The Edition of University Textbooks on
Informatics and Information Technologies

Introduction to Data Science

Giang Nguyen

STU FIIT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

SPEKTRUM STU

AIS ID: 103066

Introduction to Data Science

AIS ID: 103066

AIS ID: 103066

FIIT STU textbook for the course Intelligent Data Analysis

Introduction to Data Science

Giang Nguyen

Slovak University of Technology in Bratislava

2022

First edition.

REVIEWERS
Assoc. Prof. Ing. Ladislav Hluchý, PhD.
Assoc. Prof. Ing. Peter Bednár, PhD.
Assoc. Prof. Ing. Miloš Očkay, PhD.

Approval by the management of the
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES
SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA

PUBLISHED IN SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
IN SPEKTRUM STU PUBLISHING

SPEKTRUM STU

ISBN 978-80-227-5193-3

AIS ID: 103066

Annotation

This textbook presents an introduction to Data Science in the context of the responsible development of human-centric and trustworthy Artificial Intelligence systems. It presents the recent transition from focusing on modeling to the underlying data used to train and evaluate models. In the textbook, a systematical way to examine data is described with details about data source, data collection, data integration, and data preparation as a part of Data Science process. The aim of the process is to provide the best quality for machine learning data and to build an intelligent application in an organization. Intelligent data modelling is dedicated in one separated chapter to several selected machine learning methods and deep learning architectures. The emphasis is on model evaluation and model selection with optimizations to select the best models to deploy in production. In the context of intelligent software development, the textbook presents the most popular and the most used machine learning and deep learning frameworks and libraries with the dominance of Python open source software at small scale as well as large scale. A notation about specialized hardware for speeding up general purpose computation in the last decade is included. The textbook also emphasises ethics in Artificial Intelligence development with the most notable document "European Union Guideline on Ethics in Artificial Intelligence: Context and Implementation".

The textbook is intended for students of Data Science, Data Intelligence and Modelling courses as well as the general professional public dealing with or interested in Data Science and Artificial Intelligence. Sincere thanks and appreciations belong to all colleagues and partners for knowledge sharing and supports. Special thanks go to reviewers for constructive comments that significantly improve the quality of the work. The authors would like to thank to the management of the Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava for the approval of the study literature. The enthusiasm and the effort of students and PhD students for the Computer Science study at the faculty and for the course topics was the great motivation for the textbook preparation.

AIS ID: 103066

Contents

AIS ID: 103066

1. Introduction

This textbook presents an introduction to Data Science in the context of ubiquitous Artificial Intelligence in our lives. It is intended for students of Data Science, Data Intelligence and Modelling courses as well as the general professional public dealing with or interested in Data Science and Artificial Intelligence. The structure of the work is as follows.

Chapter 2 presents the Data Science process with concepts and methodologies in software development, software delivery, and software quality assurance such as development and operations (DevOps) and machine learning operations (MLOps). The chapter navigates readers through the beginning with data and data sources for machine learning to the deployment of building an intelligent data product in an organization. It presents data problems with possible solutions as well as worldwide and European Union concerns accompanied with law restrictions on data privacy and data protection.

The emphasis is on the trend of using Python as the main language for data analytic and data modelling as well as the dominance of large corporations in developing open source frameworks and libraries for Artificial Intelligence, machine learning, neural networks and deep learning at small scale and large scale.

Chapter 3 presents Exploratory Data Analysis (EDA) as a critical part of the Data Science process. The aim is to understand the data and to solve the quest. The chapter presents a systematical way to go through data with elementary tools such as plots, graphs and statistics. It presents EDA as the methods, which helps to make sure that the data product is performed as intended.

Chapter 4 is about data preparation for machine learning, which is the most tedious part of Data Science process. Data preparation is also called data wrangling and it has to be

done iterative according to the need and in cooperation with data collection and data integration as presented in Chapter 2. The chapter contains a number of techniques for data preparation such as data cleaning, data transformation, feature engineering including feature extraction, feature encoding, feature selection and dimensionality reduction as well as techniques to deal with imbalanced datasets. The aim is to provide the best quality for machine learning data as stated under Big Data V as Veracity.

Chapter 5 briefly presents an overview of machine learning algorithms as well as several supervised algorithms for regression and classification tasks like Linear Regression, Logistic Regression, Naïve Bayes, Decision Trees, and Random Forest in more detail. It also presents several fundamental deep learning architectures like Convolutional Neural Networks and Recurrent Neural Networks. The emphasis is on model evaluation and model selection as a part of the optimization process. The aim is to select the best models as the heart of an intelligent data product working in production within a real-world organization.

The chapter goes further through nature-inspired optimization methods, which provide a practical and elegant solution to many problems. These methods are designed to achieve approximately optimal solutions in practical execution times for non-deterministic polynomial-time hardness (NP-hard) optimization. The chapter closes with important considerations about learning system design with complexity, quality, and efficiency in realization.

Chapter 6 presents an overview of several most popular and the most used intelligent software for modelling, i.e., machine learning and deep learning frameworks and libraries with the dominance of Python open source software. Reference to broader studies with detailed comparison of software products is included in the chapter remarks.

An overview on Big Data distributed processing and analytic is also in Chapter 6, where the presence of hardware support for computation is needed and required. The chapter shows accelerated computing and Graphics Processing Unit (GPU) as the specialized hardware for speeding up general purpose computation in the last decade.

Chapter 7 presents advanced topics in Data Science and Artificial Intelligence (AI) such as responsible AI and Ethics in AI development with the most notable declarations from worldwide organizations, companies, universities, as well as the European Commission with the *European Union guideline on ethics in Artificial Intelligence: Context and implementation*. All of such efforts aim to the development of human-centric and trustworthy AI systems.

The chapter concludes the textbook on self-supervised learning as a future envision of AI presented by Yann Lecun in the Keynote Turing Award Winners Event of the Association for the Advancement of Artificial Intelligence (AAAI) conference in 2020.

2. Discover your quests with Data Science

A lot of people believe that the word *Data Science* is broadly mentioned and used after the publishing of the book *The Fourth Paradigm: Data-intensive Scientific Discovery* [1]. In this book, the first broad look at the way that increasing use of data is bringing a paradigm shift to the nature of science is presented. For reader interest, the four paradigms are empirical evidence (experimental paradigm: thousand years ago), scientific theory (theoretical paradigm: last few hundred years), computational science (computational paradigm: last few decades), and the fourth paradigm is called Data Science (the last decade).

Today, Data Science [2] is an *interdisciplinary* field that uses techniques and theories drawn from many fields within the context of mathematics, statistics, computer science, information technology, information science, domain and business knowledge (Figure 2.1). It uses scientific methods, processes, algorithms, and systems to extract knowledge and insights from many structural and unstructured data to solve analytically complex business problems [3]. Data Science is closely related to machine learning, data mining, data analytic, Data Intelligence and Modelling, Business Intelligence, Artificial Intelligence, AI-powered systems and usually Big Data [4].

> Data Science includes the processes of capturing data, analyzing, and deriving insights from it.

Data Science is all about using *data* to solve real life problems such as identifying which email is spam or not; monitoring anomaly states in systems; predicting the outcome, who will be the next President of the USA; recommendation which movie to watch, which item to buy; healthcare like tumor detection, drug discovery, medical analysis, virtual medical bots; transportation, self-driving cars; risk detection; environmental monitoring; and entertainment and Internet search.

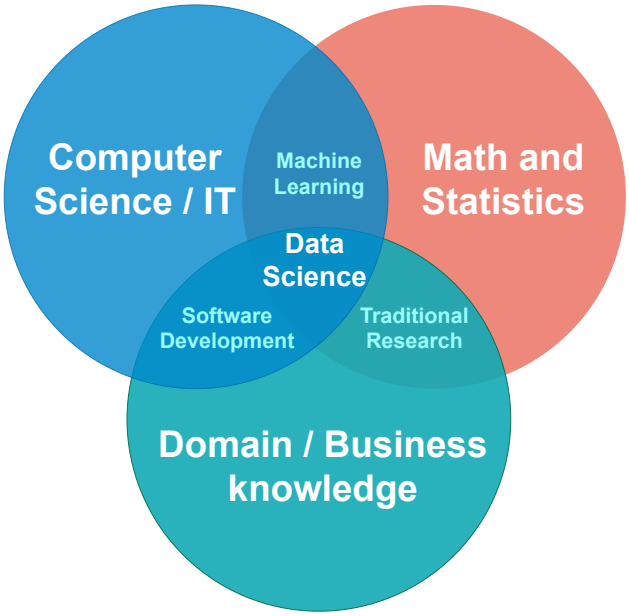


Figure 2.1: Data Science Venn diagram

In comparison with Data Science, data mining is a set of techniques mainly about finding useful information in a dataset. Data mining owes its origin to knowledge discovery in databases (KDD). Data Science aims to understand the data, extract useful information out of data, and apply this in problem solving. Prerequisites for doing Data Science are nontechnical prerequisites such as curiosity, critical thinking, proactive problem solving, effective communication, and business sense. They also are technical prerequisites such as ability to prepare data for effective analysis, to leverage machine learning and AI, to apply math and statistics properly, and ability of computer programming and databases.

> The aim of the Data Science process is to build a *data-centric product* for an organization.

A critical skill in Data Science is also the ability to decompose a data analytic problem into pieces such that each piece matches a known task of the available tools. Recognizing familiar problems and their solutions avoids wasting time and resources reinventing the wheel. It also allows people to focus attention on more interesting parts of the process that require human involvement that have not been automated, so human creativity and intelligence must come into play [5]. In many cases, the number of possible known tasks, which can be decomposed and sequentially combined to solve the problem is very high. Furthermore, here is not only one solution, but rather more possible solutions for solving the problem. Then an *intuition* about data is often mentioned, which has to be built based on the *knowledge* about the Data Science process and the *experience* of working with data. The aim is to decide on the building of a data product with feasible time constraints for an organization.

2.1 Data Science process

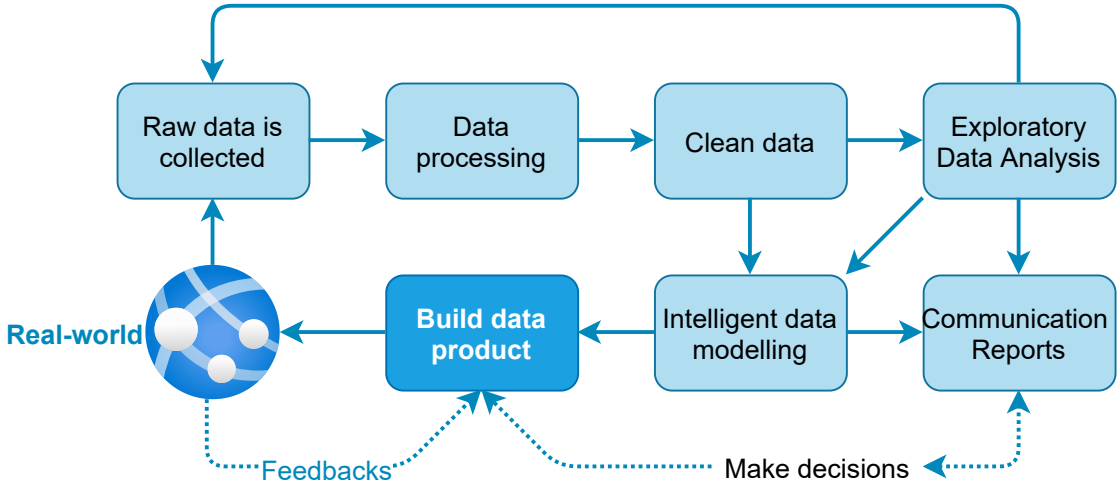


Figure 2.2: Data Science process

The Data Science process (Figure 2.2) goes though the following steps:

1. *The quest:* let have some *real-world* problem like spam filters, monitoring of medical records, recommendation systems or weather forecasting. Assumed that data on one of these things is available.
2. *The data:* raw data is collected. An example of raw data is some logs such as Enron employee emails, or recorded genetic material. Raw data have to be in a right format (like something with columns) and clean for analysis.
3. *Data processing:* Pipelines of data munging has to be built with many steps such as joining, scraping, transforming, wrangling using tools like Python, shell scripts, SQL, or all of them.
4. *Exploratory Data Analysis (EDA):* if we have clean data, we can do some kind of EDA. When do it, we may realize that data is not actually clean because of duplicates, missing values, absurd outliers, and data that was not actually logged or incorrectly logged. If that is the case, we may have to go back to collect more data, or spend more time cleaning the dataset.
5. *Communication and reports* are necessary based on EDA results and results of data modelling. This could take the form of reporting the results with visualization up to the managements and/or coworkers. The aim, in this case, is to build a data products for clients in an organization. In another case, reports can be used in scientific publishing and so on.
6. *Intelligent data modelling:* after EDA, we want to model our data using some intelligent algorithm like Logistic regression, Naïve Bayes, neural network, or something else. The algorithm is chosen based the solving problem, for example, classification problem, prediction problem, or a basic description problem.
7. *Data product building:* the key that makes Data Science special and distinct from

statistics is that this data product then gets *incorporated back into the real world*, and users interact with that product, and that generates more data, which creates a *feedback loop*.

8. *Decision making* about to build the data product (or not) is done based on previous steps. In many cases, data does not have enough power for modelling. In this case, we need to restart again with raw data from the real-world problem.

The key that makes Data Science distinct from statistics is that this data product gets incorporated back into the real world with *feedback loop* to improve its quality. Since data mining and data analytic can be viewed as a subset of Data Science, there is of course overlap. The aim of the Data Science process is to build *data-centric products* for an organization.

From the data protection viewpoint, especially in the context of the European Union (EU) General Data Protection Regulation (GDPR), Data Science ethics have to protect users and the general public. Businesses should collect the data they need but no more. Data Science ethics have to promote transparency and guard privacy for sensitive data protection and have ability to react quickly and professionally to data breaches.

