmitting structured data over the Internet. The file format is a language-independent data format and can be read by any programming language. JSON is easy to use, cross-browser compatible and widely regarded as one of the best tools for transferring data of any size and shape, including music and videos.

5.3.4 Sources of Data

Data sources have never been as dynamic and varied as they are today, as we discussed in one of our previous topics. In this section, we look at some common sources, including relational databases, flat files and XML datasets, APIs and web services, web scraping, data feeds and streams.

Companies typically use internal applications to manage day-to-day business operations, customer transactions, HR activities and workflows. Relational databases such as SQL Server, Oracle, MySQL, and IBM DB2 are used to store data in a structured manner in these systems.

Data from databases and data warehouses can be used as a source of analysis. For example, data from a retail transaction system can be used to estimate sales across multiple locations, while data from a customer relationship management system can be used to forecast sales.

Additional public and private datasets are available outside the organisation. For example, government agencies regularly release demographic and economic data. There are also organisations that sell specific data, such as B. Point-of-sale data, financial data, or weather data that companies can use to develop strategies, estimate demand, and make decisions related to sales or marketing promotions, among other things. Flat files, spreadsheet files and XML documents are common formats for of these datasets.

Flat files are text files that store data in plain text format, with a record or line by line and each value separated by delimiters such as commas, semicolons, or tabs. Unlike relational databases, which contain multiple tables, the data in a flat file results in a single table. The comma separated CSV file format is one of the most commonly used flat file formats. Table files are a type of flat file that organises data in a table format, including rows and columns. On the other hand, a worksheet can contain multiple sheets, each corresponding to a different table. Although spreadsheet data is just text, files can be saved in custom formats and contain additional information such as formatting, formulas and more.

The most common spreadsheet application is Microsoft Excel, which uses the .XLS or .XLSX file format to store data. Among them are Google Sheets, Apple Numbers and LibreOffice. Data values are recognised or tagged in XML files.

Unlike flat files, where data is “flat” or mapped to a single table, XML files can handle more complex data structures, such as hierarchical data structures. Data from online surveys, bank statements and other unstructured datasets are among the most popular uses of XML.

Notes

APIs, or application programming interfaces and web services are provided by many data providers and websites and allow different users or programs to communicate and access data for processing or analysis. Web services and APIs often listen for incoming requests from users or applications, which can be web requests or web requests and return data in plain text, XML, HTML, JSON, or multimedia files.

Let's look at some common APIs used as data sources for data analysis: Using the Twitter and Facebook APIs to collect data from tweets and posts for tasks like sentiment mining and sentiment analysis, which measure the amount of praise and Summarise criticism for a specific topic, such as government policy, product, service, or overall customer satisfaction.

Stock APIs are used for trading and analysis to retrieve data such as stock and commodity prices, earnings per share and historical prices. Data discovery and validation APIs are useful for data scientists to clean and prep data and to correlate data, for example to determine which city or state a particular zip code belongs to. APIs are also used to retrieve data from database sources inside and outside the company.

Web scraping is a technique for extracting information from unstructured sources. Online scraping, also known as screen scraping, web scraping and web data mining, allows scraping of specific data from websites based on certain parameters. Web scrapers can be used to extract text, contact information, photos, videos, product items and more from a website.

Collect detailed product information from retailers, manufacturers, and e-commerce sites for price comparison, generate leads via public data sources, extract data from posts and authors in various forums and communities and collect training and testing datasets for learning models, to name a few to name the most common applications of web scraping.

BeautifulSoup, Scrapy, Pandas and Selenium are among the most popular online scraping tools. Data streams are another popular method of combining continuous streams of data from IoT tools, devices and applications, GPS data from cars, computer programs, websites, and social networking posts. This information is usually time-stamped and geo-referenced to enable location determination.

Stock and stock tickers for financial trading, retail transaction feeds for demand management and supply chain forecasting, surveillance and video feeds for threat detection, social media feeds for sentiment analysis, sensor data streams for industrial or agricultural machinery monitoring, web click streams for networks, performance monitoring and improved design and real-time flight events for booking modification and forwarding are just a few examples of data flows and how they are used.

Apache Kafka, Apache Spark Streaming and Apache Storm are three popular streaming applications. RSS (Really Simple Syndication) feeds are another well-known data source. They are commonly used to get updated data from online forums and news sites where the information changes regularly. Updates are pushed to users' devices via a feed reader. This is an interface that converts RSS text files into an update data feed.

5.3.5 Languages for Data Professionals

We will learn some important languages in the work of data scientists. There are three types of query languages, programming languages and shell scripts. Each data scientist should be able to communicate in at least one language from each group.

Notes

Put simply, SQL, for example, is a query language for accessing and manipulating data in a database. Shells and scripting languages such as Unix/Linux Shell and PowerShell are useful for repetitive and time-consuming operational tasks. Programming languages like Python, R and Java were developed to design applications and control their behaviour.

SQL or Structured Query Language: SQL or Structured Query Language is a query language used for accessing and manipulating data in relational databases, but not only. Using SQL, we can create a series of statements to perform tasks such as inserting, updating, and deleting database entries, as well as creating new databases, tables, and views. Create stored procedures that allow you to write a series of statements and then call them.

Here are some of the benefits of using it:

- SQL: SQL is platform independent and can be used on any computer. While each provider may have different specific revisions and extensions, they can be used to search for data across many different databases and data stores. It has a simple syntax similar to English. Its syntax allows programmers to create programs with fewer lines than other programming languages, using basic keywords like “select”, “insert”, “in” and “update”.
- Can extract large amounts of data quickly and efficiently. It is based on an interpreter system that allows code to be executed as soon as it is written, allowing for rapid prototyping. One of the most commonly used query languages is SQL. Thanks to its huge user community and the huge amount of documentation created over the years, it continues to provide a standard platform to all users worldwide.
- Python: Python is a widely used high-level general-purpose, open-source programming language. Compared to some previous languages, the syntax allows developers to convey their ideas in fewer lines of code. Python is widely considered to be one of the most user-friendly programming languages and has a large programming community. It’s a great tool for novice developers due to its focus on simplicity and readability, as well as its low learning curve. It is ideal for performing computationally intensive tasks involving large amounts of data that would otherwise be time-consuming and cumbersome.

Numpy and Pandas are Python libraries that make this possible through parallel processing. It has built-in functionality for virtually every commonly used concept. Python is ideal for a wide range of applications because it supports multiple programming paradigms such as object-oriented, imperative, functional, and procedural.

Let’s look at some of the reasons why Python is currently one of the fastest growing programming languages in the world.

It’s easy to learn: Compared to other languages, Python lets you complete tasks with fewer lines of code. Python is open source, meaning it’s free and developed by the community. It works on both Windows and Linux and can be ported to different systems. It is well respected in the community and there are many excellent analysis libraries available. It includes many open-source libraries for data manipulation, visualisation, statistics, and math to name a few.

Pandas for data cleaning and analysis, Numpy and Scipy for statistical analysis, Beautifulsoup and Scrapy for web scraping, Matplotlib and Seaborn for visualising data in the form of bar charts, histograms, and pie charts and Opencv for image processing. its many libraries and functions.

Notes

R:R is a language and environment for data analysis, data visualisation, machine learning and statistics. It is known for its ability to create beautiful images, which gives it an edge over other languages in this field. It is commonly used to design statistical applications and perform data analysis.

The R offers a number of significant advantages, including:

- is an open-source, platform-independent programming language. Many programming languages, including Python, can be used with it.
- is highly extensible, meaning developers can add new functionality by specifying new features.
- Allows management of both structured and unstructured data, giving them richer data capabilities.
- Contains libraries like Ggplot2 and Plotly that provide users with visually pleasing plot plots. You can create reports with data and scripts, as well as interactive web applications.

Among other programming languages, it is the most popular for creating statistical tools. Microsystems developed Java, an object-oriented, class-based, and platform-independent programming language. It is currently one of the most used programming languages. Java is used in various data analysis techniques, including data cleansing, data import and export, statistical analysis and data visualisation.

In fact, most major big data platforms and tools such as Hadoop, Hive and Spark are written in Java. It's perfect for projects that need to be completed quickly.

A computer program developed for a UNIX shell is called a Unix/Linux shell. This is a set of UNIX commands written in plain text format to perform a specific operation. Writing a shell script is quick and easy. This is best for repetitive tasks where entering one line at a time would take too long. Shell scripts can do a variety of things including file manipulation, program execution, system administration including disk backup and registry evaluation, advanced application configuration scripts, routine backups, batch processing and more.

Microsoft PowerShell is a cross-platform automation tool and configuration framework designed to work with structured data formats such as JSON, CSV, XML and REST APIs, as well as websites and desktop applications. It has a command line shell and scripting language. Because PowerShell is object-based, you can filter, sort, measure, group, compare and do more with objects as they move through your data pipeline. It is also useful for exploring data, generating graphical user interfaces (GUIs), and creating interactive charts, dashboards, and reports.

5.4 Understanding Data Repositories and Big Data Platforms

A data repository is a term that refers to data that has been collected, organised, and extracted for use in commercial operations or exploration for reporting and data analysis purposes. A database infrastructure can be large or small and can include one or more databases that collect, manage and archive records. In this topic we give you an overview of the many types of repositories that can store your data, e.g., B. databases, data warehouses and big data stores.

5.4.1 Overview of Data Repositories

A data repository is a term that refers to data that has been collected, organised, and extracted for use in commercial operations or exploration for reporting and data analysis

Notes

purposes. A database infrastructure can be large or small and can include one or more databases that collect, manage and archive records.

We provide an overview of the many types of repositories that can store your data, such as B. databases, data warehouses and big data stores.

A database is a collection of data or information used to enter, store, search for and modify data. A database management system (DBMS) is a set of tools for creating and managing databases. Query is a function that allows you to store, manipulate and retrieve information from a database. For example, if you use the query function to search for customers who have been inactive for six months or more, the database management system will retrieve all customers from the database who have been inactive for six months or more.

Although the terms database and DBMS have different meanings, they are often used interchangeably. Databases come in all shapes and sizes. Data type and structure, query mechanisms, latency requirements, transaction rates and expected data usage all affect database selection.

There are two types of databases to consider: relational and non-relational databases. Relational databases, often referred to as RDBMS, are based on the principles of flat file organisation, with data organised in tabular form with rows and columns in a well-defined structure and schema.

RDBMS, on the other hand, is designed for data queries and operations that involve multiple tables and much larger amounts of data than flat files. A common query language for relational databases is SQL or Structured Query Language. Another option is non-relational databases, commonly referred to as NoSQL or “Not Only SQL”.

Non-relational databases emerged in response to the size, variety and speed at which data is being generated today, primarily due to advances in cloud computing, the Internet of Things, and the proliferation of social media. Non-relational databases that are fast, flexible, and scalable have enabled users to store data in a schema-free or free-form format. NoSQL is commonly used to process large amounts of data.

The Data Warehouse serves as a central repository for combining data from multiple sources and consolidating it into a comprehensive database for analytics and business intelligence using the ETL (Extract, Transform and Load) process. At its most basic level, the ETL process helps you pull data from various sources, transform it into a clean, usable state and upload it to your enterprise data repository. Ideas

Data mart and data lake, which we will discuss later, refer to data warehouses. Because most traditional business data is stored in relational databases, data warehouses and data warehouses are traditionally relational.

With the advent of NoSQL technology and new data sources, non-relational data repositories are now being used for data warehousing.

Big data stores are a type of data store that includes a distributed storage and computing infrastructure for storing, scaling, and processing large amounts of data. In general, data repositories act as data stores that help isolate data and make reporting and analysis more efficient and reliable.

5.4.2 Data Marts, Data Lakes, ETL and Data Pipelines

Data Mart is a simple data warehouse focused on a topic or industry. Because they don’t have to waste time trawling through a more complex data warehouse or manually

Notes

gathering data from multiple sources, teams can access data faster and uncover insights using the data warehouse.

What is the purpose of the data warehouse?

The database gives a single team or division of your organisation faster access to data. This holiday season, if your marketing team is looking for data to improve campaign performance, reviewing and combining data from multiple platforms can be time-consuming, inaccurate, and ultimately costly.

When tasked with finding data from multiple sources, teams often use spreadsheets to communicate and collaborate. Human error, uncertainty, difficult reconciliations, and multiple sources of truth are common consequences - the so-called “table nightmare”. Data warehouses are growing in popularity as a central place where relevant data is collected and organised before reports, dashboards and visualisations are created.