2.1.1 Data mining concepts and methodologies

Data mining is the process of discovering actionable information from large datasets. It uses mathematical analysis to derive patterns and trends that exist in data. Typically, these patterns cannot be discovered by traditional data exploration because the relationships are too complex or because there is too much data. Historically, the three most significant methodologies in data mining are SEMMA, 5A Process and *Cross-Industry Process for Data Mining (CRISP-DM)*[1].

SEMMA stands for Sample, Explore, Modify, Model and Assess. It was developed by the Statistical Analysis System (SAS) Institute Inc.. SAS declares SEMMA as a logical organization of the functional tool set of SAS Enterprise Miner for carrying out the core tasks of data mining. SEMMA mainly focuses on modeling tasks while leaving the business aspects out.

5A Process stands for Assess, Access, Analyze, Act and Automate. It was developed by the IBM Statistical Package for the Social Sciences (SPSS).

Cross-Industry Process for Data Mining (CRISP-DM)

CRISP-DM is one of the most used DM concept and methodology [6]. The project was funded in part by the European Commission (EC) with sponsors NCR Systems Engineering Copenhagen (USA and Denmark), DaimlerChrysler AG (Germany), SPSS Inc. (USA), and OHRA Verzekeringen en Bank Groep B.V. (The Netherlands). CRISP-DM succeeds because it is soundly based on the practical, real-world experience of how people

[1] CRISP-DM https://www.the-modeling-agency.com/crisp-dm.pdf

conduct data mining projects. CRISP-DM is reported as the leading and wide-accepted methodology for data mining.

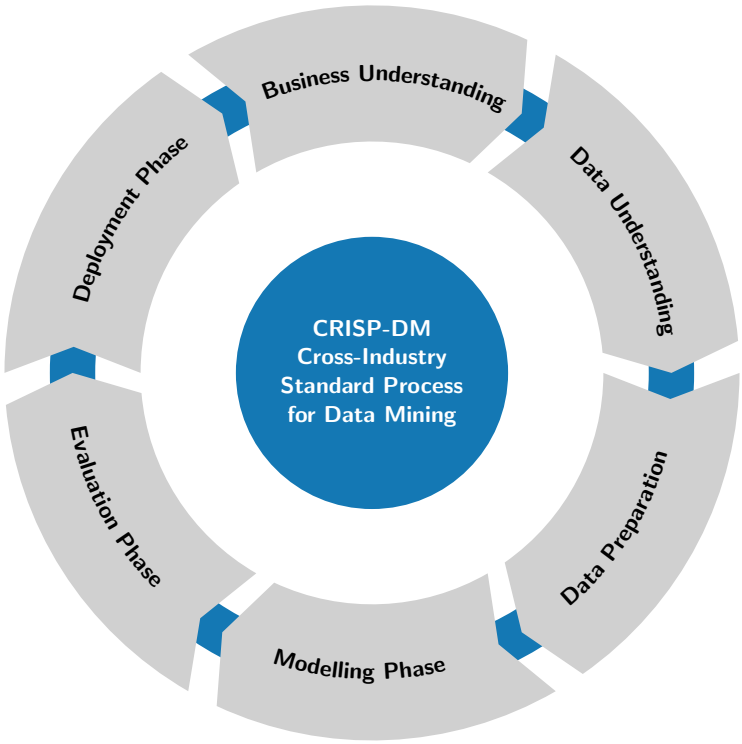


Figure 2.3: Cross-Industry Process for Data Mining

CRISP-DM consists of six steps (Figure 2.3):

1. *The business understanding* is usually based on the provided *quest formulations* and data description.
2. *The data understanding* is based on the provided data and its documentation.
3. *The data preparation* consists of data transformation, EDA and feature engineering. Each of them can be further divided into smaller sub-steps, for example, feature engineering consists of feature extraction, feature selection.
4. In *the modelling* phase, various machine learning algorithms can be applied with different parameter calibrations. The combination between data and parameter variability can lead to extensive repeating of the model train-test-evaluation cycle. If the data is large-scale, the modeling phase will have time-consuming and compute-intensive requirements.
5. *The evaluation* phase can be performed under various criteria for thorough testing of the machine learning models in order to choose the best model for the deployment phase.
6. *The deployment* phase, also called the production phase, involves usage of a trained model to exploit its functionality, and the creation of a data pipeline into production.

The whole CRISP-DM cycle is repetitive. The group of the first five phases, called the *development phase*, can be repeated with different settings according to the evaluation results. The *deployment phase* is critical for real production under repetitive requisites; it implies online evaluation, monitoring, model maintenance, diagnosis, and retraining. It is needed to highlight that *machine learning algorithms learn from data*. Therefore, in practice, it is expected that the data understanding and data preparation phases can consume a large portion of the entire time of every data-driven project.

The data preparation step consists of substeps: data transformation, EDA and feature engineering. Data preparation methods can be categorized into a number of sub-categories such as dimensionality reduction, sampling (sub-sampling, oversampling), data augmentation, statistical testing, feature engineering with feature extraction, feature encoding, feature transformation and feature selection using mutual information, chi-square χ^2 statistics, and many more. There are also many other methods, for example, overfitting prevention by regularization, threshold setting, pruning, or dropout.

Analytics Solution Unified Method

IBM Analytics Solutions Unified Method (ASUM) is currently considered as an extended and refined CRISP-DM. ASUM has five phases 1) analyze, 2) design, 3) configure and build, 4) deploy, and 5) operate and optimize. The three phases of ASUM of analyze, design, configure, and build have been combined due to the iterative nature of data mining and predictive data analytic projects [7].

Data mining techniques rely on data to be analyzed to get insights of these data providing relevant information for the problem that is being analyzed. The change in data generation and data collection lead to changes in data processing as well.

2.1.2 DevOps, MLOps, software delivery and quality assurance

Data Science process is built on software development methodologies such as development and operations (DevOps) and machine learning operations (MLOps).

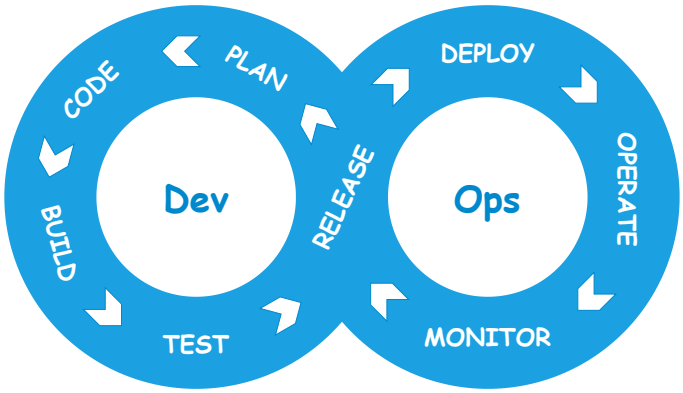


Figure 2.4: DevOps cycle

The term DevOps was created by Patrick Debois in 2009 (Figure 2.4). It can be seen as an extension of Agile[2] software development principles. In software development, Agile is a set of practices intended to improve the effectiveness of software development professionals, teams, and organizations. DevOps is a set of practices that combines software development (Dev) and IT operations (Ops) aiming to shorten the development life cycle and provide continuous integration and continuous delivery (CI/CD) with high software quality assurance (QA) (Figure 2.5). DevOps encompasses culture and people within an organization, aiming to improve collaboration between the development team and IT operations.

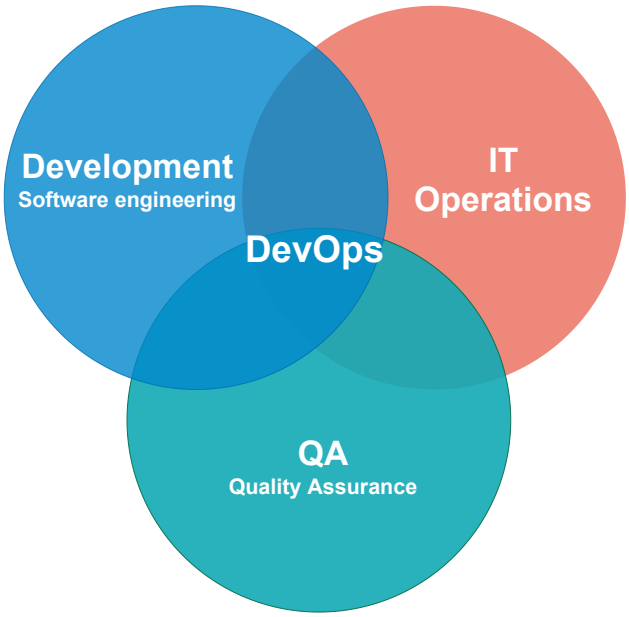


Figure 2.5: DevOps: Development, IT Operations and Quality Assurance

DevOps components are: Plan, Code, Build, Test, Release, Deploy, Operate and Monitor. Key features of DevOps architecture are Automation, Collaboration, Integration and Configuration management.

MLOps is modeled on the existing discipline of DevOps. It is a set of practices that aims to deploy and maintain machine learning models of production reliably and efficiently (Figure 2.6).

DevOps tools help organizations to deal with the challenges that come with the implementation of DevOps practices. However, there is no *"one size fits all"* and therefore as a result, there is a wide variety of DevOps tools[3] such as version control tools and collaboration (Git, Confluence, Jira), container management tools (Docker, Kubernetes,

[2]https://www.agilealliance.org/agile101/
[3]https://www.qentelli.com/thought-leadership/insights/devops-tools

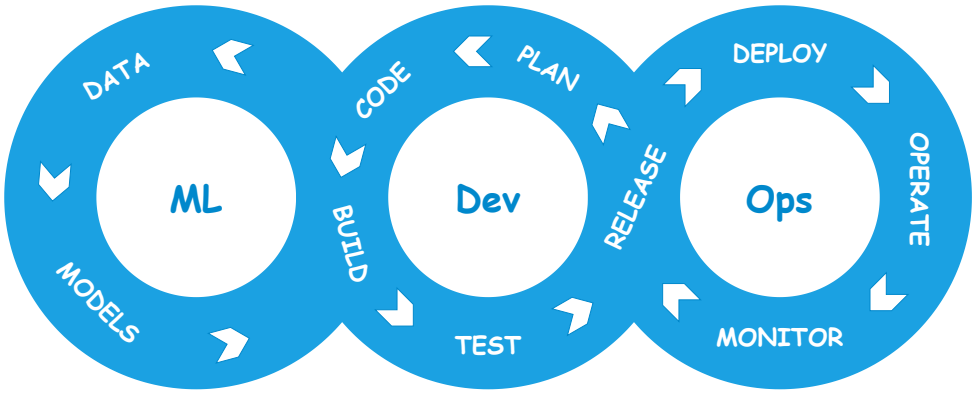


Figure 2.6: Machine learning operations MLOps = ML + Dev + Ops

Mesos, AWS), application performance monitoring tools (Prometheus, Dynatrace), deployment and server monitoring tools (Splunk, Datadog, Nagios, Snort, Sensu), configuration management tools (Ansible, Chef, Puppet), CI/CD deployment automation tools (Jenkins, Codeship, Bamboo, IBM UrbanCode), test automation tools (Test.ai, Ranorex, Selenium, JUnit), artifact management tools (NEXUS, JFRog Artifactory, CloudRepo), codeless test automation tools (AccelQ, Appvance, Testim.io). Implementation of DevOps tools in organizations in maturity operational state can not be done in a short time. It requires an cooperative effort in shifting culture to break down communication silos, which will enable to provide better software and ensure enhanced transparency in the entire chain.

2.2 Data and data sources

Today, 5Vs (Volume, Velocity, Variety, Veracity, and Value) are the five keys to making Big Data a huge business. The first V for Volume is the base of Big Data. The volume of worldwide data doubles about every 40 months. The second V is Velocity can be more important than volume in many cases. It is better to have limited data in real time than lots of data at a low speed. In general, data have to be available at the right time to make appropriate decisions. The third V is for Variety, which describes various sources of data. Data can come from many different spaces, from in-house devices to smartphones, GPS technology, or from social networks. It can have many layers and many formats.

> The forth V for Veracity in the context of this book refers to quality of machine learning data.

The fifth V is for Value. It refers to the ability to transform a tsunami of data into business. Machine learning is a kind of AI that allows software applications to improve automatically through experience and using data without being explicitly programmed to do. Machine learning algorithms use historical data as input to predict new output values.

> Machine learning algorithms learns from machine learning data.

2.2.1 Machine-generated data, human-generated data

Machine-generated data is automatically generated by a computer process, application, Internet of Things (IoT), or other mechanism without the active intervention of a human. It crosses all industry sectors, it has no single form; but rather, the type, format, metadata, and frequency respond to some particular business purpose. Although such data are frequently unstructured but they are easily transformed into a structure. Machines often create data based on a defined time schedule or in response to a state change, action, notification, or other events. The data is not often or prone to be updated or modified in a machine lifetime, and due to this characteristic, machine-generated data is often considered as highly reliable. Today, most of the growth in data is the byproduct of machine-generated data (Volume).

Human-generated data is comprised of all files that people create and share every day, one to others like emails, documents, spreadsheets, presentations, images, audio, and video files. These are the files that take up the vast majority of the digital storage space in most organisations and they are kept for significant amounts of time. Such data have large amounts of metadata associated with them. Metadata is the information about a file: who might have created it, what type of file it is, what folder it is stored in, who has been reading it, and who has access to it. The content and metadata together make up human-generated data and Big Data.

Monitoring systems

In general, monitoring systems produce domain data in divergent formats with divergent sizes and raw quality. They can be cyber-physical system, process control measurements, system records and notifications, system functionality monitoring, security and cyber-security monitoring, energy consumption monitoring, financial stock markets including cryptocurrency investments, the global positioning system (GPS), satellite systems and remote sensing, transport and safety monitoring, IoT devices (weather temperature and pressure, water level and speed, airflow speed and direction, environmental humidity, radiance and pollution), healthcare monitoring, smart watch, telemetric monitoring, personalized platforms (e.g., Google and Facebook), online purchases and shopping (e.g., Amazon and Ali-Express), survey rating, collective question and answers. Examples of monitoring systems are Python software packages like Scapy and Scrapy.

SCAPY[4] is a packet manipulation tool for computer networks. It is written in Python. It can forge or decode packets, send them on the wire, capture them, and match requests and replies. It can also handle tasks like scanning, tracerouting, probing, unit test, attacks, and network discovery. The installation is simple (`pip install -pre scapy[basic]`).

SCRAPY[5] is a tool to extract data from websites. With Scrapy, users can define their own data structures, create their own spider, leverage built-in Xpath and CSS selectors to

[4] https://scapy.net/
[5] https://scrapy.org/

extract desired data, built-in JSON, CSV, XML output, have an interactive shell console, download and process files and images. Similar tools to Scrapy are Beautiful Soup[6] and Selenium[7] for pulling data out of HTML and XML files with the simple Python software package installation (`pip install beautifulsoup4`).

It is suitable to note that, the standard package manager for Python is *pip*, which looks for packages in *PyPI*[8] (the Python Package Index, i.e., the repository of software for the Python programming language). *pip* resolves library dependencies, and installs required python packages in the current Python environment to ensure that requests will work.

2.2.2 Synthetic data and simulated data

Synthetic data is data that is artificially created rather than being generated by actual events. It is often created with the help of algorithms and is used for a wide range of activities, including as test data for new products and tools for model validation, and in AI model training. Synthetic data is important because it can be generated to meet specific needs or conditions that are not available in existing (real) data. Even such data may be artificial, but synthetic data reflects real-world data, mathematically or statistically. Research demonstrates it can be as good or even better for training an AI model than data based on actual objects, events or people.

Synthetic data can be divided into fully synthetic data and partially synthetic data. Fully synthetic data does not contain any original data. This means that reidentification of any single unit is almost impossible and all variables are fully available. Partially synthetic data, where only the data part, which is sensitive, is replaced with synthetic data.

Simulated data is synthetic data. Surely, traditional synthetic data will be enough for many applications. However, simulated data represents the ultimate goal of fully simulating the world around in synthetic detail, e.g., high-quality scanned 3D data or medical research where patient privacy is extreme strict. The simulation creates individual record data based on a detailed analysis of a real observational database. Another example is honeypots for computer security, where simulated datasets are used to increase the robustness of systems. Realistic scenarios can be realised instead of long time monitoring for rare events.

2.2.3 Supervised learning and data labelling

In machine learning, data labeling is the process of identifying raw data (images, text files, videos, etc.) and adding one or more meaningful and informative labels to provide context so that a machine learning model can learn from it. Most practical machine learning models utilize supervised learning (deep or not), which applies an algorithm to map inputs to output. Data labeling is required for a variety of use cases including natural

[6] https://pypi.org/project/beautifulsoup4/
[7] https://www.selenium.dev/
[8] https://pypi.org/

language processing, computer vision and speech recognition. In these cases, a labeled set of data is compulsory that the model can learn from to make correct decisions.

Labels might indicate categories, for example, if a photo contains a edible or inedible fungi, which words are in an audio recording, or if an x-ray image contains a tumor. Labeled data highlights data features (other words: properties, characteristics, or classifications) that can be analyzed for patterns that help predict the target[9].

Data labeling is typically done with humans making judgments about a given piece of unlabeled data. Data annotation is the task of labeling data in preparation for training a machine learning model. It is very high-cost and hard work, which maybe biased. Today, mostly labelled data is done by India data labellers to power the global AI race.

Here are also a lot of annotation tools, which partially relieve the manual labeling work such as Computer Vision Annotation Tool (CVAT)[10] for video annotation, Scale AI[11] with human-powered high-quality training data for AI applications such as self-driving cars, augmented reality (AR), virtual reality (VR), robotics, mapping; Prodigy[12] an annotation tool for AI, machine learning and NLP; as well as a number of other labelling tools such as LabelMe, LabelBox, LabelImg, Label-studio, OpenLabeler, OpenLabelling.

2.2.4 Public data source for data modelling

Here are a lot of public data repositories, which contain data in various states, in many cases in a preprocessed form suitable for data modelling, for example, by machine learning algorithms. The most well-known repositories are:

UCI	http://archive.ics.uci.edu/ml/index.php
Kaggle Datasets	https://www.kaggle.com/datasets
Google Dataset Search	https://datasetsearch.research.google.com/
IEEE Dataport	https://ieee-dataport.org/
CERN	http://opendata.cern.ch/
NASA	https://data.nasa.gov/
Open Data	http://opendata.sk; https://slovak.statistics.sk

2.2.5 Data privacy and data protection

Data privacy is an area of data protection that concerns the proper handling of sensitive data, including notably personal data but also other confidential data, such as certain financial data and intellectual property data, to meet regulatory requirements as well as protecting the confidentiality and immutability of the data [8].

[9] https://www.cloudfactory.com/data-labeling-guide
[10] https://cvat.org/
[11] https://scale.com/
[12] https://prodi.gy/

Personal data, which is also known as Personally Identifiable Information (PII), means any information which can be used to distinguish or trace the identity of an individual (e.g., name, social security number, biometric records, etc.) alone, or when combined with other personal or identifying information which is linked or linkable to a specific individual (e.g., date and place of birth, mother's maiden name, etc.).

General Data Protection Regulation (GDPR) [9] can be considered as the world's strongest set of data protection rules, which enhance how people can access information about them and places limits on what organisations can do with personal data. The full text of GDPR contains 99 individual articles and it is built on previous data protection principles. The GDPR's final form came about after more than four years of discussion and negotiations and it was adopted by both the European Parliament and European Council in 2016. GDPR came into force on May 25, 2018 [10] at the core contains seven key principles:

1. lawfulness, fairness and transparency;
2. purpose limitation;
3. data minimisation;
4. accuracy;
5. storage limitation;
6. integrity and confidentiality (security);
7. accountability.

It is suitable to note that data privacy is not data security. Data security protects data from compromise by external attackers and malicious insiders, whereas data privacy governs how the data is collected, shared, and used.

The strength of GDPR has seen it lauded as a progressive approach to how people's personal data should be handled. Some comparisons have been made with the California Consumer Privacy Act (CCPA) [11] as well as other data protection regulations around the world. One of the biggest, and most talked about, elements of the GDPR is the ability to allow the EU's Data Protection Authorities to issue fines of up to €20 million or 4 percents of annual global turnover. The biggest fines were reported €50 million (against Google), £183 million (against British Airways) and £99 million (against the hotel chain Marriott) for breaching GDPR [10].

2.3 Data gathering and data integration

The most notable problems with raw data for Data Science process are: data are not correct; data are not timely; data are not measured or indexed properly; and data do not exist. The most typical causes of such problems are: data are generated carelessly; raw data are processed inaccurately; data were tempered; the method for generating data is not rapid enough to meet the need of data; raw data are gathered inconsistently with the purpose of the analysis. Possible solutions for such data problems can be:

- to develop a systematic way to produce monitoring data,

- to automate the data ingestion process,
- to introduce quality control on data production,
- to establish an appreciated security program,
- to use data warehouse approaches,
- to use suitable search engine,
- to predict which kind of data will be needed in the future,
- to generate new data suitable for the purpose.

Data gathering

Data gathering or data collection is the process of gathering or measuring data from domain systems to answer relevant questions with relevant outcomes. The data gathering process can come repeatly through the following steps:

1. Discussing the project with subject matter experts;
2. Selection of variables to be used as input and output for a predictive model;
3. Reviewing the data that has been collected;
4. Summarizing the collected data using statistical methods;
5. Visualization of the collected data using plots and charts.

Here are great challenges to data gathering when the Internet is large and overloaded with nonrelated or malicious spam pages. Internet pages are dynamically generated using spider traps. The challenge is also with content freshness, i.e. , crawlers should be catch-up with new and up-to-date contents with content deduplication from site mirrors and duplicate pages.

Data gathering has to be realized as a trade-off between exploration and exploitation. In the data gathering context, exploration is the search for new sources of relevant pages. The aim is to gather more information. An example of the exploration process can be acquiring more information about new restaurants in surrounding locations.

Exploitation can be understood as the crawling of pages where the expected value can be predicted with high confidence. The aim is to make the best decision given current information. An example of the exploitation process is to go to the favourite restaurant. The trade-off in data gathering is the same as another online decision making. The best long-term strategy may involve short-term sacrifices and here is the need to gather enough information to make the best overall decisions.

Data integration

Data Science combines various disciplines of statistics, data mining, databases, and distributed systems to extract information from large-scale sets of data. Challenges in data integration come from the variety of physical systems with various hardware standards and distributed deployments, which lead to various data formats. They also come from logical structures with different data models and different data schemas. Furthermore, business organizations have various requirements on data security and privacy, business

rules, and requirements for different administrative zones in the business.

Data integration components are data migration (moving data between locations, formats, or applications), Enterprise Application Integration (enables interoperability between systems, Master Data Management (makes an effort to create one single master reference), data aggregation with Extract, Transform, Load operations (ETL, which combines disparate data source). The problem of entity resolution across databases can be eliminated by deduplication (eliminating duplicate exact copies of repeated data), record linkage (identifying records that reference the same entity across different sources), canonicalization (converting data with more than one possible representation into a standard form), and reference matching (the process of matching noisy records to clean ones in a deduplicated reference table).

2.4 Data Science and Python ecosystem

`Python` is a programming language created by Guido van Rossum and first released in 1991. It is a interpreted high-level general-purpose, versatile, Object-Oriented Programming (OOP) language [12], [13]. Python is useful and powerful while also being readable and easy to learn. This makes it suitable for programmers of all backgrounds and is likely the reason Python is one of the most widely used programming languages. It is successfully used in thousands of real-world business applications around the world, e.g., Google and YouTube. Python's design philosophy (THE ZEN OF PYTHON) emphasizes the simplicity principle [14]:

> Beautiful is better than ugly.
> Explicit is better than implicit.
> Simple is better than complex.

Python features a dynamic system and automatic memory management, with large and comprehensive libraries for scientific computation and data analysis. The Python ecosystem of libraries, frameworks, and tools is enormous and growing. Python runs on a wide variety of UNIX platforms, Windows, and MacOS. Python is easy to use and extensible via packages, e.g., for web scraping, data processing, data analysis, machine learning, web development, NLP, IoT development, DevOps, MLOps, general scientific computing and many other computing and scripting uses [15]. The use of Python can be shortly listed as follows:

- for ML and DL: Scikit-learn, MLLib, Tensorflow, Keras, PyTorch, fast.ai, ...
- for fast development: Jupyter Notebook, JupyterLab, JupyterHub, ...
- for data visualization: Matplotlib, Seaborn, Plotly, ...
- for web development: Django, Flash, ...
- for web crawling and scraping: Beautiful Soup, Scapy, Scrapy, ...
- for data and Big Data processing: Pandas, Apache Spark, Apache Flink, ...

- for scientific numerical calculations and statistics: NumPy (Numerical Python), SciPy (Scientific Python), StatsModels (Statistical Module)
- for NLP: Natural Language Toolkit (NLTK), Gensim, Spacy, ...
- for managing processes instead of shell scripting,
- for DevOps and MLOps: Ansible and SaltStack are written in Python.

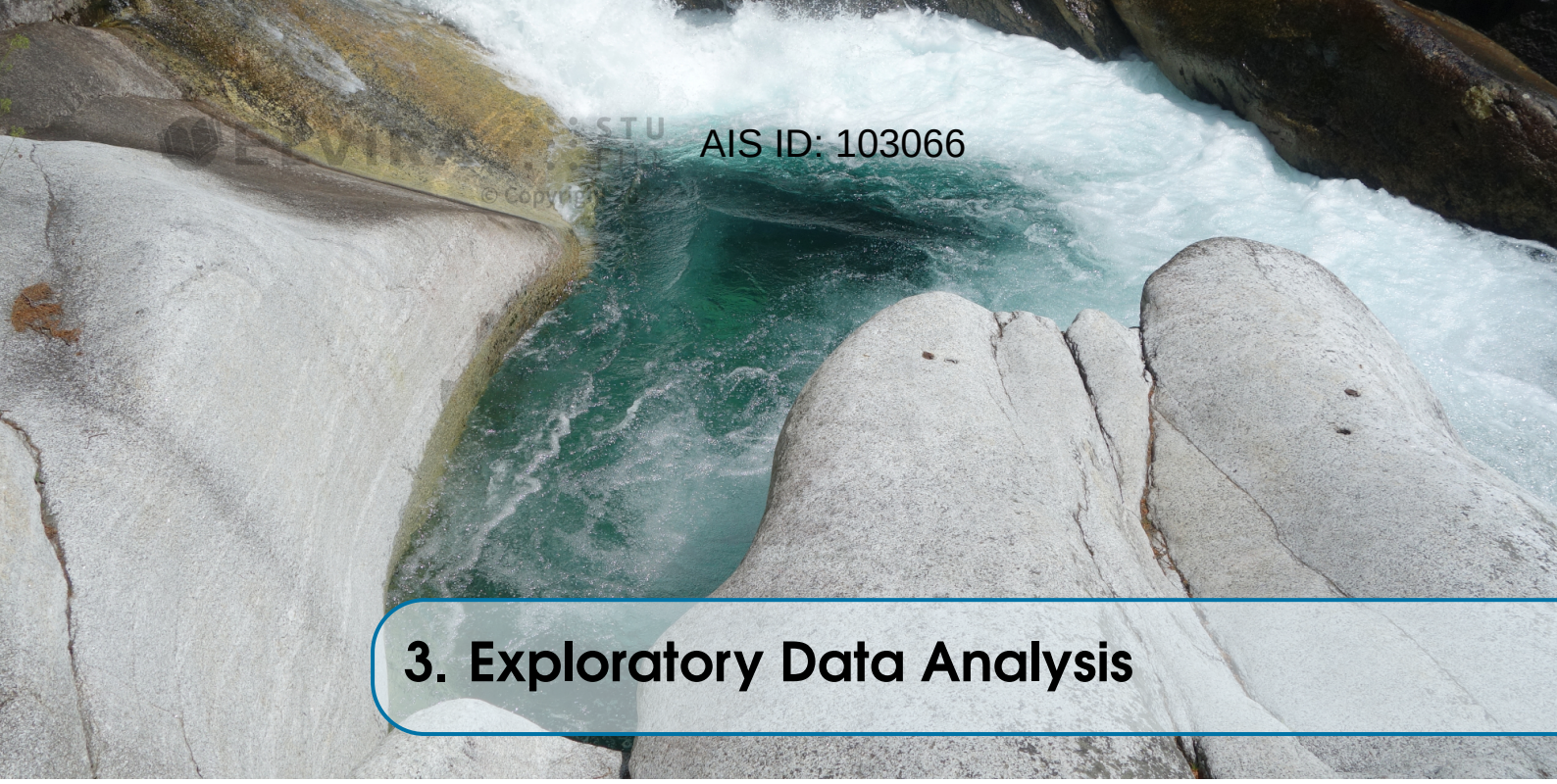
Primarily, the rationale for adopting Python for Data Science and MLOps is because it is a general purpose programming language for research, development and production (DevOps) at both small scale and large scale.