Advantages of a Data Warehouse

The data mart, dedicated to a specific team or sector, offers several advantages:

Single Source of Truth: The centralised data warehouse design ensures that everyone in a department or organisation makes decisions based on the same information. This is a significant advantage as the data and predictions based on it are reliable and stakeholders can focus on making decisions and taking action rather than debating the facts.

Faster access to data. Business teams and specific users can quickly pull the data they need from the enterprise data warehouse and combine it with data from a variety of other sources. You can get fresh data from the data mart when you need it, without having to periodically contact IT to request dumps after connecting to the data sources you want. As a result, business and IT teams benefit from increased productivity.

Faster insights lead to faster decision making: While the data warehouse enables enterprise-level decision making, the data warehouse enables service-level data analysis. Analysts can focus on specific challenges and opportunities in areas like finance and human resources, move from data to insights faster and make better decisions faster.

Easier and Faster to Implement Setting up an enterprise data warehouse that meets all of your organisation’s needs can take a lot of time and effort. On the other hand, a data warehouse is designed to meet the needs of specific business teams and requires access to fewer types of data. This facilitates and accelerates implementation.

Building flexible and scalable data management: Data warehousing is a flexible data management system that adapts to changing business needs, such as the ability to leverage information gained in previous projects to assume current responsibilities. Based on new and evolving analytics projects, teams can update and customise their data warehouse.

Intermediate Analysis: Some data analysis activities are only temporary, such as the detailed analysis of online sales during a two-week campaign before a team meeting. To carry out such a project, teams can quickly build a data warehouse.

Data Lake: Data Lake is a central repository that can store structured and unstructured data of any scale. Dashboards and visualisations help you make better decisions and perform different types of analysis, from big data processing to real-time analytics and machine learning, without having to organise your data first.

Notes

Companies that can increase the business value of their data will outperform their competitors. According to the Aberdeen survey, companies implementing Data Lake outperformed their peers in organic revenue growth by 9%.

These executives were able to perform new forms of analysis, such as machine learning, by leveraging new data sources in the data lake, such as log files, clickstream data, social media, and internet-connected devices. By attracting and retaining customers, increasing productivity, proactively maintaining equipment, and making informed decisions, they have been able to identify and capitalise on business opportunities faster.

Extract Transform Load (ETL): The value of using this data in analytics, data science and machine learning programs to generate business insights increases with the amount of data, data sources and data types in organisations. Since transforming raw and dirty data into clean, fresh, and reliable data is a critical step before undertaking these actions, the need to prioritise these tasks places increasing demands on data development teams.

ETL (Extract, Transform and Load) is a data engineering process that involves extracting data from various sources, transforming it into useful and reliable resources and loading it into systems that end users can access and use to solve business problems.

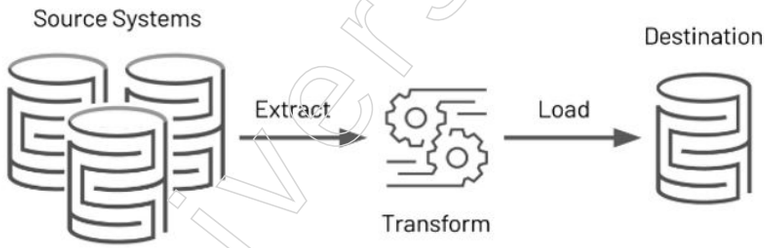


Figure: ETL Process

How Does ETL Work?

Extract

The first step in this process is to extract data from heterogeneous sources such as business systems, APIs, sensor data, marketing tools, transactional databases, and other sources. As you can see, some of these data types are semi-structured JSON server logs, while others are structured outputs of commonly used systems. The extraction can be done in different ways: There are three methods of data extraction:

Partial Retrieval: When the source system warns you that a record has changed, this is the easiest approach to retrieve the data.

Partial recovery (with update notification): Although not all systems can notify of an update, they can indicate which records have changed and provide a digest of the updated data.

Statement complete: Some systems cannot determine which data has changed. In this case, the only way to extract data from the system is to perform a deep extraction. This method requires that you have a copy of the previous declaration in the same format so that changes can be identified.

Transformation

The raw data collected from the sources is then converted in a second step into

Notes

a format that can be used by different applications. In this phase, the data is cleaned, mapped, and converted, often into a specific schema, to meet business needs. Various types of transformations are used to ensure data quality and integrity.

Data is often inserted into a temporary database instead of being loaded directly into the target data source. This step ensures quick recovery in case something goes wrong. This is where you can prepare audit reports for compliance, as well as identify and correct any errors in the data.

Upload

Finally, the loading function is the process of transferring the converted data from the staging area to the target database, which may or may not have existed before. This method can be simple or complex depending on the application needs. You can use ETL tools or custom code to run any of these processes.

Data streams

An ETL pipeline (also called a data pipeline) is a system that allows you to perform ETL tasks. Data pipelines are a set of tools and activities to move data from one system to another where it can be stored and managed differently. In addition, pipelines allow you to automatically collect, transform and consolidate data from disparate sources into a single, powerful data warehouse.

Challenges with ETL

While ETL is essential, with the exponential growth of data sources and types, building and maintaining reliable data pipelines has become one of the most challenging aspects of data engineering. Building data-true pipelines is slow and difficult from the start. Complex programming and limited reusability are used to create data pipelines.

Even if the underlying code is fairly identical, a pipeline designed in one environment cannot be used in another, leaving data engineers with a bottleneck, and having to spin the wheel every time. In addition to building pipelines, managing data quality is a challenge in increasingly complex pipeline systems. Bad data often goes unnoticed in the pipeline, reducing the value of the data set.

Data engineers should create extensive custom code to perform quality checks and validations at every stage of the pipeline to maintain quality and deliver reliable insights. Eventually, as pipelines grow larger and more complex, organisations face greater operational overhead to manage, making it extremely difficult to maintain data reliability. The computing infrastructure needs to be configured, scaled, rebooted, patched, and upgraded, which takes time and money.

Due to lack of visibility and tools, pipeline failures are difficult to detect and even more difficult to repair. Regardless of these roadblocks, a robust ETL process is essential for any business that wants to be insight driven.

Without ETL solutions that maintain a foundation of data trust, business teams are forced to make decisions based on guesswork and unreliable metrics and reports. Data engineers need tools to simplify and democratise ETL, simplify the ETL lifecycle and enable data teams to build and manage their own data pipelines to quickly gain insights for advancement.

Difference between data marts, data lakes and data warehouses

Notes

Data warehouses, data lakes and data warehouses have different purposes and requirements.

Data Warehouse is a data management system that supports companies in their business and analysis. Data warehouses often store large amounts of data, especially historical data. Data in the data warehouse typically comes from a variety of sources, including application log files and transactional applications. A data warehouse is a place to store structured data with a well-defined function.

Data Lake enables companies to store massive amounts of structured and unstructured data (e.g., from social media or click data) and make it immediately available for real-time analysis, data science and machine learning. In a data lake, data is consumed in its raw state with no modifications.

The main difference between a data lake and a data warehouse is that data lakes contain large amounts of unstructured data. Organisations don't need to know in advance how their data will be used.

A data warehouse is a basic type of data warehouse that focuses on a single object or business area, such as sales, finance, or marketing. Because of their importance, data warehouses use fewer sources than data warehouses.

5.4.3 Viewpoints: Considerations for choice of Data Repository

There are a few things to consider when choosing the right database for the job. You have to consider the use case. What is a data repository for? Will it be used to store structured data, semi-structured data, or unstructured data? Or maybe you already know what a data schema is? Are there any conditions to meet? Do you work with static, flowing or moving data? Is data encryption necessary? It is here... Do you know what data you are working with? Do you need a big data system? What are the storage requirements? Does the data need to be updated and accessed frequently, or should it simply be archived and stored in a vault for an extended period of time, for backup purposes, for example?

It is therefore possible that your company has established guidelines for databases or data stores that you can use for certain types of work. Keep all of these considerations in mind. Therefore, we take these considerations into account when deciding which data store to use. We're investigating what kinds of features this data repository should support.

The next question is what kind of access we need. Do we use it infrequently or for long running queries? Will it be primarily used for transaction processing or analysis, archiving, or data warehousing? We also pay attention to compatibility. How does this new data repository fit into the current ecosystem of programming languages, tools, and processes?

We also take into account the safety measures of this standard. The most important factor is scalability. We may be happy with current performance, but is it scalable? Is it customisable to the needs of the organisation? We don't generally have the ability to choose the type of data repository that the company uses and very few companies currently use just one data repository.

companies use the Enterprise Relational Database as their database of choice. For some small projects and microservices, we prefer to use an open-source relational database. Then there's the unstructured data source we love. So those are the three

Notes

most important. The most important thing is to consider the skills you have or want to develop in your company. Also consider the costs associated with the different options.

However, other projects use separate projects. Other projects have updated it for open source several times. Other projects go in different directions depending on where we want to be. And all of that.... The hosting platform makes a difference because it's no longer just a question of whether one wants to use IBM Db2 or another vendor's database like Microsoft SQL Server or whatever.

It's not a choice between the two options. If companies do this, does the company want to do this on AWS RDS? Maybe we should check out Amazon's Aurora. Maybe he should look at Google's relationship services. There are so many options that you must try.

The next question is how to store the data. A decision is made about how to retrieve the data and then a decision about where to retrieve the data. When it comes to data storage, all of these things need to be considered.

The structure of the data, the type of question and the amount of data entered into the database are all aspects that influence the type of data source to be chosen. In most cases, a relational database is sufficient; However, there will be situations where relational databases such as IBM Db2, Oracle or Postgres will not suffice. In certain circumstances, document repositories like MongoDB or column-wide repositories like Cassandra might work for you depending on your use case.

For example, if you process gigabytes or terabytes of data per day. On the other hand, graphical data structures like Neo4J or Apache TinkerPop can be great for you if you want to build product recommendation engines or illustrate a web of interactions between different people in social networks.

If you want to analyse terabytes or petabytes of data at the same time, the Hadoop engine with MapReduce is a good option. Therefore, regardless of the use case, choosing the right database or data source depends on the type of application, the amount of data and the data structure.

5.4.4 Foundations of Big Data

Everyone leaves a fingerprint in today's environment. The growing number of internet-connected devices we use every day collects vast amounts of data about us, from our travel habits to our physical and entertainment activities and it even has a name: big data.

Big Data: Here is Ernst and Young's definition: Dynamic, huge, and divergent amounts of data generated by people, tools and machines are called Big Data. Sophisticated and scalable new technologies are required to collect, store, and analyse the vast amounts of data to generate real-time business insights into customer information, risk, profit, efficiency, productivity management and increased shareholder value.

Big Data is a collection of structured, semi-structured and unstructured data that can be mined to generate insights and used for machine learning, predictive modelling, and other advanced analytics initiatives. Big data processing and storage systems, as well as technologies that facilitate the analysis of large amounts of data, have become an integral part of enterprise data management architecture.

While there is no single definition of Big Data, each has distinct elements, such as: B. Speed, Quantity, Variety, Veracity and Value. V Big Data are as follows: The

Notes

speed of data collection is called speed. Data is generated at breakneck speed in an endless process. With the help of almost real-time, local, and cloud-based streaming technologies, information can be processed very quickly.

The amount of data or the increase in the amount of data stored is referred to as volume. Growing data sources, higher resolution sensors and a scalable infrastructure are increasing the volume. The diversity of data is its diversity. Structured data like tweets, blog posts, images, numbers, and videos fit perfectly into the rows and columns of relational databases, but unstructured data like tweets, blog posts, images, numbers, and videos just don't fit. certain fashion.

Data comes from a variety of sources, including machines, people, and processes, both inside and outside organisations. Mobile technologies, social media, wearable technologies, GPS technologies, video and a multitude of other factors are driving forces. The quality and origin of the data as well as their topicality and accuracy are aspects of truthfulness.

Consistency, completeness, integrity, and ambiguity are just some of the attributes. Cost and the need for traceability are the two main reasons. The debate about data accuracy in the digital age continues due to the large amount of data available. Is the information correct or incorrect? Our ability and need to turn data into value is called value. Profit is not the only factor that determines value.

May provide medical or social benefits in addition to consumer, worker, or individual satisfaction. The main motivation for people who want to learn more about Big Data is to make use of it. Let's look at some examples of V in action.

Velocity: Hours of footage are posted to YouTube every 60 seconds, producing data. Consider how much data collects over the course of hours, days, and years.