2.5 Remark

Rachel Schutt et al. *Doing data science - Straight talk from the frontline.* " O'Reilly Media, Inc.", 2014. ISBN: 978-1449358655

Zed A Shaw. *Learn python 3 the hard way: A very simple introduction to the terrifyingly beautiful world of computers and code*. Addison-Wesley Professional, 2017. ISBN: 978-0134692883

AIS ID: 103066



AIS ID: 103066

3. Exploratory Data Analysis

In today's world, over 2.5 quintillion bytes ($2.5 \cdot 10^{18}$ bytes) of data are created every single day. It is a good practice to understand the data firstly and try to gather as many insights from it. Exploratory Data Analysis (EDA) is all about making sense of the data in hand, before getting them dirty with it. In computing, data is information that has been translated into a form that is efficient for movement or processing. In general, from the technical and mathematical viewpoint, data is a collection of facts, such as numbers, words, measurements, observations, or just descriptions of things. Data can be qualitative or quantitative as follows:

Numerical data or quantitative data have meaning as a count or a measurement. Numerical data can be further divided into:

- Discrete data, which can only take certain values, e.g., the number of students in a class (integer number only) or the results of rolling 2 dice (only having the values 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 and 12). In short, discrete data is counted.
- Continuous data, which can take any value within a range (float number), e. g., a person's height (can be any positive float number), time in a race (any fraction of a second). In short, continuous data is measured.

Categorical data or qualitative data represent characteristics such as a person's gender, marital status, hometown, or the types of movie they like. Categorical data can take on numerical values, but those numbers do not have mathematical meaning.

Ordinal data can be mixed with numerical and categorical data. The data fall into categories, but the numbers placed on the categories have meaning. A typical example for ordinal data is rating a restaurant on a scale from 0 (lowest) to 4 (highest) stars. Ordinal data are often treated as categorical data.

Except for above-mentioned types of data, the real-world data can have other types:

Text data, which usually consists of files or documents. They can represent words, sentences or even paragraphs of free flowing text. The inherent unstructured (no neatly formatted data columns) and noisy nature of textual data makes it harder for intelligent methods to directly work on raw text data. There can be multiple ways of cleaning and pre-processing textual data, e.g., using natural language processing techniques and pipelines [16].

Time series data is a collection of observations obtained through repeated measurements over time. Such data is everywhere, since time is a constituent of everything that is observable. As our world gets increasingly instrumented, sensors and systems are constantly emitting a relentless stream of time series data. Such data has numerous applications across various industries, for example, tracking daily, hourly, or weekly weather data; tracking changes in application performance; tracking medical devices to visualize vitals in real time; tracking network logs [17].

Multimedia data refers to data that consist of various media types like text, audio, images, animation, video, and interactivity are combined.

3.1 Understand your data with EDA

EDA is a critical part of the Data Science process, and also represents a philosophy or way of doing statistics practiced by a strain of statisticians coming from the Bell Labs tradition. John Tukey, a mathematician at Bell Labs, developed EDA in contrast to confirmatory data analysis. In EDA, there is no hypothesis and there is no model. The *exploratory* aspect means that your understanding of the problem you are solving, or might solve, is changing as you go.

> Exploratory Data Analysis (EDA) is an attitude, a state of flexibility, a willingness to look for those things that we believe are not there, as well as those we believe to be there. (John Tukey)

Generally speaking, EDA is a method of systematically going through the data by plotting distributions of all variables, plotting the time series of data, transforming variables, looking at all pairwise relationships between variables using scatter plot matrices, and generating summary statistics for all variables including variables mean, minimum, maximum, upper and lower quartiles, and identifying outliers.

> The basic tools of EDA are plots, graphs, and summary statistics.

But as much as EDA is a set of tools, it's also a *mindset*. And that mindset is about your relationship with the data. You want to understand the data, gain intuition, understand the shape of it, and try to connect your understanding of the process that generated the data to the data itself. EDA happens between you and the data and is not about proving anything to anyone else yet [2]. There are important reasons anyone working with data

should do EDA. Namely, to gain intuition about the data; to make comparisons between distributions; for sanity checking (making sure the data is on the scale you expect, in the format you think it should be); to determine where the data is missing or if there are outliers; and to summarize the data.

At the end, EDA helps to make sure that the product is performing as intended.

EDA strategy and methods

The EDA strategy is to examine variables one by one, then look at the relationships among the different variables. It starts with graphs, then adds numerical summaries of specific aspects of the data. EDA is an iterative process to identify and prioritize relevant questions in a decreasing order of importance. EDA common questions can look like:

- What is a typical value?
- What is the uncertainty for a typical value?
- What is a good distributional fit for a set of numbers?
- Does an engineering modification have an effect?
- Does a factor have an effect?
- What are the most important factors?
- Are the measurements coming from different laboratory equivalents?
- What is the best function for relating a response variable to a set of factor variables?
- What are the best settings for factors?
- Can we separate signal from noise in time-dependent data?
- Can we extract any structure from multivariate data?
- Does the data have outliers?

The objectives of EDA are to discover patterns, spot anomalies, frame hypothesis, and check assumptions. Methods for EDA are univariate analysis and bivariate analysis for variables are done by discovery of trends, distribution, mean, median, mode, outlier detection, spread measurement, correlations, hypothesis testing and visual exploration.

3.2 EDA and descriptive statistics

Exploratory Data Analysis (EDA) uses many statistic methods such as numeric summaries, plot, aggregation, distribution, density for deep and clear understanding of the data, which provides the foundation for feature engineering and model selection to solve the problem [18] . Before making inferences from the data, it is essential to examine all variables. Data can be divided into several basic types like categorical such as binary, nominal, ordinal; and quantitative such as discrete and continuous.

Dimensionality of data can be named as follows:

- Univariate: measurements made on one variable per subject.
- Bivariate: measurements made on two variables per subject.
- Multivariate: measurements made on many variables per subject.

3.2.1 Frequency analysis

Frequency analysis is a part of descriptive statistics. In statistics, frequency is the number of times an event occurs. It deals with the number of occurrences (frequency) and analyzes measures such as:

- Measures of central tendency like mean, median, and mode;
- Measures of dispersion such as standard deviation, variance (and range);
- Percentile value, which shows the value shows what percent of values in a data set fall below a certain percent;
- Skewness and kurtosis.

It is essential for the analysis and interpretation of any data at a glance.

Measure of central tendency

Measures of central tendency try to describe the set of data through a value that represents the central position within that data set. The most popular measures of central tendency used for frequency analysis are mean, median, and mode.

Let N be the population size and n be the sample size.

The *mean* (or the average value) of a population is usually denoted as μ (Equation 3.1):

$$\mu = \frac{x_1 + x_2 + \cdots + x_N}{N} \tag{3.1}$$

The mean of a sample is usually denoted as $\overline{x}$ (Equation 3.2):

$$\overline{x} = \frac{x_1 + x_2 + \cdots + x_n}{n} \tag{3.2}$$

The *median* is the middle value of the sorted values. It is calculated as in Equation 3.3 for an odd number of observations:

$$x_{median} = x_{\frac{n+1}{2}} \tag{3.3}$$

If there are an even number of observations, find the middle two values and average them (Equation 3.4).

$$x_{median} = \frac{x_{\frac{n}{2}} + x_{\frac{n}{2}+1}}{2} \tag{3.4}$$

The *mode* is the most likely or frequently occurring value. The most common measure of central tendency is *mean*. For skewed distribution or when there is concern about outliers, the *median* may be preferred. *Mode* is a rarely used measure of central tendency.

■ **Example 3.1** Let $speed = \{99, 86, 87, 88, 111, 86, 103, 87, 94, 78, 77, 85, 86\}$ then the mean $\mu = 89.769$; $median = 87$; $mode = 86$ with $count = 3$. ■

Measures of dispersion (scale)

Measures of dispersion describe the spread measurement (or variability) of a dataset. The most popular measures of dispersion used for frequency analysis are standard deviation, variance, and range.

Variance: the population variance is denoted by σ^2 (Equation 3.5).

$$\sigma^2 = \frac{\sum_{i=1}^{n} \left(x_i - \mu\right)}{N} \tag{3.5}$$

The sample variance is denoted by s^2 (Equation 3.6).

$$s^2 = \frac{\sum_{i=1}^{n} \left(x_i - \overline{x}\right)}{n-1} \tag{3.6}$$

Standard deviation: is the average deviation of points from the mean value of the population, denoted as σ (Equation 3.7).

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N} \left(x_i - \mu\right)^2}{N}} \tag{3.7}$$

The standard deviation for the sample is denoted as s (Equation 3.8).

$$s = \sqrt{\frac{\sum_{i=1}^{n} \left(x_i - \overline{x}\right)^2}{n-1}} \tag{3.8}$$

■ **Example 3.2** Let $speed = \{99, 86, 87, 88, 111, 86, 103, 87, 94, 78, 77, 85, 86\}$ then the variance $\sigma^2 = 85.715$; the standard deviation $\sigma = 9.258$. ■

Percentile values

In general, the n^{th} percentile is a value such that $n\%$ of the observations fall at or below or it. Frequency analysis commonly uses percentile values like quartiles, deciles, and percentiles. The first quartile Q_1 is the value for which 25% of the observations are smaller and 75% are larger. The second quartile Q_2 is the same as the median, i.e., 50% are smaller, 50% are larger. Only 25% of the observations are greater than the third quartile Q_3. Concretely:

$Q_1 = 25^{th}$ percentile,
$Q_2 = 50^{th}$ percentile,
$Q_3 = 75^{th}$ percentile.

Interquartile range (IQR) is a measure of statistical dispersion, being equal to the difference between 75^{th} and 25^{th} percentiles, or between upper Q_3 and lower quartily Q_1. The IQR is a robust measure of spread (Equation 3.9).

$$IQR = Q3 - Q1 \tag{3.9}$$

Levene test of variances

Variance of samples can be tested using Levene test, which tests if all input samples are from populations with equal variances. The test is used to test if k samples have equal variances[1]. Equal variance across samples is called *homogeneity of variance*.

The assumptions:
- Independent observations;
- The test variable is quantitative, i.e., not nominal or ordinal.

The hypotheses:
- H_0: all input samples are from populations with equal variances;
- H_1: at least one pair of testing groups do not have equal variance.

and meaning
- Fail to reject H_0: equal variances
- Reject H_0: other variances

The W statistic to test if the means are different is calculated as in Equation 3.10.

$$W = \frac{(N-k)}{(k-1)} \cdot \frac{\sum_{i=1}^{k} n_i \left(\mu_i - \mu\right)^2}{\sum_{i=1}^{k} \sum_{j=1}^{n_i} \left(\mu_{ij} - \mu_i\right)^2} \tag{3.10}$$

where

k is the number of different samples to which the observations belong to;
n_i is the size of the i^{th} sample;
N is the total size of all samples;
μ_{ij} can be mean, median or trimmed mean of the i^{th} sample;
μ_i are the sample means of the μ_{ij};
μ is the overall mean of μ_{ij}.

[1] https://www.itl.nist.gov/div898/handbook/eda/section3/eda35a.htm

The W statistic (Equation 3.10) is equivalent to the one-way between-groups analysis of variance (ANOVA) or the F statistic (Equation 3.24).

Another name for homogeneity of variance is *homoscedasticity*, which means *having the same scatter* or the values in datasets are scattered to the same extent. The opposite of homoscedasticity is *heteroscedasticity*, which means *having the different scatter*. Homoscedasticity is one of the assumptions of linear regression modelling (Section 5.1.1)

3.2.2 Normal distribution

A large fraction of the statistic field is concerned with data that assumes that it was drawn from a *normal distribution*. Normal distribution is also called *Gaussian distribution* (Equation 3.11, Figure 3.1).

$$F(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{3.11}$$

where μ is the mean and σ is the standard deviation.

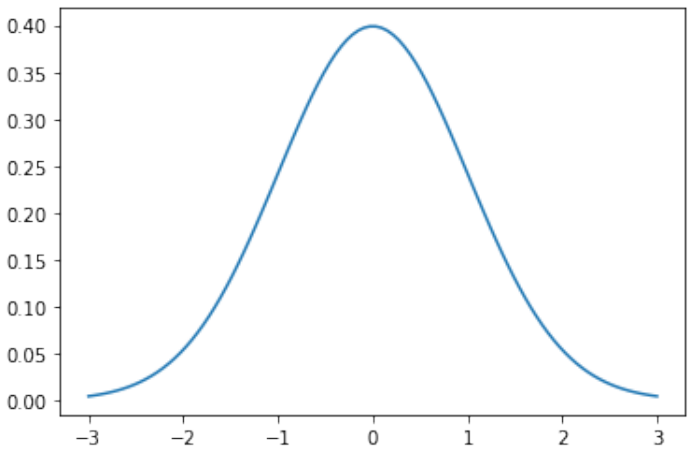


Figure 3.1: Normal distribution $F(x)$ of variable x, where $\mu = 0$ and $\sigma = 1$

Equation 3.11 in the simplified form of the mean $\mu = 0$ and the standard deviation $\sigma = 1$ is as follows:

$$F(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \tag{3.12}$$

The normality assumption is used in machine learning for:

- checking input data for modeling in the case of fitting models,
- model evaluation results in the case of model selection,
- residual errors from model predictions in the case of regression.