Volume: The world's population is estimated to be around 7 billion people, with the vast majority utilising digital gadgets. Mobile phones, desktop and laptop computers, wearable devices and so on are all examples of electronic devices. Every day, these gadgets generate, capture and store 2.5 quintillion bytes of data. That's the same as ten million Blu-ray DVDs.

Variety: Let's consider the many forms of data. Text, images, movies, sound, health data from wearable devices and a variety of other sorts of data from internet of things devices Veracity. We must find techniques to produce dependable and accurate insights because 80% of data is considered unstructured. The information needs to be organised, evaluated, and visualised.

Data analysts today use big data to gain insights and solve problems that arise from these vast amounts of data. Due to the amount of data collected, traditional data analysis tools are not applicable; However, new methods that use distributed computing power can solve this problem.

Apache Spark, Hadoop, and their ecosystem provide methods to extract, load, analyse and process data across distributed computing resources, resulting in new insights and insights. This gives companies more opportunities to communicate with customers and improve their services. So, remember, the next time you put on your smartwatch, unlock your phone, or record your workout, your data will embark on a journey that could take it halfway around the world, through big data analytics and back to you..

Notes

5.4.5 Big Data Processing Tools

Big data processing technologies allow you to work with huge amounts of structured, semi-structured and unstructured data to extract value from it. We will cover Apache Hadoop, Apache Hive and Apache Spark, three open-source technologies that play an important role in big data analytics.

Hadoop: Hadoop is a set of technologies for storing and analysing large amounts of data in a distributed manner. Hive is a Hadoop-based data warehouse for data exploration and analysis. Spark is a distributed data analytics platform for performing advanced real-time data analytics.

Hadoop is an open-source Java-based platform for storing and processing large datasets in computer clusters. A node is a single computer in a distributed Hadoop system, while a cluster is a collection of nodes. Hadoop scales from a single node to an infinite number of nodes, each with their own storage and processing capabilities. Hadoop is a reliable, scalable, and cost-effective data storage system that requires no formatting.

Hadoop can be used to: Process new data formats, including streaming audio, video, social media comments and click data, as well as structured, semi-structured and unstructured data not typically used in a data warehouse.

All stakeholders should have real-time self-service access. Consolidate enterprise-wide data and migrate cold data to a Hadoop-based system to optimise and reduce costs in your enterprise data warehouse. The Hadoop Distributed File System (HDFS) is a big data storage system running on shared network-connected hardware and is one of the four main components of Hadoop.

By dividing files across multiple nodes, HDFS provides scalable and reliable storage for large datasets. Split large files across multiple computers so they can be viewed in parallel. This allows calculations to be performed in parallel on each node where the data is stored. It also prevents data loss by replicating blocks of files across multiple nodes, making it fault tolerant.

Let's look at an example to see how this works.

Imagine a file containing the phone numbers of everyone in the United States. the numbers of people whose last name begins with the letter A can be entered on server 1, B on server 2 and so on. Parts of this directory would be stored in the cluster using Hadoop. Your application would need blocks from each server in the cluster to rebuild the entire directory. By default, HDFS replicates these smaller pieces to two back-to-back servers, ensuring availability in the event of a server failure.

In addition to increased availability, this has many advantages. It allows a Hadoop cluster to break up the work into small pieces and run those tasks across all machines in the cluster for better scalability. Finally, you benefit from data locality, which moves computation closer to the node where the data is stored. This is essential when working with large amounts of data as it reduces network congestion and increases throughput.

Some of the other benefits of using HDFS are listed below:

- HDFS is designed to automatically detect and fix hardware problems for fast recovery.
- HDFS offers fast data transfer, the data can be accessed continuously.

Notes

- ❖ HDFS can span hundreds of nodes or computers in a single cluster and contain huge data sets.
- ❖ HDFS is portable across hardware platforms and compatible with multiple underlying operating systems to ensure a high level of portability.

Hive: Hive is open-source data warehousing software that allows you to read, write and manage large datasets stored in HDFS or other data storage systems such as Apache HBase. Hadoop is designed for long sequential scans and because Hive is built on Hadoop, queries have relatively high latency, making it less suitable for applications that require fast responses.

Hive is not designed for transaction processing, which often involves many writes. Best suited for data warehousing operations such as ETL, reporting and data analysis, Hive provides features that simplify access to SQL data.

Hive should allow non-SQL-savvy developers to interact with petabytes of data through a SQL-like interface called HiveQL. Traditional relational databases are designed for interactive querying of small to medium-sized data sets and struggle to process large data sets. On the other hand, thanks to batch processing, Hive runs fast in a huge, distributed database.

Hive converts HiveQL queries into MapReduce or Tez jobs that run on Yet Another Resource Negotiator, Apache Hadoop's Distributed Job Scheduling (YARN) architecture. Searches for data stored in a distributed storage system, e.g., B. the distributed file system "HDFS" from Hadoop. or Amazon Simple Storage Service (S3). Hive tracks its database and table metadata in a metastore, a database or file-based storage system that allows for easy data query and discovery.

Hive provides HCatalog, a storage and table management layer that reads data from the Hive metastore to allow Hive, Apache Pig and MapReduce to work seamlessly together. HCatalog uses a metastore to allow Pig and MapReduce to use the same data structures as Hive, eliminating the need to redefine metadata for each engine. WebHCat, a RESTful API for HCatalog, can be used by custom applications or third-party integrations to access and reuse Hive metadata.

Spark: Spark is a general-purpose data processing engine capable of mining and processing massive amounts of data for a variety of applications including interactive analytics, stream processing, machine learning, data integration and ETL. It uses in-memory processing to significantly speed up calculations by only moving data to disk when memory is limited. Java, Scala, Python, R and SQL are some of the programming languages supported by Spark.

Can run standalone or on top of other frameworks such as Hadoop with standalone clustering technology. It can also access data from a variety of data sources including HDFS and Hive, making it extremely flexible. The main use case for Apache Spark is the ability to quickly process streaming data and perform complex analysis in real time.

Apache Spark is a distributed open-source big data solution. For fast analytical queries on data of any size, use caching and efficient query execution. It provides code reuse for multiple tasks (batch processing, interactive queries, real-time analytics, machine learning and graphical computation) and exposes programming APIs in Java, Scala, Python and R is used by companies around the world including FINRA, Yelp, Zillow, DataXu, Urban Institute and CrowdStrike. With 365,000 Meetup members in 2017, Apache Spark has become one of the most popular platforms for distributed big data processing.

Notes

The key features of the Spark platform software are Spark tools used for efficient and scalable data processing in big data analysis. The Apache license is used for open-source Spark. It includes five core data processing tools including GraphX, MLlib, Spark Streaming, Spark SQL, and Spark Core.

GraphX chart data analysis tool is designed to manage and manage chart data. The MLlib Spark tool is used to implement machine learning on a distributed dataset. Spark Streaming, on the other hand, is used to process the streaming data. Spark SQL is the most popular tool for structured data analysis. RDD (Resilient Data Distribution) is supported by Spark Core.

5.5 Gathering Data

The process of gathering data for use in business decisions, strategic planning, research, and other purposes is called data collection. This is an important part of data analysis and research projects: effective data collection provides the information you need to answer questions, examine business performance or other results, and predict future outcomes, trends, activities, and future scenarios.

In companies, data is collected on several levels. During transactions and data entry, information systems collect data about customers, employees, sales, and other aspects of business operations. Customer reviews are also collected through surveys and social media monitoring. Data analysts, other analysts and business users then collect relevant data from internal systems and, if necessary, also from external data sources for research. This final process is the first step in data preparation, which involves gathering information and preparing it for use in BI and analytics applications.

5.5.1 Identifying Data for Analysis

The first step in identifying data is to decide what information you want to collect. You will decide at this stage

(a) the exact information you need, e.g.,

(b) data sources you will use to obtain them.

Consider the example of a company that wants to implement targeted marketing based on the age group that buys most of its products. Your goal is to create a reach that appeals to this target group and motivates them to convince their friends and colleagues to buy these products.

Customer Profile, Purchase History, Location, Age, Education, Occupation, Income and Marital Status are just a few examples of the clear information you will find from this use case. You can also collect customer complaints data for this segment to understand what challenges they are facing as this may discourage them from promoting your products to ensure they get even more insight into this segment.

Customer service survey ratings can help you determine how satisfied they are with the resolution of their problems. If you dig a little deeper, you might want to see how these customers are discussing your products on social media and how many of their contacts are participating in these discussions, e.g., B. how many likes, shares and comments they receive on their publications.

The next step in the process is to develop a data collection strategy. You have to spend time collecting the identified data. Some of the data you require is required on an ongoing basis, while others are only required for a specific period of time.

Notes

Real-time update of numbers may be necessary when collecting data about website visitors, for example. On the other hand, if you're collecting data for a specific event, you have a fixed start and end date. At this stage, you can also determine how much data you need for reliable analysis. Whether the volume is defined by a segment, e.g. For example, all customers aged 21-30, or a data set of one hundred thousand customers aged 21-30.

You can also use this phase to identify dependencies, risks, risk mitigation plans and many other relevant elements of your initiative. The aim of the plan should be to provide the clarity needed for its implementation. In the third step of the process, you must decide how you will collect the data. Here's how to get the data you need.

You decide how to get data from a variety of data sources of your choice, e.g., B. internal systems, social networks, and external data providers. The type of data, the time period for which you need the data and the amount of data will affect your methods. Once you have decided on your data collection plan and methods, you can put your data collection strategy into action and start collecting data. If the situation changes while implementing your plan on the ground, you need to continuously adapt your plan.

The data you collect, the source of that data and the methods you use to collect it affect its quality, security, and privacy. None of these considerations are ad hoc; All apply to the full life cycle of the data analysis process. Working with data from multiple sources without evaluating it against a quality metric is a recipe for disaster.

To be reliable, data must be error-free, accurate, complete, relevant, and available. To ensure that your analysis is based on quality data, you must first provide quality attributes, measurement, and control points. You should also pay attention to data governance issues, including security, regulation, and compliance. Data usability, integrity and availability are subject to data management policies and procedures.

Failure to comply can result in millions of dollars in fines and damage not only agreements but also the credibility of the organisation. Another important factor to consider is data security. When collecting data, check the boxes for privacy, user licenses and regulatory compliance. It is necessary to plan the inspection, validation, and audit trail.

A loss of confidence in the data used for analysis can jeopardise the process, leading to skewed results and penalties. Finding the right data is a crucial step in the data analysis process. This ensures that you can look at the topic from different angles and, if implemented correctly, your conclusions are reliable and believable.

5.5.2 Data Sources

Data sources can be primary, secondary, or external data sources, which can be internal or external to the company. Let's look at some examples to understand what we mean by primary, secondary and external data sources. The word "raw data" refers to information that comes directly from a source. This can come from internal sources such as company data, CRM, HR, or workflow systems.

Data collected directly through surveys, interviews, observations, and focus groups can also be included. Information that comes from existing sources such as external databases, research articles, publications, internet research and training materials, or financial records and is available as public data is called secondary data.

Primary, secondary and third-party data can be found in databases. Internal applications are used by most companies to manage operations, workflows, and customers. External databases can be rented or purchased on a subscription basis.

Notes

Many companies have migrated or are in the process of migrating to the cloud, which is rapidly becoming a source of real-time data and on-demand information. The Internet is a repository of publicly available data that businesses can use. Individuals for non-commercial or commercial use.

The Internet is a vast repository of publicly available data. This can include publicly available manuals, government papers, documents, and articles, as well as social media and interactive platforms such as Facebook, Twitter, Google, and YouTube. Instagram is becoming an increasingly popular source of user data and opinions. These data sources are used by companies to obtain quantitative and qualitative information. Customers who are already customers and those who want to become one.

Wearable devices, smart buildings, smart cities, smartphones, medical devices and even home appliances generate sensor data, which is often used as a data source. Individuals, companies, and governments can be both data providers and data consumers in data exchanges that are sources of third-party data, which includes the voluntary sharing of data between data providers and data recipients.

Data from business applications, tracking devices, social media activity, location data and customer behaviour data can be exchanged. Questionnaires are distributed to a specific group of people as part of a survey. For example, determine if existing customers are willing to pay for a new version of a product. Surveys can be conducted online or on paper.

Census data is also a popular source of household data such as wealth and income and population data. Interviews are a good way to get qualitative information like opinions and experiences from people. Consider a conversation to find out what problems a customer service representative faces on a daily basis.

Interviews can be conducted by telephone or by direct observation. Participants are monitored at a specific location or while performing a specific learning task. Observe how consumers move around the e-commerce site, for example for analysis purposes.

They are also constantly evolving. By leveraging additional and external data sources to supplement your core data, you can discover challenges and solutions in new and relevant ways.

5.5.3 How to Gather and Import Data

In this topic, we will explore many ways and tools to extract data from the data sources discussed earlier in this course, such as: databases, networks, sensor data, data exchange and many other data sources. We will also discuss how to import data into different types of data repositories.

In computing, data is information that has been translated into a form that is efficient for movement or processing. Relative to today's computers and transmission media, data is information converted into binary digital form. It is acceptable for data to be used as a singular subject or a plural subject. Raw data is a term used to describe data in its most basic digital format.

SQL stands for Structured Query Language and is a query language for extracting data from relational databases. Among other things, SQL provides simple commands to specify the data to be retrieved from the database and the table to be retrieved, to group records with matching values, to set the order in which query results are displayed and to limit the number of results returned per query can add more features and functionalities.

Notes

You can use SQL queries or SQL-like tools to query non-relational databases. Some non-relational databases have their own query tools, such as Cassandra's CQL and Neo4J's GraphQL.The

API (Application Programming Interface) is commonly used to extract data from multiple data sources. APIs are used by applications that request data and access an endpoint that contains data. Examples of endpoints are databases, web services and data marketplaces. APIs are also used to validate data.

For example, a data scientist can use an API to validate postal addresses and zip codes. Web scraping, also known as screen scraping or web harvesting, is a technique for extracting specific data from websites based on predefined criteria.

Web scraping is a technique for extracting data from a website, such as text, contact information, photos, videos, podcasts, and product listings. RSS feeds are another popular source for getting the latest data from online forums and news sites where information is regularly updated.

Data Streams is a popular method of collecting continuous streams of data from instruments, IoT devices and applications, including GPS data from cars. Data is also retrieved from social networking sites and interactive platforms using feeds and data feeds.

Data sharing platforms allow data providers and consumers to share information. Data exchange follows a clearly defined set of standards, protocols, and data transmission formats. These platforms not only facilitate the exchange of data, but also ensure data security and control. They provide data licensing, anonymisation and protocols to protect personal data, a legal framework, and a secure analytics environment.

AWS Data Exchange, Crunchbase, Lotame and Snowflake are just a few examples of great data exchange solutions. Various alternative data sources can be used for special data requirements. For example, research firms like Forrester and Business Insider are believed to provide solid statistics on marketing trends and ad spend.

For example, Gartner and Forrester are well-known research and advisory firms that provide strategic and operational advice. There are many well-known names in user behaviour data, mobile and online usage, market research and demographic studies.

Before data can be accessed, explored, and analysed, it must first be discovered and extracted from various data sources and then downloaded or imported into a data repository. The import process combines data from multiple sources to create a unified view and interface for querying and manipulating data. Depending on the type of data, the amount of data and the nature of the target repository, different tools and procedures may be required.

Some data types are better suited to certain data stores. Structured data is stored in relational databases that have a well-defined schema. Structured data can only be stored in a relational database as a target system, such as data from OLTP systems, spreadsheets, online forms, sensors, networks, and web logs. NoSQL can also be used to store structured data.

Semi-structured data such as email data, XML files, compressed files, executable binaries, and TCP/IP protocols have certain organisational characteristics but no fixed schema. NoSQL clusters can store semi-structured data. Semi-structured data is often

Notes

stored and exchanged using XML and JSON. JSON is also the most popular data format for web services.

Data from websites, social media channels, photos, videos, documents, media logs and surveys are examples of unstructured data that lack a frame and cannot be organised into a chart. Data lakes and NoSQL databases are efficient options for storing and manipulating large amounts of unstructured data.

All kinds of data and schema can be placed in data lakes. Data pipelines and ETL technologies provide automated operations that streamline the data import process. Data import is typically done using tools such as Talend and Informatica and computer languages such as Python and R and related libraries.

5.6 Data Wrangling

The process of cleaning, organising, and enriching raw data into a desired format for better decision making in less time is called data wrangling. Data disputes are becoming more and more common in large companies today. Data has become more diverse and unstructured, requiring more time to sort, cleanse and organise the data before more detailed analysis. At the same time, since almost every business decision depends on data, business users have less time to wait for data prepared by technical staff.

This requires a self-service paradigm and a shift from IT data preparation to a more democratised data collection model. Using this self-service paradigm of data collection tools, analysts can process more complex data faster, create more accurate answers and make more accurate judgments. With this capability, more and more companies are using data processing technology to prepare data for analysis.

5.6.1 What is Data Wrangling?

The process of cleaning, organising, and transforming raw data into a format that analysts need to make quick decisions is called data shuffling. Data management, also known as data cleansing or data ingestion, enables organisations to work with more complex data, deliver more accurate results and make better decisions in less time. Depending on the data and the goal you want to achieve, the exact procedures will vary from project to project. More and more companies are using data processing solutions to prepare data for further analysis.

Wrangling is the process of 'processing' data (aka 'munging', hence the alternative term 'data wrangling techniques') and making it usable (or usable) to the system. You can think of this as part of the preparation process for other data operations.

mapping is often done in conjunction with data wrangling. The part of the reasoning process that identifies the source data fields with their respective target data fields is called "data mapping". While wrangling is about transforming data, mapping is about connecting the dots between different elements.

Benefits of Data Wrangling

- Data wrangling improves data usability by converting data into a format that is suitable with the end system.
- It enables users to create data flows quickly and easily using an easy user interface, as well as plan and automate the data-flow process.
- Combines numerous sorts of data and their sources (like databases, web services, files, etc.)

Notes

- Allow people to effortlessly analyse vast amounts of data and exchange data-flow methodologies.

Steps in Data Wrangling

Data wrangling is an iterative process, similar to most data analytics processes, in which the data engineer repeats these steps to arrive at the intended predictions. Data wrangling can be broken down into six steps:

1. Discovering: Before imputing data, the first step in data wrangling is to analyse it. Wrangling must be done in a methodical manner, based on a set of criteria that can be used to demarcate and split the data - these criteria are identified in this step.
2. Structuring: In the vast majority of cases, the raw data collected as user information is unstructured. The data should be restructured in a way that is more appropriate for the analytical method being employed. The data should be separated into categories based on the first step's categorisation to make it easier to utilise. To improve analysis, we must choose one column that may be split into two or rows that may be split, a process known as feature engineering.
3. Cleaning: Outliers in processed datasets are bound to exist and they can skew the study' results. For best results, the dataset should be cleansed. The data is completely cleansed in this step to ensure high-quality data analysis. To produce higher-quality processed data, null values should be imputed and the formatting should be consistent.
4. Enriching: After the data has been processed, it must be enhanced, which is done in the fourth phase. This means you'll need to assess what's in the data and decide whether to upscale, downsample, or enhance it. There are two methods for resampling data: one is downsampling the data and the other is upsampling the data to create synthetic data.
5. Validating: Validation is a set of repeated programming methods used to check the consistency and quality of data after it has been processed. You may need to validate data or examine whether the attributes are typically distributed to determine whether the fields in the data set are accurate.
6. Publishing: The wrangled and processed data is published for further use, which is the sole objective of data wrangling. If necessary, the entire data wrangling process should be thoroughly documented for ease of use by users and clients.

5.6.2 Tools for Data Wrangling

There are various data processing technologies that can be used to extract, import, structure, and cleanse data before feeding it into BI and analytics systems. For data

Notes

disputes, you can use automated tools to evaluate data mappings and review sample data at any stage of the translation process.

This helps identify and fix data mapping problems. When working with very large datasets, automated data cleansing becomes important. The struggle is the responsibility of the data team or data analyst for manual data cleaning tasks. Data laypeople, on the other hand, are responsible for cleaning up the data before using it in smaller setups.

Basic Data Munging Tools

- Excel Power Query / Spreadsheets — the most basic structuring tool for manual wrangling.
- OpenRefine — more sophisticated solutions, requires programming skills
- Google DataPrep - for exploration, cleaning, and preparation.
- Tabula — Swiss army knife solutions — suitable for all types of missing data
- Data Wrangler — for data cleaning and transformation.
- CSVKit — for data converting

Data Wrangling in Python

- Numpy (aka Numerical Python): It's the most fundamental data science Python package. Numpy can be used to execute operations on n-arrays and matrices in Python. It supports vectorisation of mathematical operations on the NumPy array type, which aids performance and, as a result, speeds up the execution of python code.
- Pandas: It speeds up and simplifies data analysis tasks. This is particularly useful for data structures with labelled axes. Common errors that can be derived from misaligned data during data scraping are prevented by some data alignment.
- Matplotlib: It's the most widely used visualisation module in Python. Line graphs, pie charts, histograms and other professional-grade figures can be created.
- Plotly: graphs that are interactive and publishable Line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axis, polar graphs, and bubble charts are all possible with this software.
- Theano: Numpy-like python library for numerical calculation. This toolkit is designed to quickly define, optimise, and analyse multi-dimensional array mathematical expressions.

Data Wrangling in R

- Dplyr - The R package for data munging is a must-have. The best tool for data framing. Particularly effective for category-based data management.
- Purrr - For list function operations and error checking, this is a suitable choice.
- Splitstackshape - It's an oldie, but a goodie. It's useful for simplifying the viewing of complex data sets.
- JSOnline - It's a great and simple parsing tool.
- Magrittr - it's useful for handling disjointed sets and putting them together in a more logical manner.

5.6.3 Data Cleaning

One of the most crucial aspects of machine learning is data cleaning. It is crucial in the construction of a model. It isn't the most glamorous aspect of machine learning,

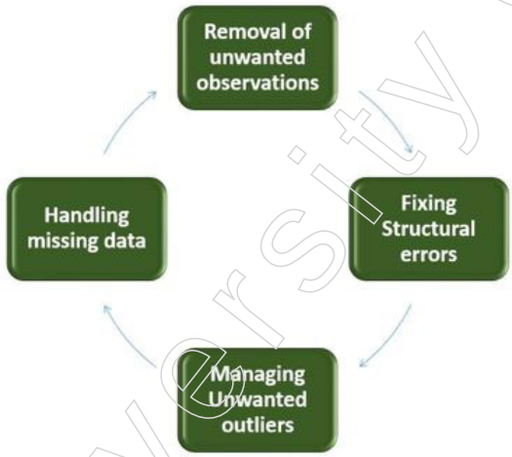
Notes

but there aren't any hidden tricks or secrets to discover. However, proper data cleaning determines whether a project succeeds or fails. Because "better data beats fancier algorithms," professional data scientists typically devote a significant amount of effort to this step.

If we start with a clean dataset, there's a strong chance we'll be able to produce good results using basic techniques, which can be quite useful in terms of computing, especially when the dataset is enormous.

Different sorts of data will, of course, necessitate different types of cleansing. This methodical approach, on the other hand, might always serve as a decent beginning point.

Steps involved in Data Cleaning:



1. Removal of unwanted observations

   This includes removing variables from your dataset that are duplicated, redundant, or irrelevant. Duplicate observations are most common during data gathering and irrelevant observations are ones that don't pertain to the problem you're attempting to answer.

   - ❖ Redundant observations reduce efficiency by a significant amount because the data repeats and may add to the correct or incorrect side, resulting in untrustworthy results.
   - ❖ Any form of data that is of no benefit to us and may be eliminated directly is referred to as irrelevant observations.

2. Fixing Structural errors

   Structural errors are errors that occur during measurement, data transfer, or other comparable situations. Typos in feature names, the same attribute with a different name, mislabelled classes, i.e., separate classes that should really be the same and inconsistent capitalisation are examples of structural errors.

   - ❖ For example, the model will treat America and America as distinct classes or values, despite the fact that they both reflect the same value, or red, yellow, and red, yellow as distinct classes or characteristics, despite the fact that one class might be contained in the other two. So, there are a few structural flaws in our model that cause it to be wasteful and produce low-quality outcomes.

3. Managing Unwanted outliers

   With some sorts of models, outliers can cause issues. Linear regression models,

Notes

for example, are less resistant to outliers than decision tree models. In general, we should not delete outliers unless there is a compelling reason to do so. Removing them increases performance in some cases, but not in others. As a result, an outlier must be removed for a valid reason, such as suspicious observations that are unlikely to be part of true data.