Normality test can be done by visual plotting such as histogram plot, distribution plot, and QQ plot (other names: Quantile-Quantile plot, Q-Q plot).

Large parts of the field of statistics are dedicated to methods for Gaussian (normal) distribution. Much of data in machine learning often fits a normal distribution. Normal distribution can be described using summary statistics with N is population size and n is sample size. Normality can be tested using Shapiro-Wilk test. It can be tested also by other ones, for example, Anderson-Darling test, which tests if a data sample comes from a specific distribution. The Anderson-Darling test is a modification of the Kolmogorov-Smirnov (KS) test and it gives more weight to the tails of the distribution than does the KS test.

Other distributions of normal distribution are, for example, uniform distribution, Bernoulli distribution, binomial distribution, Poisson distribution, exponential distribution, log normal distribution, Student's t distribution, Chi-squared distribution, Gamma and Beta distribution.

Shapiro-Wilk test

Shapiro-Wilk test checks if a sample $(x_1, x_2, \ldots, x_n)$ comes from a normal distribution. It first quantifies the similarity between the observed and normal distributions as a single quantity. After that, the test computes which percentage of the testing sample overlaps with the normal curve as a similarity percentage.

The hypotheses are:

- H_0: The sample comes from a normal distribution.
- H_1: The sample is not coming from a normal distribution.

Shapiro-Wilk test calculates W statistic (Equation 3.13) as follows:

$$W = \frac{\left(\sum_{i=1}^{n} a_i \, x_{(i)}\right)^2}{\sum_{i=1}^{n} \left(x_i - \overline{x}\right)^2} \tag{3.13}$$

where

n is the sample size;
$\overline{x}$ is the sample mean;
$x_{(i)}$ is the i^{th} order statistic, i.e., the i^{th} smallest number in the sample); $x_{(i)} \neq x_i$;
a_i are generated from means, variances and covariances of the order statistics.

$$(a_1, \cdots, a_n) = \frac{m^T \, V^{-1}}{C} \tag{3.14}$$

V is the covariance matrix of those normal order statistics.

$$C = \left\| V^{-1} m \right\| = (m^T \, V^{-1} \, V^{-1} \, m)^{\frac{1}{2}} \tag{3.15}$$

m_i are the expected values of the order statistics of independent and identically distributed (IID) random variables sampled from the standard normal distribution.

$$m = (m_1, m_2, \cdots, m_n)^T \tag{3.16}$$

The test has limitations. The most visible limitation is its bias in sample size n. The larger the sample, the more likely is the statistically significant result of the test. The minimum sample size is 20 and the maximum sample size is 5000, but since no distribution is exactly a normal distribution, especially for a very large sample.

Skewness and Kurtosis

Skewness is a measure of asymmetry[2] (Figure 3.2, Equation 3.17 of the third standardized moment).

$$skewness = E\left[\left(\frac{X - \mu}{\sigma}\right)^3\right] \tag{3.17}$$

where X is random variable, μ is the mean, σ is the standard deviation, and E is an expectation function.

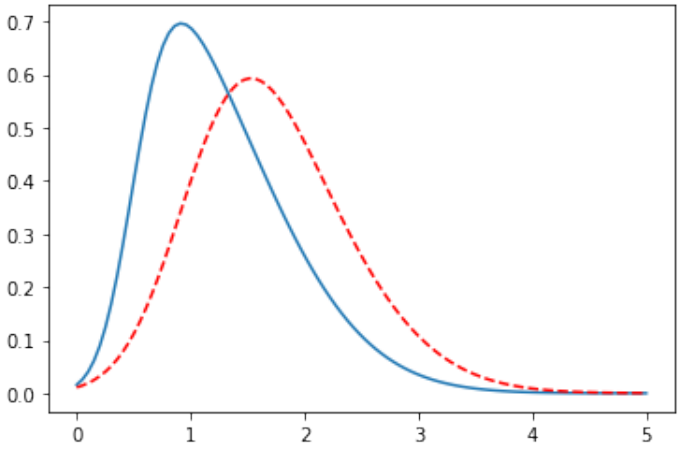


Figure 3.2: Skewness

For the univariate data sample $X = (x_1, x_2, \cdots, x_n)$, the formula for skewness is as follows:

[2] https://www.itl.nist.gov/div898/handbook/eda/section3/eda35b.htm

$$skewness = \frac{\sum_{i=1}^{n}(x_i - \bar{x})}{n\, s^3} \tag{3.18}$$

where $\bar{x}$ is the sample mean, s is the sample standard deviation, and n is the sample size.

The distribution of the dataset is symmetric if it looks the same to the left and right of the center point.

$skewness = 0$: normally distributed;
$skewness > 0$: more weight in the left tail of the distribution;
$skewness < 0$: more weights in the right tail of the distribution.

Many software libraries actually compute the adjusted skewness as follows:

$$skewness_{adjusted} = \frac{\sqrt{n\,(n-1)}}{n-2} \cdot \frac{\sum_{i=1}^{n}(x_i - \bar{x})}{n\, s^3} \tag{3.19}$$

Kurtosis is a measure of whether the data are peaked or flat[3] relative to a normal distribution (Figure 3.3, Equation 3.20 of the fourth standardized moment).

$$kurtosis = E\left[\left(\frac{X - \mu}{\sigma}\right)^4\right] \tag{3.20}$$

where X is random variable, μ is the mean, σ is the standard deviation, and E is an expectation function.

For the univariate data sample $X = (x_1, x_2, \cdots, x_n)$, the formula for kurtosis is as follows:

$$kurtosis = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^4}{n\, s^4} \tag{3.21}$$

where $\bar{x}$ is the sample mean, s is the sample standard deviation, and n is the sample size.

Datasets with high kurtosis tend to have a distinct peak near the mean, decline rather rapidly, and have heavy tails. Data sets with low kurtosis tend to have a flat top near the mean rather than a sharp peak.

For a symmetrical normal distribution, the *mean* is equal to the *median*. For a right skew, the *mean* is greater than the *median*. For a left skew, the *mean* is less than the *median*.

[3] https://www.itl.nist.gov/div898/handbook/eda/section3/eda35b.htm

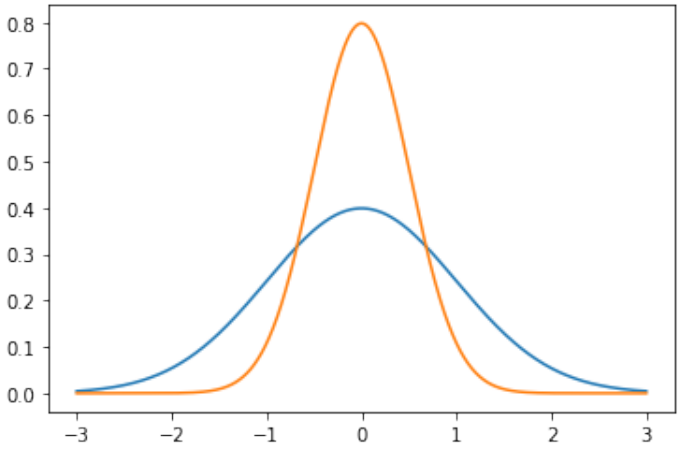


Figure 3.3: Kurtosis

3.3 Statistics and machine learning

Statistics is considered as a prerequisite to applied machine learning. It can be divided into descriptive statistics and inferential statistics.

Descriptive statistics, which summarizes raw observations into information. The aim is to understand and share information. Descriptive methods apply to univariate variables to know their shape, center, spread, and relative position. Descriptive method applies to bivariate and multivariate relations for correlation and regression.

Inferential statistics, which aids in quantifying the properties of the domain or population from a smaller set of observations called a sample. Inferential methods can be grouped into Central Limit Theorem, binomial theorem, hypothesis and significance testing, around normal distribution and means.

In general, statistics and machine learning are very closely related fields [18], [19]. The line between them can be very fuzzy. Machine learning is a subfield of Computer Science and Artificial Intelligence, it deals with building systems including algorithms and models that can learn from data and observations instead of explicitly programmed instructions (e.g., rules). Statistics is a subfield of mathematics. Statistical modeling is a formalization of relationships between variables in the data in the form of mathematical equations. Statistics in machine learning deals with:

- problem framing (EDA, data mining),
- data understanding (summary statistics, data visualizations),
- data cleaning (data corruption, errors, losts, outlier detection,imputation),
- data selection (data sampling, feature selection),
- data preparation (scaling, encoding) and data transformation,
- model evaluation (experimental design, splitting methods),

- model configuration and section (statistical hypothesis testing, estimation statistics), model presentation and model predictions (estimation statistics).

In this context, data *population* refers to all possible observations from the investigation domain; data *sample* refers to a randomly selected set of observations from the population.

3.4 Statistical inference

Statistical inference is the process of drawing conclusions about an underlying population based on a sample or a subset of the data. In many cases, it is not practical to obtain all measurements in a given population. Inferential statistical analysis infers the properties of a population by testing hypotheses and deriving estimates. It is assumed that the observed data set is sampled from a larger population [19]. Inferential statistics is different from descriptive statistics (Section 3.2), which is solely concerned with properties of the observed data, and does not rely on the assumption that the data come from a larger population.

3.4.1 Statistical significance testing

A test statistic is a quantity calculated by a statistical test in a standardized way across libraries. Different test statistics are used in different statistical tests. It describes how far the observed data is from the null hypothesis (H_0) or no relationship between variables or no difference between sample groups. It tells how different two or more groups are from the overall population mean. The chance of rejecting H_0 when it is true is close to 5% regardless of sample size. If the number of machine learning algorithms is high, the number of already implemented statistical tests is very high too.

> The main assumptions of statistical tests are data are normally distributed, comparing samples have similar variance, and data are independent.

Parametric tests assume underlying statistical distributions in the data. Several conditions of validity must be met so that the result of a parametric test is reliable. An example is the Student's t-test for two independent samples is reliable only if each sample follows a normal distribution and sample variances are homogeneous.

If the data does not meet the normality assumptions, non-parametric statistical tests can be used, which require fewer parameters but provide weaker inferences. Non-parametric tests do not rely on any distribution. They can be applied even if parametric conditions of validity are not met. Parametric tests often have non-parametric equivalents. Non-parametric tests are more robust, but parametric may have more statistical power.

The number of existing statistical tests in various implementations is high. It is important to know which tests can be used and for what purpose. Statistical tests usually have prerequisites that must be fulfilled. Some of them are accurate only in a certain range. The difference in observations may occur, then statistics provide statistical tools to decide

whether the observed differences are significant. However, statistics does not say whether the found difference is practical, because it always depends on the specific problem. Examples of frequently used significance tests for hypothesis testing are:

- Student's T-test: parametric test for two samples,
- Analysis of Variance (ANOVA): parametric test for more than two samples,
- Mann-Whitney U Test: non-parametric test for two samples,
- Kruskal-Wallis H Test: non-parametric test for more than two samples.

Student's T-test (or T-test)

The t score is the ratio between the difference between two groups and the difference within the groups. The larger the t score, the more difference there is between groups. The smaller the t score, the more similarity there is between groups.

The assumptions:

- The data must be randomly sampled from the population of interest.
- The data variables follow a normal distribution.

The hypotheses:

- H_0: the means of two groups are equal ($\mu_1 = \mu_2$).
- H_1: the means are not equal ($\mu_1 \neq \mu_2$).

and meaning

- Fail to reject H_0: no difference between the sample means;
- Reject H_0: some difference between the sample means

The t statistic to test whether the means are different is calculated as in Equation 3.22:

$$t = \frac{\mu_1 - \mu_2}{s_p \sqrt{\frac{1}{n_1} - \frac{1}{n_2}}} \tag{3.22}$$

where

$$s_p = \sqrt{\frac{(n_1 - 1)s_{X_1}^2 + (n_2 - 1)s_{X_2}^2}{n_1 + n_2 - 2}} \tag{3.23}$$

and

n_j is the sample size in the j^{th} group,
μ_j is the sample mean of the j^{th} group,
s_p is the pooled standard deviation for $n = n1 = n2$
$s_{X_1}^2$ and $s_{X_2}^2$ are the unbiased estimators of the variances of the two samples.
the denominator of t is the standard error of the difference between two means.

Analysis of Variance (ANOVA)

The Analysis of Variance (ANOVA) test has the same purpose as the Student's T-test but for more data samples[4].

The assumptions:

- The population from which samples are drawn should be normally distributed.
- The sample cases should be independent of each other.
- The variance among the samples should be approximately equal.

The hypotheses:

- H_0: the means across k groups (samples) are equal ($\mu_1 = \mu_2 = \cdots = \mu_k$).
- H_1: the means are not all equal.

and meaning

- Fail to reject H_0: all sample distributions are equal;
- Reject H_0: one or more sample distributions are not equal.

The F statistic to test whether the means are different is calculated as in Equation 3.24.

$$F = \frac{\sum_{i=1}^{k} \frac{\mu_i - \mu}{k-1}}{\sum_{i=1}^{k}\sum_{j=1}^{n_j} \frac{(x_{ij} - \mu_i)^2}{N-k}} = \frac{between\ group\ variability}{within\ group\ variability} \tag{3.24}$$

where

k is the number of independent samples,
n_j is the size of the j^{th} sample,
x_{ij} is individual observation,
μ_i is the mean of the i^{th} sample,
μ is the overall sample mean,
N is the total sum of all sample sizes.

Mann-Whitney U Test (or U Test)

Mann-Whitney U Test is similar to T-test for two data samples, but it is a nonparametric test for at least 20 observations in each sample.

The assumptions:

- Two samples must be independent and observations must be independent.
- The dependent variable should be ordinal or continuous.
- Samples are not normally distributed.