4. Handling missing data

In machine learning, missing data is a surprisingly difficult problem. We can't easily ignore or delete the observation that isn't there. They must be handled with caution because they may indicate something significant. The following are the two most frequent approaches to dealing with missing data:

1. Dropping observations with missing values.
   - ♦ It's possible that the fact that the value was missing was instructive in and of itself.
   - ♦ Furthermore, in the real world, you will frequently be required to generate predictions based on new data, even if some features are lacking!
2. Imputing the missing values from past observations.
   - ♦ Again, "missingness" is usually always instructive in and of itself, therefore you should alert your algorithm if a value is missing.
   - ♦ Even if you create a model to infer your values, you aren't adding any actual data. You're simply reiterating the patterns established by prior features.

Missing data is akin to missing a piece of a puzzle. Dropping it is the same as pretending the puzzle slot isn't there. If you infer it, you're attempting to fit a piece from another jigsaw component into the puzzle.

As a result, missing data is usually instructive and indicative of something significant. And, by highlighting missing data, we must be aware of our algorithm. You're essentially allowing the algorithm to estimate the ideal constant for missingness instead of just filling it in with the mean by employing this technique of flagging and filling.

Some techniques for data purification

- ❖ Openrefine
- ❖ Trifacta Wrangler
- ❖ TIBCO Clarity
- ❖ Cloudingo
- ❖ IBM Infosphere Quality Stage

5.7 Communicating Data Analysis Findings and Incomplete Section

The most crucial part of every study is data analysis. The information gathered is summarised throughout the data analysis phase. It involves applying analytical and logical reasoning to data to find patterns, correlations, and trends.

Data analysis is the systematic application of statistical and/or logical tools to explain and show, condense, and recap and appraise data. According to Shamoo and Resnik, various analytic methods "provide a means of extracting inductive inferences from data and distinguishing the signal (the phenomenon of interest) from the noise (statistical fluctuations) inherent in the data" (2003).

Notes

While statistical methods can be employed in qualitative research, data analysis is often a continuous iterative process in which data is acquired and processed almost simultaneously. Researchers do look for patterns in observations during the data collection phase (Savenye, Robinson, 2004). The type of analysis depends on the qualitative approach (field study, ethnographic content analysis, oral history, biography, unobtrusive research) and the data type (field notes, documents, audiotape, videotape).

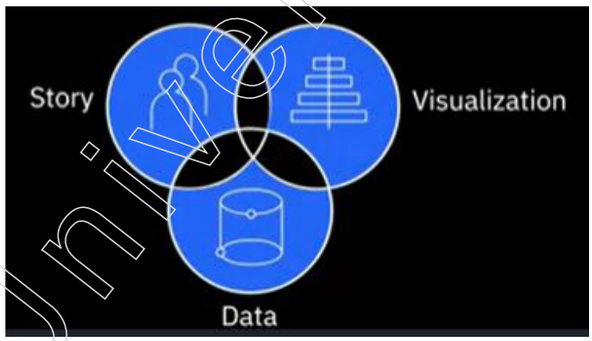
5.7.1 Overview of Communicating and Sharing Data Analysis Findings

Understanding the problem to be solved and the intended outcome to be obtained is the first step in the data analysis process. Finally, the findings must be communicated in a way that has an impact on decision-making.

Data initiatives are the consequence of a combination of factors.

- a cross-functional endeavour combining people with multi-disciplinary skills,
- with the results being incorporated into a bigger commercial initiative

The ability of others to understand and trust our insights to take action is critical to the success of communication. As a result, as data analysts, we must tell a story with our data by clearly illustrating the insights and crafting a structured narrative that is specifically aimed to our target audience.



We need to reconnect with our audience before we start creating the communication. Begin by posing the following questions to ourselves:

- ❖ Who is our intended audience?
- ❖ What is crucial to them?
- ❖ What will persuade people to believe in me?

Our target audience will be diverse in terms of the business activities they represent, whether they play an operational or strategic role in the organisation, how they are affected by the problem and other considerations. Our presentation should be tailored to the amount of knowledge that our audience already possesses. We'll decide what information and how much of it, is necessary to better comprehend our findings based on our understanding of the audience.

It's tempting to show off all of the data we've been working with, but we need to think about which parts are more valuable to our audience.

A presentation isn't the same as a data dump. Facts and data alone do not sway people's opinions or motivate them to act. We need to tell an engaging story. Only include the information that is required to solve the business challenge. Our audience will struggle to understand our message if we provide too much detail.

Notes

We must begin our presentation by demonstrating to our audience that we understand the business challenge. It's easy to assume that we all know what we're here for, but expressing our awareness of the situation at hand and the desired goal is a terrific way to get their attention and build trust. Another crucial component in establishing a connection between us and our audience is speaking in the language of the organisation's business domain.

The next phase in communication design is to arrange and organise our presentation so that it has the most impact. Refer to the information we've gathered. Remember that the audience sees the data, which is at the heart of everything we're presenting, as a black box. People are unsure whether or not they can believe our findings if the legitimacy of our data cannot be established. Data sources, theories and validations should all be shared. Along the way, work to establish the trustworthiness of our conclusions - don't overlook any critical assumptions established during the study.

Based on the facts we have, organise information into logical categories.

- Do you, for example, have both qualitative and quantitative data?

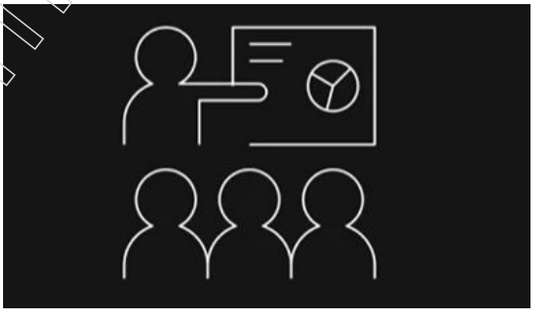
Consider if we should tell our story from the top down or from the bottom up. Depending on the audience and use scenario, either can be effective. In order to be successful, we must be constant in our approach. It's critical to figure out which communication forms will be most beneficial to our target audience. Is it necessary for them to take an executive summary, a fact sheet, or a report with them? The forms we use should be determined by how our audience will utilise the information we've delivered. Insights must be communicated in a way that motivates others to take action. If our audience doesn't understand the importance of our insight or isn't convinced of its utility, the insight will be useless.

A thousand-word article will not have the same impact on the audience as a graphic in terms of building a clear mental image. A compelling visualisation uses graphical representations of facts and data to tell a story. Data visualisations, such as graphs, charts, and diagrams, are a terrific way to make data come to life. We have tools that can assist us exhibit patterns and conclusions concerning hypotheses, whether it's a comparison, a relationship, a distribution, or a composition.

Data is valuable because of the stories it tells. Our target audience must be able to trust us, comprehend what we're saying and relate to our findings and conclusions.

5.7.2 Viewpoints: Storytelling in Data Analysis

We'll examine what data professionals have to say about the role storytelling plays in data analysis and the life of a data analyst.



The importance of storytelling in the life of a data analyst cannot be emphasised. It's vital to improve your data storytelling skills. Humans, we believe, instinctively

Notes

comprehend the world through tales. If you want to persuade someone to do something using data, the first step is to deliver a clear, short, and compelling story. One also believes that developing a story whenever a data analyst is working with a dataset can assist them better comprehend the underlying dataset and what it is doing.

There will always be a compromise between providing a clear, consistent, and straightforward tale and ensuring that you explain all of the complexity found in the data. Finding that balance, we believe, might be difficult, but it is vital.

In the life of a data analyst, the skill of storytelling is crucial. It doesn't matter how much or how interesting information you've gathered. It's all for naught if you can't find a means to communicate that to your audience, whether it's a customer or a director or executive level individual. You must find a means to explain this and it is usually better to do it in a visual or by telling a story so that they can see how the information might be valuable.

Organisations believe that storytelling is a necessary skill set. It's similar to the final mile of a delivery. With a little training, a lot of people can manage the technical side. The capacity to extract value from data and communicate it, on the other hand, is in limited supply. If you're thinking about a long-term career, we believe knowing how to tell a captivating story using statistics is essential.

The importance of storytelling in data and analytics cannot be overstated. This is how your message is actually delivered. Everyone can exhibit figures, but if there isn't a story behind them, if there isn't a compelling reason to act, then what you're providing will ultimately fall flat with your audience.

For example: People were asked to present their pitches at Stanford and the pitches contained only KPIs, figures and data, but they also told the tale. After the presentations, the audience members were asked what they recalled from each of the presentations and it was those stories that stayed with them.

5.7.3 Introduction to Data Visualisation

Data visualisation is one of the interactive methods for generating new ideas and discoveries. It's a dynamic instrument that opens up new study avenues and speeds up the scientific process. Every day, a great amount of data is generated due to widespread usage of the Internet and Web. There is a requirement to comprehend vast and complex data sets. When a huge amount of data is accessible, it must be processed using various data processing methods and presented using various techniques and methodologies.

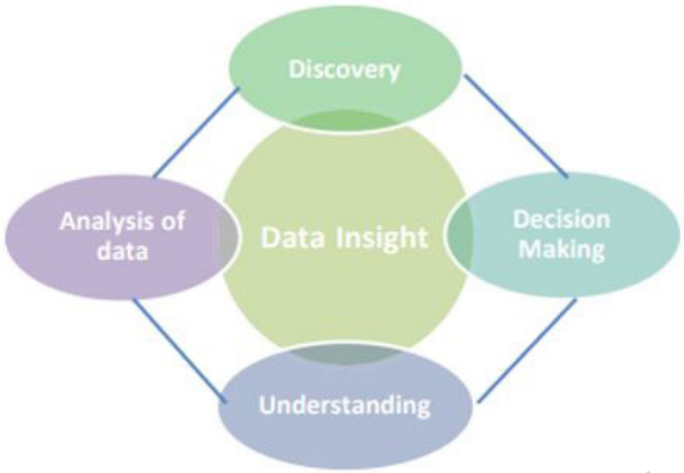
Data visualisation is critical to every company's success because it allows them to effectively manage their data and make the greatest use of it in order to turn it into knowledge. It is the process of transforming data and figures into a visual representation.

Different computer graphics effects are used in data visualisation techniques. It assists stakeholders in making quick and effective decisions. It also allows for a better knowledge of pattern identification, trend analysis and extracting relevant information from pictures.

The world is being increasingly flooded with data on a daily basis, necessitating the need to handle and show data in an understandable manner. Visualisation is a tool for analysing data in a variety of ways in order to make better decisions. The practise of presenting data and information graphically or pictorially is known as data visualisation. It is quickly establishing itself as a powerful, widely applicable, and acceptable tool for

Notes

reading and analysing huge, complicated data sets. It's become a simple and quick way to communicate facts and ideas in a global manner.



Data visualisation is concerned with the creation, design, and implementation of graphical representations of data that make data easier to comprehend. It's often referred to as information visualisation or scientific visualisation.

For millennia, people have utilised visuals, graphs, charts, and maps to grasp facts and information. Computer advancements have made it possible to handle and process large amounts of data at a quick rate. Today, data visualisation is evolving into a fusion of art and science that will have a noticeable impact in the next years.

Although visualising data can be difficult, it is far easier to interpret data in this format than it is in text, numbers, or big tables with many rows and columns. Understanding data and its composition might help one choose the right data visualisation technique. The goal of all visualisation techniques is to address the same problem in a different way.

Data visualisation can be divided into two types, each with a different purpose: explanation and exploration. When there is a lot of data but little information about it and the goals are imprecise, exploration data visualisation is effective. Explanatory data visualisation is used when a large amount of data is accessible, but we don't know what it is. Both of these categories aid in the visual display of facts.