The hypotheses:

[4] https://www.statsdirect.com/help/analysis_of_variance/one_way.htm

- H_0: the distributions of both samples are equal.
- H_1: the distributions are not equal.

and meaning

- Fail to reject H_0: sample distributions are equal;
- Reject H_0: sample distributions are not equal.

The test statistic U is the smaller of U_1 (Equation 3.25) and U_2 (Equation 3.26):

$$U_1 = n_1 n_2 + \frac{n_1(n_1+1)}{2} - R_1 \tag{3.25}$$

$$U_2 = n_1 n_2 + \frac{n_2(n_2+1)}{2} - R_2 \tag{3.26}$$

where

n_1 is the size of the first sample,
n_2 is the size of the second sample,
R_1 is the sum of the ranks for the first sample,
R_2 is the sum of the ranks for the second sample.

Kruskal-Wallis H Test (or H Test)

The Kruskal-Wallis H Test is the nonparametric version of the one-way ANOVA. This test can be used to determine whether more than two independent samples have a different distribution. It can be thought of as the generalization of the Mann-Whitney U Test for more data samples[5].

The assumptions:

- Samples must be independent and observations must be independent.
- The dependent variable should be ordinal or continuous.
- Samples are not normally distributed.
- All samples should have the same shape distributions.

The hypotheses

- H_0: all data samples were drawn from the same distribution;
- H_1: the distributions are not equal.

and meaning

- Fail to reject H_0: all sample distributions are equal;
- Reject H_0: One or more sample distributions are not equal.

[5] https://www.spss-tutorials.com/kruskal-wallis-test/

The test statistic H is defined as:

$$H = \frac{12}{N(N+1)} \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3(N+1) \tag{3.27}$$

where

N is the total sample size;
n_i is the sample size i^{th} sample;
k is the number of samples;
R_i is the rank sum for i^{th} sample.

3.4.2 Basic steps of hypothesis testing

1. Determine the null hypothesis (H_0) and alternate hypothesis (H_1)
2. Set a significance level α
3. Compute the p_{value} using a suitable test statistic T
4. Make a decision
   - IF $p_{value} \leq \alpha$ (significance level)
     THEN the result is statistically significant
     THEN reject H_0 and accept H_1
   - IF $p_{value} > \alpha$ THEN fail to reject H_0 and reject H_1.

The test statistic T is simply the corresponding standardized score computed assuming the null hypothesis (H_0) is true. p_{value} is found from a table of percentiles for standardized scores p_{value} is the evidence against a H_0. The smaller the p_{value}, the stronger the evidence that you should reject the H_0.

- if $p_{value} \leq \alpha$ then significant result, then reject H_0 and accept H_1
- if $p_{value} > \alpha$ then not significant result, then fail to reject H_0 and reject H_1

Results of statistical tests can be divided into

- Type I Error (α): the incorrect decision of rejecting a true H_0 (False Positive).
- Type II Error (β): the incorrect decision of *do not reject* a false H_0 (False Negative).

Examples of hypotheses, which are often used in machine learning:

- Testing that assumes that the data has a normal distribution.
- Testing that assumes that the two samples were drawn from the same underlying population distribution.

3.4.3 Statistical power analysis

What affects statistical power? (BEAN)

1. B: Statistical Power = $1 - \beta$
   - Statistical Power is the probability of a True Positive (TP) result
2. E: Effect Size
   - Cohen's d for the difference between groups.

- Pearson Correlation Coefficient (PCC) for the relationship between variables

3. A: Alpha Significance (α)
   - The significance level (α) is often set to 5% or 0.05
4. N: Sample Size (n) is the number of observations in the sample.

Pearson's correlation

Pearson Correlation Coefficient (PCC) between two samples x, y of the size n is calculated as follows:

$$r_{xy} = \frac{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)\left(y_i - \bar{y}\right)}{\sqrt{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2}\sqrt{\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2}} \tag{3.28}$$

PCC has values in the range $[-1, 1]$. Positive correlation means both variables change in the same direction. Neutral correlation means no relationship in the change of the variables. Negative correlation means variables change in opposite directions. It is also known under other names such as Matthews Correlation Coefficient (MCC) or ϕ (phi) coefficient.

Correlation and covariance

For the two quantitative variables, the basic statistics of interest are the sample covariance and sample correlation. The sample covariance is a measure of how much two variables "co-vary", i.e., how much (and in what direction) should we expect one variable to change when the other changes. Covariance provides the a measure of strength of correlation between two variables or more sets of variables. The general formula for sample covariance is as in Equation 3.29:

$$cov(X, Y) = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{n - 1} \tag{3.29}$$

It is worth to mention that

$$cov(X, X) = var(X) \tag{3.30}$$

In the case of more variables, the covariance matrix element C_{ij} is the covariance of x_i and x_j. The element C_{ii} is the variance of x_i. Covariance tends to be hard to interpret, so correlation is often used instead.

If $cov(x_i, x_j) = 0$ then the variables are uncorrelated.
If $cov(x_i, x_j) > 0$ then the variables are positively correlated.
If $cov(x_i, x_j) < 0$ then the variables are negatively correlated.

Here is suitable to mention:

> Correlation does not imply causation !

Correlation is a statistical technique which tells us how strongly a pair of variables are linearly related and change together. Causation takes a step further than correlation. It says any change in the value of one variable will cause a change in the value of another variable, which means one variable makes the other to happen.

Effect size by Cohen's d

Cohen's d measures the difference between the mean of two Gaussian-distributed variables X_1, X_2 of sizes n_1, n_2 and means μ_1, μ_2 as in Equation 3.32.

$$d = \frac{\mu_1 - \mu_2}{s} \tag{3.31}$$

using pooled standard deviation s and variance s^2

$$s = \sqrt{\frac{(n_1 - 1)s_{X_1}^2 + (n_2 - 1)s_{X_2}^2}{n_1 + n_2 - 2}} \tag{3.32}$$

The score is standardized, the interpretation of the result is summarized as below:

Small effect:	$d = 0.20$
Medium effect:	$d = 0.50$
Large effect:	$d = 0.80$

p_{value} can inform whether an effect exists, it does not reveal the size of the effect. An effect size is a measure of the strength of the relationship between two variables in a dataset. This is particularly useful when comparing the change for two different kinds of measures. If the value of Cohen's d is negative, this means that there was no improvement, e.g., the posttest results were lower than the pretests.

Confidence interval and confidence levels

Confidence interval is a range of values that is likely to include a population value with a certain degree of confidence. Confidence interval (formula) is calculated as follows:

If $(n \geq 30)$ then the confidence interval is:

$$\mu \pm z\frac{\sigma}{\sqrt{n}} \tag{3.33}$$

If $(n < 30)$ then the confidence interval is:

$$\mu \pm t\frac{\sigma}{\sqrt{n}} \tag{3.34}$$

where n is sample size, μ is mean, σ is standard deviation.

The z-value for 95% confidence is $Z = 1.96$

z-table `https://www.statisticshowto.com/tables/z-table/`
t-table `https://www.statisticshowto.com/tables/t-distribution-table/`

The degrees of freedom in a sample is:

$$df = n - 1 \tag{3.35}$$

Confidence levels indicate the probability with which the estimation of the location of a statistical parameter (e.g., mean) in a sample survey is also true for the population. Confidence level is expressed as a percentage, for example, a 95% confidence level.

One important thing about the confidence level is that it has to be defined prior to examining the data. Most commonly, the 95% confidence level is used.

Notes on statistical testing

The formulation *Failed to reject the null hypothesis (H_0)* does not mean that we accept it. It is quite possible that the test was not performed with sufficient force to reject.

Repeated testing on the same data increases the probability of the Type I Error - False Positives (FP)

$$\alpha^* = 1 - (1 - \alpha)^k \tag{3.36}$$

where α is the level of each test and k is the number of interim looks.

Larger samples allow to increase the level of significance of the finding, because with a larger sample, the probability of a result is likely to increase. This effect is expected because the larger the sample size, the more accurately it is expected to reflect the behavior of the entire population.

3.5 Probability, information theory and entropy

Quantifying the amount of information requires the use of probabilities, hence the relationship of information theory to probability. The foundation concept of information is *the quantification of the amount of information in things* like events, random variables, and distributions [20].

$$information(x) = h(x) = -log\ p(x) \tag{3.37}$$

where log is the base-2 logarithm and $p(x)$ is the probability of the event x. The unit of information depends on the base of the logarithm, which can be common (log_{10} resp. log),

natural (log_e resp. ln) or base 2 (log_2 resp. lg) logarithm. It can be in *Shannon's bits*, *nats* or *hartleys*.

> If the base is 2, which is the most used in Computer Science, then the information unit is *shannon* equal to the information content of one *bit*.

In this work, we denote log as the most used form of logarithm, i.e., with the base 2 or generic base as in complexity formulas.

Entropy

Entropy H can be calculated for a random variable X in C discrete states as:

$$Entropy = H(X) = -\sum_{i=1}^{C} p_i \, log \, p_i \tag{3.38}$$

The entropy of a probability distribution can be interpreted as a measure of uncertainty associated with a random variable drawn from a given distribution.

Similar to entropy, Gini coefficient (other names: Gini ratio, Gini index, Gini impurity) is calculated as in Equation 3.39. It measures the inequality among the values of a frequency distribution.

$$Gini = 1 - \sum_{i=1}^{C} p_i^2 \tag{3.39}$$

A Gini coefficient of zero (or 0%) expresses a perfect equality, where all values are the same. A Gini coefficient of one (or 100%) expresses the maximal inequality among values. The computation for Gini coefficient is faster than the computation for entropy because it does not contain *logarithm*.

■ **Example 3.3** A variable (sample) S consists of 14 observations, where 9 observations belong to class c_1 and 5 observations belong to class c_2.

$$H(S) = -\frac{9}{14} \, log_2\Big(\frac{9}{14}\Big) - \frac{5}{14} \, log_2\Big(\frac{5}{14}\Big) \approx 0.940 \tag{3.40}$$

$$Gini(S) = 1 - \left(\Big(\frac{9}{14}\Big)^2 + \Big(\frac{5}{14}\Big)^2\right) \approx 0.459 \tag{3.41}$$

■

The *cross entropy* between distributions p and q is defined by:

$$H(p,q) = -\sum_{i=1}^{C} p_i \, log \, q_i \tag{3.42}$$

The cross entropy is the expected number of bits needed to compress some data samples drawn from distribution p using a code based on distribution q.

The *joint entropy* of two random variables X and Y is defined as

$$H(X,Y) = -\sum_{x,y} p(x,y) \, log \, p(x,y) \tag{3.43}$$

If X and Y are independent, then

$$H(X,Y) = H(X) + H(Y) \tag{3.44}$$

The *conditional entropy* quantifies the amount of information needed to describe the outcome of a random variable Y given that the value of another random variable X is known.

$$H(Y|X) = -\sum_{x,y} p(x,y) \, log \, p(y|x) \tag{3.45}$$

$$= -\sum_{x,y} p(x,y) \, log \, \frac{p(x,y)}{p(x)} \tag{3.46}$$

$$= -\sum_{x,y} p(x,y) \, log \, p(x,y) + \sum_{x} p(x) \, log \, \frac{1}{p(x)} \tag{3.47}$$

$$H(Y|X) = H(X,Y) - H(X) \tag{3.48}$$

If Y is a deterministic function of X, then knowing X determines Y and $H(Y|X) = 0$.
If X and Y are independent, knowing X tells nothing about Y and $H(Y|X) = H(Y)$.

For a variable X with continuous distribution $p(x)$, the entropy $H(X)$ is calculated as

$$H(X) = -\int dx \, p(x) \, log \, p(x) \tag{3.49}$$

Mutual Information (MI) is used also to test non-linear feature correlations for variables with continuous distribution.The MI of the two random variables, X and Y is a measure of the mutual dependence between them.

$$I(X,Y) = -\int\int dx \, dy \, p(x,y) \, log \frac{p(x,y)}{p_x(x) \, p_y(y)} \tag{3.50}$$

In the case of variables with continuous value distribution, it is hard to calculate MI from

real values (Equation 3.50). Therefore, MI estimation relies on nonparametric methods based on entropy estimation from κ-nearest neighbors distances to avoid the binning approach [21], [22]. Small values of κ reduce general systematic errors, while large κ leads to smaller statistical errors.

In probability theory and information theory, the Mutual Information (MI) of two random variables is a measure of the mutual dependence between two variables. It quantifies the *amount of information* in units such as *Shannon's bits, nats or hartleys*. The concept of MI is intimately linked to *entropy* of a random variable. MI is calculated between two variables and measures the reduction in uncertainty for one variable given a known value of the other variable.

> Mutual Information (MI) and Information Gain (IG) are calculate the same thing, although the context or usage of the measure often gives different names.

Information Gain (IG)

Information Gain (IG) comes from information theory, which is a field of study concerned with quantifying information for communication (Claude Shannon). The expected IG is the reduction in information entropy H from a prior state to a state that takes some information as given:

$$IG(S, A) = H(S) - H(S|A) \tag{3.51}$$

where S is a set of training examples and $H(S|A)$ is the conditional entropy of S given the attribute A.

IG measures the reduction in entropy or surprise by splitting a dataset according to a given value of a random variable. IG is equal to zero if and only if two random variables are independent. Higher values mean higher dependency [3], [23]. Equation 3.51 can be written also as follows:

$$IG(S, A) = H(S) - \sum_{a_i}^{A} \frac{|S_{a_i}|}{|S|} H(S_{a_i}) \tag{3.52}$$

where

S is a set of training examples,
$|S|$ is the number of observations of S,
A is an attribute of S, A has a set of values $\{a_1, a_2, ...a_k\}$,
S_{a_i} is the subset of S where observations have the attribute value a_i,
$|S_{a_i}|$ is the number of observations of S_{a_i}.

■ **Example 3.4** A sample S consists of 14 observations, where 9 observations belong to class c_1 and 5 observations belong to class c_2. The attribute A of S has two values $\{a_1, a_2\}$. For the value a_1, 6 observations belong to class c_1 and 2 observations belong to class c_2.

For the value a_2, 3 observations belong to class c_1 and 3 observations belong to class c_2.

$$H(S) = -\frac{9}{14}\, log_2\Big(\frac{9}{14}\Big) - \frac{5}{14}\, log_2\Big(\frac{5}{14}\Big) \approx 0.940 \tag{3.53}$$

$$H(S_{a_1}) = -\frac{6}{8}\, log_2\Big(\frac{6}{8}\Big) - \frac{2}{8}\, log_2\Big(\frac{2}{8}\Big) \approx 0.811 \tag{3.54}$$

$$H(S_{a_2}) = -\frac{3}{6}\, log_2\Big(\frac{3}{6}\Big) - \frac{3}{6}\, log_2\Big(\frac{3}{6}\Big) = 1.000 \tag{3.55}$$

$$IG(S,A) = H(S) - \frac{8}{14}\, H(S_{a_1}) - \frac{6}{14}\, H(S_{a_2}) \tag{3.56}$$

$$= 0.940 - \frac{8}{14}\, 0.811 - \frac{6}{14}\, 1.000 \approx 0.048 \tag{3.57}$$

■

Likelihood and log-likelihood

In statistics, probability is used to find the chance of occurrence of a particular situation, whereas likelihood[6] is used to generally maximize the chances of a particular situation to occur. Many probability distributions have unknown parameters, which can be estimated using sample data. The *likelihood function (LF)* provides an idea of how well the data summarizes these parameters (Equation 3.58).

$$LF = \prod_{n=1}^{N} f\Big(y_n | y_{n-1}, y_{n-2}, \cdots, y_1, \theta_1, \theta_2, \cdots, \theta_k\Big) \tag{3.58}$$

In many cases, it is more convenient to work with the *natural* logarithm (logarithm with the base e denoted as ln) of the likelihood function than with the likelihood function itself. The natural logarithm of the likelihood function (Equation 3.59) is called the *log-likelihood function (LLF)*.

$$LLF = ln(LF) = \prod_{n=1}^{N} ln\left(f\Big(y_n | y_{n-1}, y_{n-2}, \cdots, y_1, \theta_1, \theta_2, \cdots, \theta_k\Big)\right) \tag{3.59}$$

where

LF is the likelihood function;
LLF is the log-likelihood function;
f is the conditional probability density function;
y_n is the value of the series at the point n;
$y_{n-1}, y_{n-2}, \cdots, y_1$ are previous values of the series before the point n;
$\theta_1, \theta_2, \cdots, \theta_k$ are the parameters of the statistical model.

[6] Likelihoods are a key part of Bayesian inference.

Maximum Likelihood Estimation (MLE) is a method of statistical inference. It estimates the parameters of an assumed probability distribution, given some observed data by maximizing a likelihood function. The statistical model with the MLE score is considered as a better model.

■ **Example 3.5** Let y be the actual (true) output $y = [1,\ 1,\ 0]$ and p be the predicted probability of y_i is 1; $p = [0.8,\ 0.4,\ 0.1]$. Likelihood calculated according to Equation 3.58 and log-likelihood calculated according to Equation 3.59 are:

$$likelihood = (0.8) \cdot (0.4) \cdot (1 - 0.1) = 0.288 \tag{3.60}$$

$$log\text{-}likelihood = ln(0.8) \cdot ln(0.4) \cdot ln(1 - 0.1) \approx -0.0215 \tag{3.61}$$

■

Perplexity and cross entropy

The concept of *entropy* is widely used in information theory as well as in machine learning. In the context of entropy and probability, *perplexity* or predictive likelihood is often interpreted as a measure of predictability. The perplexity of a discrete probability distribution p of N discrete states is defined as:

$$perplexity(x) = 2^{H(p)} \tag{3.62}$$

where

$$H(p) = -\frac{1}{N}\sum_{i=1}^{N} p(x_i)\ log_2\ p(x_i) \tag{3.63}$$

For any two distributions p and q, the *cross entropy* $H(p, q)$ is defined as

$$H(p, q) = -\frac{1}{N}\sum_{i=1}^{N} p(x_i)\ log_2\ q(x_i) \tag{3.64}$$

A language model is a probability distribution over sentences. Perplexity is often used to evaluate the quality of statistical language models. The lower the score, the better the model for the given data. The best language model is the one that best predicts an unseen test set. If the probability distribution of a sentence following a sequence of sentences as p, the probabilistic model is denoted as m then the cross entropy $H(p, m)$ is defined as:

$$H(p, m) = -\frac{1}{N}\sum_{i=1}^{N} p(x_i)\ log_2\ m(x_i) \tag{3.65}$$

The perplexity of the model is the exponential of its cross entropy:

$$perplexity(m) = 2^{H(p,m)} \tag{3.66}$$
$$= 2^{-\frac{1}{N}\sum_{i=1}^{N} p(x_i)\ log_2\ m(x_i)} \tag{3.67}$$

■ **Example 3.6** Let assume the data D is a single long document $\{x_1, x_2, \cdots, x_N\}$ of N words and all words have the same frequency $p(x_i) = \frac{1}{N}$. Let q be the proposed probability language model built based on D.

$$perplexity(D) = 2^{H(p,\ q)} \tag{3.68}$$
$$= 2^{-\sum_{i=1}^{N} p(x_i)\ log_2\ q(x_i)} \tag{3.69}$$
$$= 2^{-\sum_{i=1}^{N} \frac{1}{N}\ log_2\ q(x_i)} \tag{3.70}$$
$$= \prod_{i=1}^{N} \left(\frac{1}{q(x_i)}\right)^{\frac{1}{N}} = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{q(x_i)}} \tag{3.71}$$

$$perplexity(D) = \sqrt[N]{\frac{1}{q(x_1)q(x_2)\cdots q(x_N)}} \tag{3.72}$$

■

3.6 Software available

`Python` is a programming language created by Guido van Rossum and first released in 1991 [12]. The rationale for using Python for Data Science, machine learning as well as EDA is because Python is a general purpose programming language for research, development, and production at small and large scales. Python features a dynamic system and automatic memory management, with large and comprehensive libraries for scientific computation and data analysis.

`SciPy` is an open-source Python library used for scientific computing and technical computing [24]. SciPy builds on the NumPy array object and is part of the NumPy stack which includes tools like Matplotlib, Pandas, and SymPy.

`NumPy` is the fundamental package for scientific computing with Python [25]. Besides its obvious scientific uses, NumPy can also be used as an efficient multidimensional container of generic data. NumPy stack has similar users to MatLab, GNU Octave, and SciLab.

`StatsModels` is a Python module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration [26]. An extensive list of result statistics are available for each estimator.

`Pandas` is a Python package providing fast, flexible, and expressive data structures designed to make it easier to work with relational or labelled data [27]. Its two primary data structures, Series (one-dimensional) and DataFrame (two-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering.

`MatplotLib` is a comprehensive library for creating static, animated, and interactive visualizations in Python [28]. It is the base for `seaborn` [29], which is a Python data visualization library with a high-level interface for drawing attractive and informative statistical graphics.

`R` is a free software environment for statistical computing and graphics, including linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering. It compiles and runs on a wide variety of UNIX platforms, Windows, and MacOS. R is easy to use and extensible via packages. The comprehensive R archive network (CRAN) offers more than 10 thousands packages, and the list is getting longer.

The difference between Python and R is largely philosophical. Python is a general-purpose language designed by programmers for programmers; R was built for statistical analysis. Trends (such as Google and KDD trend) shows that R was slightly ahead in 2014 and 2015 in Data Science.

R Since 2016, Python is the most popular programming language in Data Science.

The number of frameworks and libraries providing or using intelligent techniques is high. `MatLab` is a multi-paradigm numerical computing environment. MatLab is quite popular with over 2 million users across industry and academia. On the other hand, MatLab is a proprietary product of MathWorks, so users are subject to vendor lock-in and future development will be tied to the MatLab language. The two most popular free alternatives to MatLab are GNU `Octave` and `SciLab`.

`Statistical Analysis System (SAS)` currently is a proprietary software package written in C for advanced data analytics and business intelligence with more than 200 components. Another similar proprietary software package is `Statistical Package for the Social Sciences (SPSS)`. An open-source alternative of SPSS is GNU `PSPP`.

3.7 Remark

R George Casella et al. *Statistical inference*. Cengage Learning, 2002. ISBN: 0-534-24312-6

4. Data preprocessing for modeling

Today, 5Vs (Volume, Velocity, Variety, Veracity and Value) are the five keys to making Big Data a huge business. The aim of the data transformation is to have a data product, which works efficiently within the organization and provides a new generation of services to support the decision making and decision process.

- *Volume* is the base of Big Data, the volume doubles about every 40 months;
- *Velocity* can be more important than Volume in many cases. It is better to have limited data in real time than lots of data at a low speed. Data have to be available at the right time to make appropriate decisions.
- *Variety* describes various sources of data. Data can come from many different sources, from in-house devices to smartphones, GPS technology, or from social networks. It can have many layers and many formats.
- *Veracity*, which in the context of this work, is equivalent to quality of machine learning data. We have all the data, but could we be missing something? Are the data "clean" and accurate? Do they really have something to offer? Machine learning algorithms learn from machine learning data.
- *Value* refers to the ability to transform a tsunami of data into business.

Data processing for modeling (other names: data preparation, data preprocessing) is the process of transforming raw data into a form that is appropriate for modeling using machine learning algorithms [30], [17]. Data preparation must be prepared to avoid data leakage, which is a huge problem in modeling in general. Leakage refers to information that helps to predict something, and the fact that the use of this information to predict is unfair. A careful application of data preparation techniques is required to avoid data leakage, and this varies depending on the model evaluation scheme used, such as *train-test splits* or *k-fold* cross-validation.

4.1 V as Veracity

Data preprocessing for machine learning faces many problems. Each Data Science (machine learning) project is different because the specific data at the core of the project is different. Since data and models are so diverse, it's difficult to generalize the practice of feature engineering across projects. The philosophy of data preparation is to discover how to best expose the underlying structure of the problem to the learning algorithms.

Figure 4.1: V as Veracity stands for quality data for machine learning model training.

Data preprocessing is important because of the presence of unformatted real-world data.

- Inaccurate data (missing data): data is not continuously collected, a mistake in data entry, technical problems with biometrics.
- Noisy data (erroneous data and outliers): the technological problem of gadgets that gather data, a human mistake during data entry. Raw data contains statistical noise, errors, and conflicting examples;
- Inconsistent data: existence of duplication data, human data entry, containing mistakes in codes or names, violation of data constraints.

Problems with data can be listed furthermore as follows:

- Not enough data to capture relationships between input and output;
- Too much data for the specific quest or too complex data;
- Missing data: Not A Number (NaN), empty cells, and many more ...

Usually in machine learning tasks, raw data cannot be used directly. Machine learning algorithms usually require input data to be numbers. Machine learning model performance depends on machine learning data. Statistical noise and errors in the data may need to be corrected and complex nonlinear relationships have to be teased out. Other tasks of data preparation are data integration and data transformation, which integrates data from multiple sources such as databases, data cubes, and files. Data transformation can be composed of data normalization (scaling to a specific range), data aggregation, and data reduction (to obtain reduced representation in Volume but produces similar analytical results). Next steps can be data discretization, dimensionality reduction, or feature engineering as a part of data preparation for machine learning.

To avoid data leakage, data preparation must be fit on the training dataset only. Once fit, the data preparation pipeline can then be applied to the test dataset.

1. Split data into train and test datasets.
2. Fit data preparation on training dataset to get data preparation pipeline.
3. Apply data preparation pipeline to train and test datasets.
4. Evaluate models.

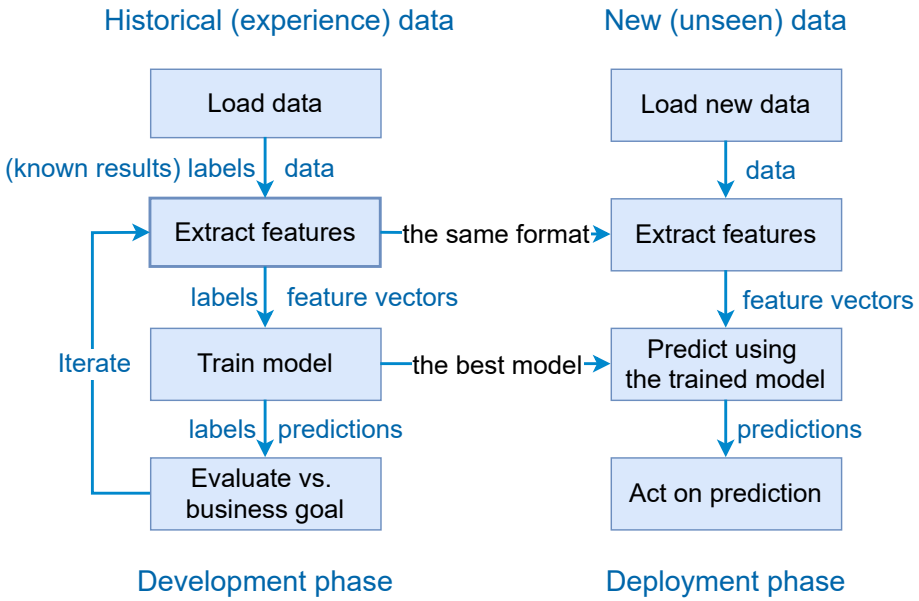


Figure 4.2: Simplified machine learning workflow for supervised learning.

More generally:

> The entire modeling pipeline must be prepared only on the training dataset to avoid data leakage.

This might include data transformation for data modelling as well as data modelling itself, but also other techniques such feature selection, dimensionality reduction, feature engineering, and many more.