Importance of Visualisation

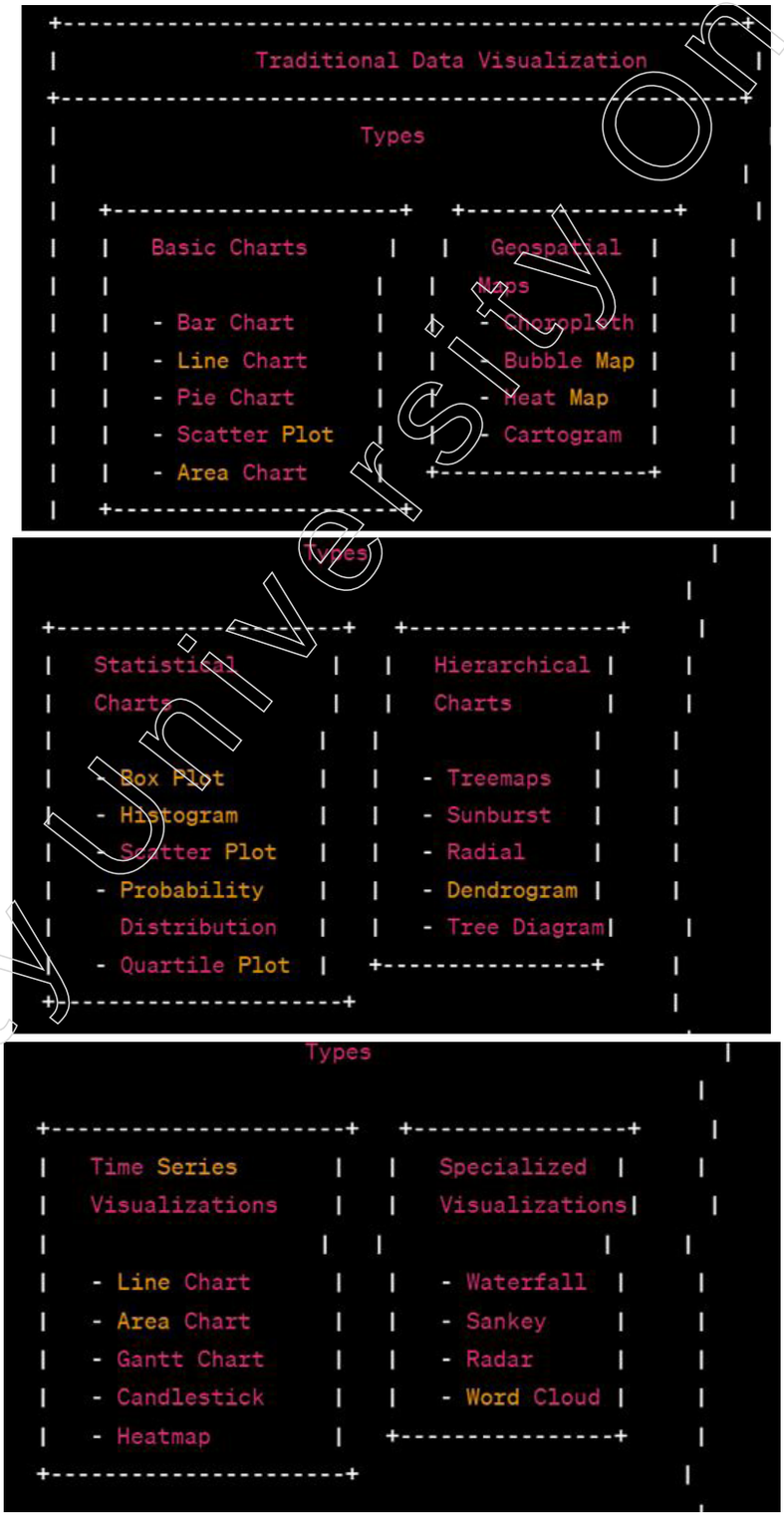
Every day, a great amount of data is generated due to widespread usage of the Internet and Web. There is a requirement to comprehend vast and complex data sets. Every organisation that retains records must deal with data and make decisions. When a huge amount of data is accessible, it must be processed using various data processing methods and presented using various techniques and methodologies.

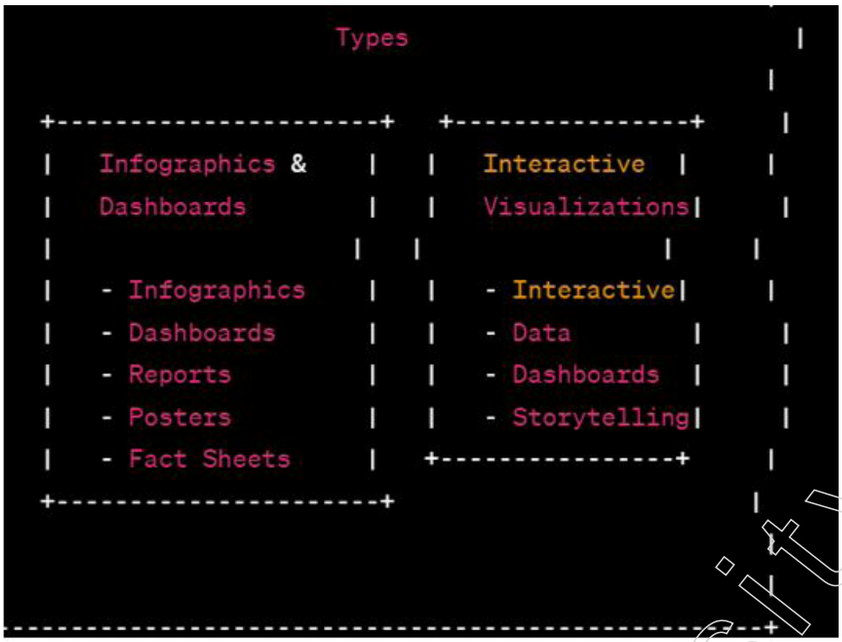
It is critical to the success of any business because it enables them to effectively manage data and make the greatest use of that data in order to convert it into knowledge. It is the process of transforming data and figures into a visual representation. Different computer graphics effects are used in data visualisation techniques. Data visualisation improves learning, comprehension, and reasoning, as well as assisting stakeholders in making quick and effective decisions. It also allows for a better knowledge of pattern identification, trend analysis and extracting relevant information from pictures.

Notes

Data visualisation truly aids in the accurate, clear, and efficient communication of complicated data. It actually absorbs data in new and more constructive ways, allowing businesses to make more informed and valuable decisions. It depicts the relationships and patterns that exist between operational activities. To put it another way, data visualisation is a new business language.

Traditional Data Visualisation Techniques





This diagram outlines several categories of traditional data visualization techniques, including basic charts, statistical charts, time series visualizations, and specialized visualizations like infographics and interactive dashboards. Each category contains specific types of visualizations suited for different data presentation needs.

There are a variety of tools and approaches that are used to convert data into a visual form that cannot be changed directly by a human. Microsoft Word, Microsoft Excel, Microsoft Spreadsheet and Microsoft PowerPoint are some of the most popular multifunctional applications with database connectivity that may be used to visualise data and provide excellent outcomes. Organisations that do not require highly specialised data visualisation can benefit from these software. Pie chart, line chart, bar chart, area chart, graphs, map, heat map and other traditional data visualisation techniques are used to depict data.

5.7.4 Introduction to Visualisation and Dashboarding Software

In this data-driven world, it's critical to comprehend the data in order to gain actionable insights. And data visualisation is a critical component of comprehending the data's underlying patterns and layers! Which of these options appeals to you the most? A visually appealing and informative bar chart or a dull spreadsheet with the same data?

Humans are visual animals, thus data visualisation charts such as bar charts, scatterplots, line charts, geographical maps and so on are essential. They provide information just by looking at them, whereas you would ordinarily need to read spreadsheets or text reports to comprehend the data. Data visualisation tools are popular because they make it simple for analysts and statisticians to develop visual data models based on their specifications by combining an interface, database connectivity and Machine Learning technologies in one accessible location.

Tableau

Tableau is a data visualisation application that allows data analysts, scientists, statisticians, and other professionals to analyse data and form clear conclusions based on their findings. Tableau is well-known for its ability to quickly process data and provide

Notes

the desired data visualisation output. And it may do so while maintaining the highest level of security, with the assurance that security vulnerabilities will be addressed as soon as they happen or are discovered by users.

Tableau also allows users to prepare, clean and format their data before creating data visualisations that can be shared with other Tableau users. Tableau is a data visualisation tool that may be used by a single data analyst or by entire business teams and companies. It offers a 14-day free trial before moving on to the premium version.

Microsoft Excel

- Microsoft Excel is a data visualisation tool with a user-friendly design; thus, it isn't difficult to use.
- In Excel, you may visualise data in a variety of ways. One of these is to use scatter plots, which show the relationship between two datasets that you want to compare. You may also observe how different variables are related to one another to determine whether they are connected.
- For market research or financial planning, many data analysts utilise scatter plots to examine statistical, scientific, medical, and economic data.

Microsoft Power BI

Microsoft Power BI is a data visualisation technology that aims to instil a data-driven business intelligence culture in today's businesses. To that end, it provides self-service analytics tools for analysing, aggregating, and sharing data in a meaningful way.

Customers may choose from hundreds of data visualisations, as well as built-in Artificial Intelligence capabilities and Excel interface options, using Microsoft Power BI. All of this comes at a very reasonable monthly cost of $9.99 per user for Microsoft Power BI Pro. It also offers a variety of assistance options, including FAQs, forums, and live chat with the staff.

Looker

Looker is a data visualisation tool that can go into the data and analyse it to extract helpful insights. It delivers real-time data dashboards for in-depth research, allowing businesses to make quick decisions based on the data visualisations. Looker also connects to Redshift, Snowflake, BigQuery and more than 50 SQL dialects, allowing you to connect to numerous databases without difficulty.

Data visualisations created using Looker can be shared with anyone using any tool. You can also export these files in any format right away. It also offers customer service, which allows you to ask any query and get it answered. A pricing quote can be requested by filling out a form.

Sisense

Sisense is a data visualisation system based on business intelligence that offers a variety of tools to help data analysts simplify difficult data and get insights for their organisation and outsiders. Sisense thinks that, in the end, every business will be data-driven and every product will be data-related in some way. As a result, it makes every effort to deliver various data analytics tools to business teams and data analysts so that they may assist in transforming their organisations into data-driven enterprises of the future.

Sisense is very simple to set up and use. It takes less than a minute to set up and data analysts can get their work done and get results almost immediately. Sisense also

Notes

allows users to export their files in a variety of formats, including PowerPoint, Excel, MS Word, PDF, and others. Sisense also offers full-time customer support if users have any problems. A pricing quote can be requested by filling out a form.

SAP Analytics Cloud

SAP Analytics Cloud combines business intelligence and data analytics to assist you analyse your data and visualise it to forecast business consequences. It also gives you access to the most up-to-date modelling tools, which assist you by alerting you to potential data problems and categorising various data metrics and dimensions. SAP Analytics Cloud also suggests data Smart Transformations that result in improved visuals.

If you have any issues or business questions about data visualisation, SAP Analytics Cloud can answer them using conversational artificial intelligence and natural language technologies, ensuring complete customer satisfaction. This platform is free for the first 30 days, after which you must pay $22 per month for the Business Intelligence package.

Python Libraries

The Python programming language comes with a variety of libraries that can be used for a variety of tasks. Similarly, Python provides a number of packages for data visualisation, allowing users to explore and analyse datasets in great detail.

Each visualisation library has its own set of requirements. Using specific libraries for various tasks makes it easier and more accurate for the user to execute the task. Some liberators are more effective than others.

The following are libraries for data visualisation in Python programming:

- Matplotlib
- Ggplot
- Pygal
- Missingno
- Seaborn
- Plotly
- Gleam
- Leather
- Geoplotlib
- Bokeh
- Folium

Plotly

Plotly's connection with analytics-oriented programming languages like Python, R and Matlab allows for more elaborate and sophisticated visualisations. It's based on the open source d3.js JavaScript visualisation libraries, but this commercial package (which also comes with a free non-commercial licence) adds layers of user-friendliness and support, as well as built-in support for APIs like Salesforce.

Qlikview

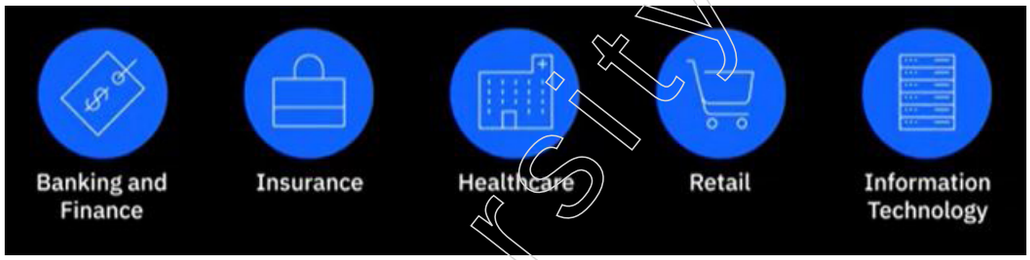
The second significant player in this field and Tableau's largest competitor, is Qlik, with their Qlikview product. The vendor has over 40,000 customer accounts in over 100 countries and those who use it frequently point to its highly flexible configuration

Notes

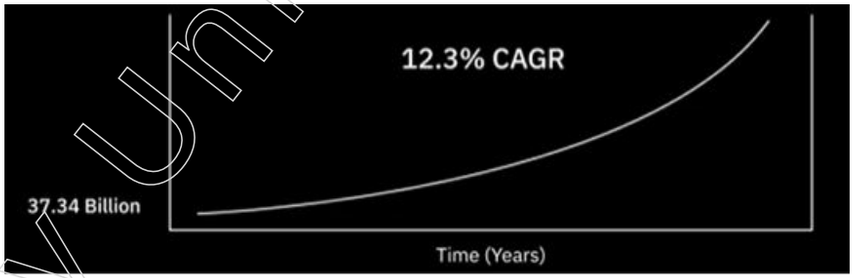
and extensive feature set as a major benefit. This, on the other hand, may imply that learning to use it to its greatest capacity takes longer. Qlikview delivers extensive business intelligence, analytics, and enterprise reporting capabilities in addition to data visualisation and one particularly likes the clear and clutter-free user interface. Qlikview is frequently used in conjunction with Qliksense, a sister programme that handles data exploration and discovery. There is also a vibrant community and there are numerous third-party materials accessible online to assist new users in learning how to use it in their projects.

5.8 Overview to Career Opportunities in Data Analysis

Job openings for data analysts can be found in industry, government, and academia. Skilled data analysts are needed in every business, including banking and finance, insurance, healthcare, retail, and information technology. These positions are in high demand in both large corporations and small firms in San Francisco.



According to Forbes, the global big data analytics industry, which was worth 37.34 billion dollars in 2018, is predicted to increase at a compound annual growth rate of 12.3% from 2019 to 2027, reaching 105.08 billion dollars. Currently, the demand for talented data analysts outnumbers the supply, implying that organisations are ready to pay a premium to hire them.



Data analysts must be able to understand a wide range of work tasks. The roles in the professional path can be broadly classified as follows:

- ❖ data analyst specialist roles
- ❖ domain specialist roles.

Data analytics is more crucial than ever in this digital age. There are numerous job openings in a variety of industries and the demand for data analytics professionals is growing by the day. The following are some of the job opportunities that demand data analytics professionals:

1. Data Scientist

A data scientist gathers and analyses data in order to make informed judgments utilising data visualisation. A data scientist must have a broad understanding of data, strong data analytics and data visualisation skills and knowledge of programming languages such as SQL, Python, Scala, and others.

Notes

A Data Scientist earns an average of Rs 634,645 a year.

2. Data Engineer

A data engineer aids in the development, deployment and optimisation of the data architecture that supports various data analytics operations. In general, a data engineer works with big data sets and frequently assists data scientists in making this data understandable through data cleansing and profiling.

A Data Engineer earns an average of Rs 809,923 a year.

3. Business Analyst

A business analyst uses data analytics to comprehend business models, corporate reports, technology integration documentation and other business methods to help an organisation solve its business challenges.

A Business Analyst's annual compensation averages Rs 588,313 in India.

4. Statistician

A statistician gathers, analyses, and interprets statistical data in order to generate meaningful and cohesive information. Statistical simulations, mathematical modelling, analysis and interpretation of various survey results, business forecasts based on data analytics and so on are some of the common duties of statisticians.

A statistician's average annual pay is Rs 475,370.

5. Machine Learning Engineer

A machine learning engineer uses data analytics to analyse and evaluate algorithms and statistical models for machine learning. A machine learning engineer must have a strong understanding of both programming and statistics.

A Machine Learning Engineer earns an average of Rs 705,035 a year.

6. Quantitative Analyst

A quantitative analyst uses data analytics to evaluate vast amounts of data in order to comprehend financial risk management, investment patterns, exchange rate trends, the stock market, and other financial issues.

A Quantitative Analyst earns an average of Rs 842,849 a year.

These are just a few examples of jobs that involve data analytics. However, data analytics as a field is broad and the options it offers are limitless. In the subject of data analytics, there are several job options and opportunities, with even more growth projected in the future. As a result, a career in data analytics is a profitable one with a lot of room for advancement.

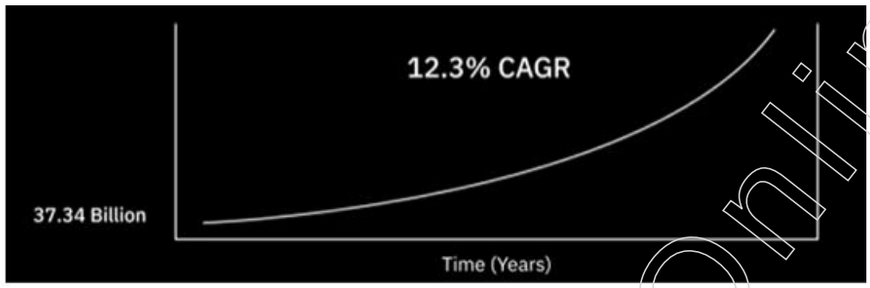
5.8.1 Career Opportunities in Data Analysis

Job openings for data analysts can be found in industry, government, and academia. Skilled data analysts are needed in every business, including banking and finance, insurance, healthcare, retail, and information technology. These positions are in high demand in both large corporations and small firms in San Francisco.

According to Forbes, the global big data analytics industry, which was worth 37.34 billion dollars in 2018, is predicted to increase at a compound annual growth rate of 12.3% from 2019 to 2027, reaching 105.08 billion dollars. Currently, the demand for

Notes

talented data analysts outnumbers the supply, implying that organisations are ready to pay a premium to hire them.



Data analysts must be able to understand a wide range of work tasks. The roles in the professional path can be broadly classified as follows:

- ❖ data analyst specialist roles
- ❖ domain specialist roles.

Data analytics is more crucial than ever in this digital age. There are numerous job openings in a variety of industries and the demand for data analytics professionals is growing by the day. The following are some of the job opportunities that demand data analytics professionals:

1. Data Scientist

   A data scientist gathers and analyses data in order to make informed judgments utilising data visualisation. A data scientist must have a broad understanding of data, strong data analytics and data visualisation skills and knowledge of programming languages such as SQL, Python, Scala, and others.

   A Data Scientist earns an average of Rs 634,645 a year.

2. Data Engineer

   A data engineer aids in the development, deployment and optimisation of the data architecture that supports various data analytics operations. In general, a data engineer works with big data sets and frequently assists data scientists in making this data understandable through data cleansing and profiling.

   A Data Engineer earns an average of Rs 809,923 a year.

3. Business Analyst

   A business analyst uses data analytics to comprehend business models, corporate reports, technology integration documentation and other business methods to help an organisation solve its business challenges.

   A Business Analyst's annual compensation averages Rs 588,313 in India.

4. Statistician

   A statistician gathers, analyses, and interprets statistical data in order to generate meaningful and cohesive information. Statistical simulations, mathematical modelling, analysis and interpretation of various survey results, business forecasts based on data analytics and so on are some of the common duties of statisticians.

   A statistician's average annual pay is Rs 475,370.

6. Machine Learning Engineer

   A machine learning engineer uses data analytics to analyse and evaluate algorithms

Notes

and statistical models for machine learning. A machine learning engineer must have a strong understanding of both programming and statistics.

A Machine Learning Engineer earns an average of Rs 705,035 a year.

7. Quantitative Analyst

A quantitative analyst uses data analytics to evaluate vast amounts of data in order to comprehend financial risk management, investment patterns, exchange rate trends, the stock market, and other financial issues.

A Quantitative Analyst earns an average of Rs 842,849 a year.

These are just a few examples of jobs that involve data analytics. However, data analytics as a field is broad and the options it offers are limitless. In the subject of data analytics, there are several job options and opportunities, with even more growth projected in the future. As a result, a career in data analytics is a profitable one with a lot of room for advancement.

5.8.2 Viewpoints: Get into Data Profession

We'll look at what data experts have to say about how they got into this field.

Barnes, Asha

"Before I took the job, my current career as a data specialist didn't exist." I saw that there was a demand in our firm for data to be delivered in a faster, more efficient manner than going to the IS department, which would hold a meeting to discuss the meeting, gather requests and then deliver an end product that people didn't like. However, you had to wait until the end of the line and then repeat the process to acquire what you wanted. I built a company database with access to more information by addressing a demand at the company to give reports in two weeks. We now have analysts who are able to fill that gap in the company."

Vidisha Vachharajani, Vidisha Vachharajani, Vidisha Vachharajan

"I stumbled into the role of data professional by chance." I was working on my PhD in Economics at the University of Illinois, Urbana-Champaign, when a colleague recommended that a master's degree in statistics would be a great addition. That's how I got into the Illinois statistics programme as well. But once I got started, I was really hooked and there was no turning back. To put it another way, my original ambition of being an economist has turned into a job involving data, modelling, analytics, insight collecting, communication, visualisation and, of course, data-driven problem-solving."

Erin Huang is a student at the University of Michigan.

"I stumbled into a data analyst position in a financial data company." My company began hiring equities data analysts in [inaudible], China, at the time and I was extremely fortunate to be hired because they were seeking for someone with financial analytical skills, which I can bring to the table. Following that, my team began looking for someone with technical skills such as Python, R and Sickle.

"I've always loved numbers," Joye Sistrunk says. When you deal with numbers for a long time, they begin to tell a story and the capacity to look at those numbers and articulate that story is what appeals to me. Having always been drawn to numbers, whether it's Excel spreadsheets, QuickBooks, or any other datasets that can help us drive the information we're looking for, especially in the financial industry where we're looking at profit, loss, and balance sheet, as well as what happens when one company

Notes

buys another. We're continually looking at that data for people to talk to and discuss the company's past and future."

5.8.3 Viewpoints: What do Employers look for in a Data Analyst?

Employers seek Data Analysts who are trustworthy. During the hiring process, they may ask if you would prefer to fulfil a deadline or obtain the proper answer if you only had one choice. They're always on the lookout for someone who can say, "I want to double-check that the information is correct." Missing a deadline isn't as bad as a corporation making a multimillion-dollar choice based on inaccurate data, or someone losing their job because the data wasn't pulled or reported accurately. Integrity is far more important than anything else.

The ability to communicate well is the most important quality that employers look for in Data Analysts. You can undertake the most brilliant analysis in the world, but if you can't communicate it to external stakeholders, it's useless. The ability is in high demand. Fluency with numbers, the capacity to understand sophisticated analyses and the ability to understand AB tests and what the findings of AB tests mean, as well as the implications of those results, are all things that organisations look for when hiring a Data Analyst.

Employers are increasingly looking for Data Analysts with exceptional SQL expertise. Employers are also searching for a growth attitude and a willingness to learn in Data Analysts, as the field is changing at a rapid rate. They're searching for someone with programming expertise, such as Python, R and SQL. Simultaneously, they're on the lookout for some personalities. Whether you are detail-oriented, enjoy working with data, are a problem solver and so on.

They hire people on a regular basis as employers. So, what exactly are they looking for? They are seeking for detail-oriented employees who are a little bit overachievers. They don't want to stop at doing what's in front of them; they want to go even further. They are searching for individuals with great objectives and the ability to think beyond the box. If they tell them to do ABC, they won't simply do it; they'll go above and beyond and offer some options.

People who can troubleshoot problems. It's not only that you're detail-oriented or that you're brilliant with numbers. However, you must be able to think outside the box, as well as solve problems and troubleshoot issues. Employers will be looking for this now more than ever before. They're looking for the ability to understand data, which can signify a variety of things. Be able to think about it and be comfortable with it in multiple formats. That is to say, know what facts you require to solve the problems at hand. It is critical to have a strong understanding of data.

Another important talent is problem-solving. Meaning, if a Data Analyst is confronted with an issue, they should be able to figure out how to fix it by analysing data in whatever format it is in, analysing it and presenting the insights that will solve the problem. They must also be very dynamic in the sense that if they are suddenly confronted with a totally different data set that looks nothing like it did before, they must be able to adjust. As a result, the ability to be dynamic and adaptable is essential.

They must also be able to swiftly learn new technological abilities. That is to say, if one SQL DIAdem is used in one scenario, they must be able to work under a different paradigm in another. If there's a location that uses RStudio but knows Python, they'll need to be able to rapidly take up RStudio and that. Employers look for a few qualities in a competent Data Analyst: the ability to learn quickly, adaptability and data knowledge.

Notes

5.8.4 The Many Paths to Data Analysis

When we were kids, data science didn't really exist. We didn't wake up one day and declare, "When I grow up, I want to be a data scientist." It didn't exist, to be sure. We had no idea we'd end up working in data science. There was no such thing as data science while we were growing up. And we believe it is brand new. It wasn't until 2009, 2011 that data science became a reality. The word was coined by DJ Patil or Andrew Gelman. There was a time when statistics were used. And none of us wanted to be one of them. We want to start a company.

Then we discovered data science, which turned out to be a lot more exciting. That's how we got started: we studied statistics. We wanted to be a singer and then a doctor at various points in my life. Then we realised we were mathematicians. As a result, we focused on a quantitative analytical field. And we decided that we wanted to deal with data after that.

Data science as we know it today isn't necessarily the case. When we were in first year of mechanical engineering, we had our first exposure with data science. Data science is used by strategic consulting businesses to make judgments. As a result, it was our first experience with data science.

We had a difficult challenge to address and the techniques we had at the time weren't up to the task. We got our math degrees at the worst possible time, shortly after the financial crisis, when you had to be helpful to obtain a job. As a result, we decided to pursue a degree in statistics. After that, we did enough jobs with the title of data scientist that we became one.

There are several routes into the field of data analysis that you might take. While some organisations may require an academic degree as a prerequisite, even if you don't have one, you still have various possibilities that can help you get into the field of data analysis, or even make a lateral transfer. Let's begin with the most straightforward path.

A bachelor's degree in data Analytics, Statistics, Computer Science, Management Information Systems, or Information Technology Management will give you a leg up on the competition. Alternatively, you could enrol in online training classes to gain the necessary information.

Multi-course specialisations in data analysis are available through online learning platforms such as Coursera, edX and Udacity. Some of the world's greatest domain specialists devised and conducted these courses. Because you should have a good concept of the technical, functional, and soft skills you'll need to be a data analyst by now, picking the proper learning path should be simple.

You can continue to advance your knowledge and skills in specific areas as you gain additional work experience, such as statistics, spreadsheets, SQL, Python, data visualisation, problem-solving, storytelling, or making effective presentations. These courses also provide you with hands-on exercises and projects to help you get a feel for how your knowledge and abilities are applied in the real world. These work can even be added to your portfolio.

So, if you don't have an academic degree, these courses can help you acquire entry-level positions and work your way up as your experience grows. Let's consider a scenario in which you have a few years of experience in a different profession and wish to make the jump to data analysis. If you plan effectively, there's a decent chance you'll be able to complete it successfully.

Because data analysis is such a broad profession, you should first examine the information and abilities you'll need, as well as the numerous employment chances and

Notes

growth opportunities accessible on the path you're pursuing. To interact with people in this industry and obtain insights into real-world issues, you can use online resources, forums and your network of friends and colleagues.

If you're currently employed in a non-technical position, you might want to explore pursuing a career as a Domain Specialist or Functional Analyst. If you're in sales, you might want to start by positioning and skilling yourself for a career as a Sales Analyst. You start with industry experience and develop skills in other areas such as statistics and programming, for example. If you're currently employed in a technical position, you'll be able to quickly learn the tools and software required for the data analyst position.

You're also likely entering in with the benefit of a thorough understanding of the subject or sector from which you come. You might already be using some of the other abilities, such as problem-solving, project management, communication, and storytelling, in your current employment. Trainings, online courses, communities of practise and forums can all help you improve your talents. The discipline of data analysis is a fast-paced one. Regardless of official qualifications, you will be able to carve a way forward if you are curious, open to learning new things and enthusiastic about the area.

Summary

- Exploratory Data Analysis (EDA) is a crucial initial step in the data analysis process that involves visually and statistically examining data to gain insights, detect patterns and identify relationships between variables.
- It helps analysts understand the underlying structure and characteristics of the dataset, which is essential for making informed data-driven decisions and guiding further analysis.
- Descriptive Statistics: Summary statistics like mean, median, standard deviation and quartiles are calculated to understand the central tendencies and variability of the data.
- Data Visualisation: Visual representations, such as histograms, scatter plots, box plots and bar charts, are created to reveal patterns, distributions, and outliers in the data.
- Correlation Analysis: Correlation coefficients are calculated to measure the strength and direction of relationships between variables.
- Data Profiling: Examining data quality by identifying missing values, duplicate records, and outliers.
- Data Clustering: Grouping similar data points together to identify inherent structures in the dataset.
- By visualising and summarising data during EDA, analysts can make informed decisions on how to proceed with data cleaning, preprocessing, and modelling.
- Exploratory Data Analysis facilitates effective communication of data insights, supports hypothesis generation, and provides a solid foundation for more sophisticated analyses like regression, classification, or predictive modelling.
- Ultimately, EDA helps analysts gain a deeper understanding of their data, leading to more accurate and meaningful results in the subsequent stages of the data analysis process.

Glossary

- Exploratory Data Analysis (EDA): The process of visually and statistically examining data to gain insights, detect patterns and identify relationships between variables in the initial stages of data analysis.

Notes

- Descriptive Statistics: Summary statistics that provide a concise representation of data, including measures of central tendency (e.g., mean, median) and measures of variability (e.g., standard deviation, range).
- Data Visualisation: The use of graphical representations, such as histograms, scatter plots, box plots and bar charts, to visually explore and present patterns and distributions in the data.
- Histogram: A graphical representation of the frequency distribution of continuous data, showing the number of data points falling into predefined intervals or bins.
- Scatter Plot: A two-dimensional plot that displays the relationship between two continuous variables, each represented by a point on the graph.
- Box Plot (Box-and-Whisker Plot): A graphical representation that shows the distribution of data through quartiles, including median, minimum, maximum and outliers.
- Bar Chart: A graphical representation of categorical data using rectangular bars of varying heights or lengths to represent the frequency or proportion of each category.
- Correlation: A statistical measure that indicates the strength and direction of the linear relationship between two continuous variables.
- Data Profiling: The process of examining and summarising the quality and structure of the dataset, including identifying missing values, duplicates, and outliers.
- Outliers: Data points that significantly differ from the rest of the data, potentially indicating errors, anomalies, or interesting observations.
- Data Distribution: The way data is spread or distributed across different values, showing patterns such as normal, skewed, or bimodal distributions.
- Data Clustering: The process of grouping similar data points together based on their similarities or distances to identify underlying structures in the dataset.
- Heatmap: A graphical representation of data where individual values are represented as colours on a grid, allowing visualisation of patterns and correlations.
- Exploratory Data Visualisation: The use of interactive visualisations and dashboards to dynamically explore and analyse data, enabling deeper insights and patterns discovery.
- Cross-Tabulation (Crosstab): A tabular representation of data that shows the relationship between two categorical variables, displaying frequency counts and percentages.
- Data Density Plot: A graphical representation that displays the density of data points across continuous variables, useful for identifying clusters and data concentration.
- Data Skewness: A measure of the asymmetry of a distribution, indicating whether data is skewed to the left (negatively skewed) or to the right (positively skewed).
- Data Transformation: Preprocessing techniques applied to data, such as normalisation, standardisation, or logarithmic scaling, to improve its distribution and make it more suitable for analysis.
- Data Imputation: The process of filling missing values in the dataset using various statistical or machine learning methods.
- Data Segmentation: Dividing data into meaningful subsets based on certain criteria to perform separate analyses or comparisons.

Notes

Check Your Understanding

1. What does EDA stand for in data analysis?
   a) Exploratory Data Aggregation
   b) External Data Analysis
   c) Exploratory Data Analysis
   d) Extract Data Aggregation
2. What is the primary purpose of Exploratory Data Analysis?
   a) To clean and preprocess data
   b) To build predictive models
   c) To extract insights and patterns from data
   d) To validate data analysis results
3. Which of the following is NOT a component of EDA?
   a) Data Visualisation
   b) Data Cleaning
   c) Data Preprocessing
   d) Data Modelling
4. What type of statistics provides a concise representation of data, including measures of central tendency and variability?
   a) Inferential Statistics
   b) Descriptive Statistics
   c) Hypothesis Testing
   d) Predictive Statistics
5. Which visualisation is used to display the distribution of continuous data through quartiles and outliers?
   a) Box Plot
   b) Histogram
   c) Scatter Plot
   d) Bar Chart
6. What is the graphical representation of the frequency distribution of continuous data called?
   a) Heatmap
   b) Bar Chart
   c) Histogram
   d) Scatter Plot
7. What does correlation measure in Exploratory Data Analysis?
   a) Causation between variables
   b) The distribution of data in a dataset
   c) The strength and direction of the linear relationship between continuous variables
   d) The strength of the relationship between categorical variables
8. What is the purpose of data profiling in EDA?
   a) Identifying patterns in data
   b) Calculating summary statistics
   c) Discovering relationships between variables
   d) Examining data quality and structure
9. What is the process of grouping similar data points together based on their similarities called?
   a) Data Clustering
   b) Data Aggregation
   c) Data Segmentation
   d) Data Classification
10. What is the primary goal of data transformation in EDA?
    a) To identify outliers
    b) To visualise data in graphs
    c) To improve data distribution and make it suitable for analysis

Notes

d) To calculate summary statistics

11. Which type of plot displays the density of data points across continuous variables?
    a) Scatter Plot b) Bar Chart
    c) Box Plot d) Data Density Plot
12. Which type of data skewness indicates a longer tail to the left of the data distribution?
    a) Positive Skewness b) Negative Skewness
    c) Symmetrical Skewness d) No Skewness
13. What is the process of filling missing values in a dataset called?
    a) Data Imputation b) Data Interpolation
    c) Data Preprocessing d) Data Transformation
14. What visualisation technique shows the relationship between two continuous variables?
    a) Bar Chart b) Histogram
    c) Box Plot d) Scatter Plot
15. Which EDA technique involves dividing data into meaningful subsets based on certain criteria?
    a) Data Clustering b) Data Segmentation
    c) Data Aggregation d) Data Preprocessing
16. Which plot is used to display the frequency or proportion of each category in categorical data?
    a) Scatter Plot b) Histogram
    c) Bar Chart d) Box Plot
17. What visualisation technique allows the examination of the relationship between two categorical variables?
    a) Scatter Plot b) Box Plot
    c) Heatmap d) Crosstab
18. What is the process of identifying and correcting errors, inconsistencies, and missing values in data?
    a) Data Cleaning b) Data Preprocessing
    c) Data Visualisation d) Data Aggregation
19. What is the primary purpose of Exploratory Data Visualisation?
    a) To preprocess and transform data
    b) To visually explore data and discover patterns
    c) To build predictive models d) To validate data analysis results
20. What visualisation technique shows the distribution of data through quartiles, including the median, minimum, maximum and outliers?
    a) Scatter Plot b) Histogram
    c) Box Plot d) Bar Chart

Notes

Exercise

1. What is Data Analytics?
2. Explain Modern Data Ecosystem.
3. What is Key Players in the Data Ecosystem?
4. Explain The Data Analyst Role.
5. Describe Responsibilities of a Data Analyst?
6. Explain The Data Ecosystem and Languages for Data Professionals.
7. Describe Understanding Data Repositories and Big Data Platforms.
8. What is Gathering Data?
9. Explain Data Wrangling.
10. What is Communicating Data Analysis Findings and Incomplete Section ?

Learning Activities

1. Discuss The Data Ecosystem and Languages for Data Professionals with an example.
2. Describe Exploratory Data Analysis of any company.

Check Your Understanding- Answers

1. c	2. c	3. d	4. b
5. a	6. c	7. c	8. d
9. a	10. c	11. d	12. b
13. a	14. d	15. b	16. c
17. d	18. a	19. b	20. c

Further Readings and Bibliography

1. “Introduction to Machine Learning” by Ethem Alpaydin
2. “Pattern Recognition and Machine Learning” by Christopher Bishop
3. “Machine Learning: A Probabilistic Perspective” by Kevin P. Murphy
4. “Machine Learning Yearning” by Andrew Ng
5. “Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow” by Aurélien Géron
6. “An Introduction to Statistical Learning” by Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani
7. “Deep Learning” by Ian Goodfellow, Yoshua Bengio and Aaron Courville
8. “Machine Learning” by Tom Mitchell, McGraw-Hill Education, 1st Edition.
9. “Artificial Intelligence: A Modern Approach” by Stuart Russell and Peter Norvig, Pearson, 3rd Edition
10. “Deep Learning” by Ian Goodfellow, Yoshua Bengio and Aaron Courville, Publication: The MIT Press, 1st Edition
11. “Natural Language Processing with Python” by Steven Bird, Ewan Klein and Edward Loper, O’Reilly Media, 1st Edition