4.2 Data preparation

Data preparation[1] can be repeated as many times as it is necessary. It can consume up to 80% of the entire time of the project. Data preparation can be divided into data transformation and feature engineering:

Data transformation (or data mugging): the aim of this step is to prepare high-quality data for the machine learning process in order to train high-quality models. It covers a lot of problems such as how to clean the data, how to complete the data, discovery of missing elements. This step can not be properly realized without knowledge gained from the Business and Data Understanding [6]. Certain specific sequence of tasks done in this step can be used to build the data pipeline in the Deployment phase. Frequent tasks are:

- Data transformation deals with merging, ordering, reshaping datasets;
- Data cleaning deals with missing or de-noising data;

[1] http://insidebigdata.com/2014/06/05/data-munging-exploratory-data-analysis-feature-engineering/

- Data sampling selects a representative subset from a large population of data;
- Data scaling deals with adding, removing and scaling variables;
- Data manipulation manipulates raw data to produce a single input;
- Data normalization organizes data for more efficient access;
- Feature engineering creates features that make machine learning algorithms work;
- Feature extraction, which is a part of feature engineering, pulls out specified data that is significant in some particular context.

Feature engineering uses domain knowledge of the data and EDA to create features that make machine learning algorithms work. It also determines which variables contribute the most to the predictive power of a machine learning methods. There are many available methods for this task to choose from like Information Gain (IG), Pearson Correlation Coefficient (PCC), forward selection and backward elimination.

In many cases, the process of feature engineering is labeled as *"an art as a science"*, which relies on hard experienced work in EDA in combination with intuition and knowledge of data and Data Science process. Detailed domain documentation and strong collaboration with domain experts can improve the process significantly.

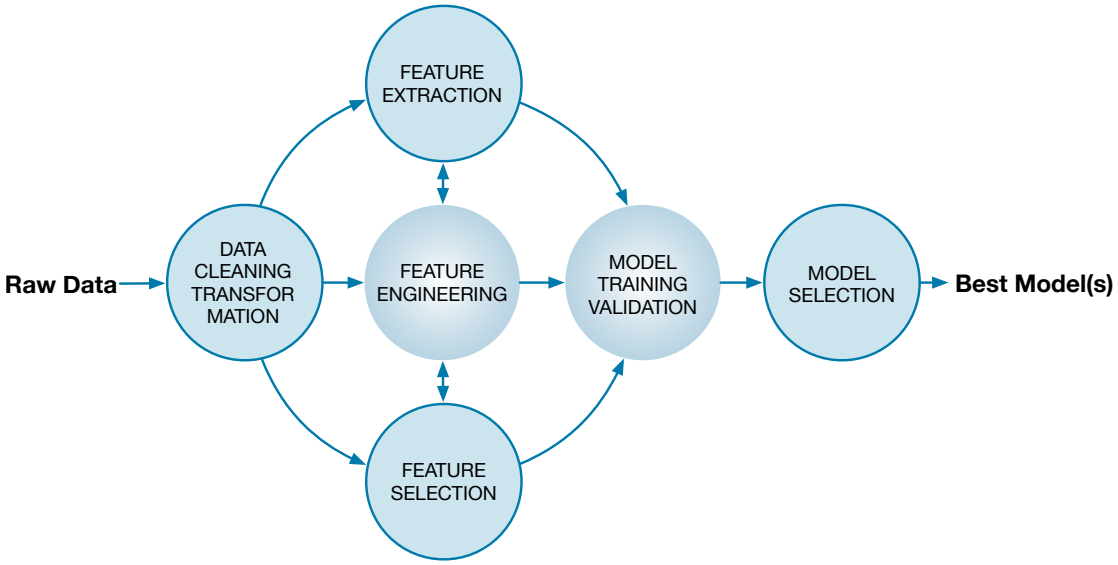


Figure 4.3: Data preprocessing for machine learning.

Data preparation can also be divided by another way into: data extraction, data integration from various resources, data interpolation, where the aim is to provide data in requested ranges with the specified granularity, and data reduction, which can lead to modeling simplification by removing irrelevant variables, and can increase the quality of machine learning model(s). Data transformation can be realized in many cases as Extract, Transform, and Load (ETL) process. Both Feature Engineering and EDA can deal with large-scale matrix operations.

4.3 Data cleaning

Predictive modeling is mostly data preparation. Preliminary tasks of data preparation are

- Linear column removing
  - Identify and remove columns that contain a single value.
  - Consider columns that have very few values.
  - Identify and remove rows that contain duplicate data.
  - Identify and remove columns that have a low variance.
- Missing data imputation: It is important to understand that there is no perfect way to deal with missing data, especially if the number of missing values in your data is big enough above 5%.

4.3.1 Ourlier detection

An outlier is an observation that is unlike the other observations. They are rare, distinct, or do not fit in some way. Outliers can be shown as: measurement of input errors, data corruptions, and true outlier observations. Identifying outliers and bad data is one of the most difficult parts of data cleanup. It can take a long time to get right and it has to be done cautiously. Outlier detection can be done as

- Outlier detection by standard deviation: 3x standard deviation (σ) from the mean (μ) is a common cut-off in practice for identifying outliers in a Gaussian-like distribution.
- Limits on the sample values that are a factor $k = 1.5$ of the IQR below the 25% or above 75% in a Gaussian-like distribution.
- Outlier detection by clustering uses a simple method to estimate values such as DBScan Clustering, Isolation Forest, and Random Forest, which works well with low dimension data (few features).

4.3.2 Missing values

There is no perfect way to deal with missing data, especially if the number of missing values in the data is above 5%. Missing data must be marked with NaN, they can be replaced with statistical measures to calculate the column of values. Imputation techniques have to be used cautiously. The list of such techniques is as follows:

- Delete rows with missing data;
- Mean/Median/Mode/constant imputation:
  - The *mean* strategy replaces missing values using the mean along each column. It can only be used with numeric data.
  - The *median* strategy replaces missing values using the median along each column. It can only be used with numeric data.
  - The *mode* strategy replaces missing by the most frequent value along each column. It can be used with strings or numeric data.
  - The *constant* strategy replaces missing values with *fill_value*. It can be used with strings or numeric data.
- Predicting the missing values:

  - The KNNImputer is a data transform that is first configured based on the method used to estimate the missing values.
  - The default distance measure is a Euclidean distance measure, that is, NaN aware
  - The number of neighbors is set to 5 by default and can be configured by the *n_neighbors* argument.
- Using an algorithm which supports missing values, like random forests.

4.4 Data transformation

Data transformations scale numerical data, make the data more Gaussian or discrete data.

Data transformation applies a deterministic mathematical function $f_{transform}$ to each data point x_i in dataset X by replacing it with the transformed value $y_i = f_{transform}(x_i)$.

Inverse transformation is done as follows. The output from machine learning modelling can be transformed back to the original scale using the inverse f_{invert} of the transformation $f_{transform}$ that was applied to the data.

4.4.1 Data scaling

Many machine learning algorithms perform better or when features are relatively similar to normally distributed. Examples of such machine learning algorithms are linear regression, logistic regression, k-Nearest Neighbors algorithm, neural networks, Support Vector Machine, Principal Components Analysis (PCA), Linear Discriminant Analysis (LDA).

There are several scaling approaches such as:

Normalization is the process of transforming numeric data to the same scale as other numeric data. Normalization has the formulation:

$$x_{normalization} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{4.1}$$

Standardization or *z-normalization* has the formulation:

$$x_{standardized} = \frac{x - \mu}{\sigma} \tag{4.2}$$

where μ is the mean of variable x and σ is the standard deviation of x.

Robust scaler removes the median and scales the data according to the quantile range.

$$x_{robust} = \frac{x - x_{median}}{IQR} \tag{4.3}$$

In machine learning, every dataset does not require normalization (and/or standardiza-

tion). These approaches are required only when features have different ranges because this can lead to unstable or poor performance of models, increasing sensitivity to inputs or higher generalization error. They are necessary in the case of very different values within the same feature (for example, city population). Without normalization (and/or standardization), the training could blow up to NaN (not a number) if the gradient update is too large.

4.4.2 Data transformers

Power tranformer transform data to similarly normally distributed data. Many machine learning algorithms perform better or when features are relatively similar to normally distributed.

- Yeo-Johnson transform: works with positive and negative values;
- Box-Cox transform: only works with strictly positive values;
- Replacing the data with logarithms, square root or inverse to remove skew according to the setting value of the parameter λ:
  $\lambda = -1.0$ is a reciprocal transform $\frac{1}{x}$
  $\lambda = -0.5$ is a reciprocal square root transform $\frac{1}{\sqrt{x}}$
  $\lambda = 0.0$ is a log transform $\log(x)$
  $\lambda = 0.5$ is a square root transform $\sqrt{x}$
  $\lambda = 1.0$ is no transform.

Quantile transformer is applied to each feature independently. First, an estimate of the cumulative distribution function of a feature is used to map the original values to a uniform distribution. The obtained values are then mapped to the desired output distribution using the associated quantile function. This transform is nonlinear. It may distort linear correlations between variables measured at the same scale but renders variables measured at different scales more directly comparable.

4.4.3 Data discretization

Data discretization (also known as data bucketing or data binning) is used to reduce the effects of minor observation errors, e.g., the original data values fall into a given small interval, are replaced by a value representative of that interval, often the central value. Data discretization transfers numeric (usually continuous) data to categorical data. It is necessary about boundary setting and the bucketing type to apply.

- Buckets with equally spaced boundaries: the boundaries are fixed and encompass the same range.
- Buckets with quantile boundaries: each bucket has the same number of points. The boundaries are not fixed and could encompass a narrow or wide span of values.

4.5 Feature engineering

Feature engineering can use domain knowledge of the data and EDA to create features that make machine learning algorithms work. It also determines, which variables contribute the most to the predictive power of a machine learning methods. In many cases, the process of feature engineering which relies on hard testing work in EDA in combination with intuition and knowledge of data and Data Science process. Detailed domain documentation and strong collaboration with domain experts can improve the process significantly.

4.5.1 Feature extraction

Feature extraction[2] extract features in a format supported by machine learning algorithms from datasets consisting of formats such as text and image. It is usually used in the context of machine learning, pattern recognition, and image processing. Examples of feature extraction approaches are:

- *Bag-of-Words (BoW)*: a technique for natural language processing that extracts the words (features) used in a sentence, document, website, etc. and classifies them by frequency of use. This technique can also be applied to image processing.
- *Autoencoders*: the purpose of autoencoders is unsupervised learning of efficient data coding. Feature extraction is used here to identify key features in the data for coding by learning from the coding of the original data set to derive new ones.
- *Image processing*: algorithms are used to detect features such as shaped, edges, or motion in a digital image or video.

4.5.2 Feature encoding

Most of machine learning libraries work (only) with numbers, then encoding categorical data is an unavoidable step. There are a number of encoding methods, for example:

- ordinal encoding or label encoding,
- one hot encoding,
- binary encoding,
- BaseN encoding,
- hashing (using md5 algorithm),
- target encoding, and
- leave one out encoding.

Feature encoding can lead to *sparse data* and consequently to the curse of dimensionality. In general, sparse data refers to rows of data where many of the values are zero. Examples of sparse data are from many domain data such as text classification, recommender systems, customer-product purchases, user-song listen counts, user-movie ratings, natural language processing approach with Bag-of-Words, and one hot encoding.

[2] https://deepai.org/machine-learning-glossary-and-terms/feature-extraction

The fundamental reason for the curse of dimensionality is that high-dimensional functions have the potential to be much more complicated than low-dimensional ones. High-dimensionality might mean hundreds, thousands, or even millions of input variables. The presence of non-informative variables can add uncertainty to the predictions and reduce the overall effectiveness of the model.

The way to obtain a reduced representation in volume but produces the same or similar analytical results goes through data reduction, which is a part of the data preparation. Data preparation techniques have to be done on the data prior to data modeling.

4.5.3 Feature selection

Feature selection is a technique used to select a subset of input features that are most relevant to the target variable that is being predicted. It is a process of reducing the number of input variables when developing a predictive model. Feature selection aim is to reduce the number of input variables, to reduce the computational cost of modeling, and to improve the machine learning model performance. Feature selection can be divided into:

- *Filter* approach, that uses a specific metric as the feature filter. They are, for example, Variance Threshold, Mutual Information, Chi-Squared (χ^2), and F-value.
- *Wrapper* methods, which can be divided further into Forward Selection, Backward Elimination, and Recursive Feature Elimination.
- *Embedded* methods are usually are implemented inside of algorithms with built-in feature selection methods like Least Absolute and Selection Operator (Lasso), Ridge, and Elastic Net. Just certain machine learning methods have this option.

Filter feature selection methods

Filter feature selection methods use statistical techniques to evaluate the relationship between each input variable and the target variable. These scores are used to rank and choose those input variables to be used in the model.

Variance Threshold is the simplest method, which removes all features whose variance does not meet some threshold (linear column removing). This method has to be applied very cautiously.

Mutual Information (MI): the nonlinear relationship (or dependency) between two variables can be measured by Mutual Information with is a *non-negative* value. Mutual Information is is closely related to *information entropy* (Section 3.5).

Chi-square (χ^2) statistic compares the size and discrepancies between the expected results and the actual results, given the size of the sample and the number of variables in the relationship.

$$\chi^2_{df} = \sum \frac{(X_i - Y_i)^2}{Y_i} \tag{4.4}$$

where

df is the degree of freedom,
X contains observed values,
Y contains the expected values.

F-value or *F statistic* is a value produced from *ANOVA* test or a regression analysis to find out if the *means* between two populations are significantly different or not. It's similar to a T statistic from the *Student's t-test*.

$$F_{value} = \frac{variance_{dataset_1}}{variance_{dataset_2}} = \frac{\sigma_1^2}{\sigma_2^2} \tag{4.5}$$

Wrapper feature selection methods

This kind of feature selection creates many models with different subsets of input features. The selection of features is done based on the performance metrics of the best performing model. Wrapper feature selection methods can be computationally (very) expensive.

Forward Selection is an iterative process with the empty list of features at the beginning. During each iteration, feature(s) are added with the intent of improving the performance of the model. If the model performance is improved, feature(s) are kept, otherwise discarded. The process continues until the model performance is not improved.

Backward Elimination is an iterative process with the full list of all features at the beginning. Feature(s) with the least significance are removed during each iteration, and the process then checks if the performance of the model improves. The process is repeated until no significant improvement is observed.

■ **Example 4.1** Feature selection using *SelectFromModel* method in `scikit-learn` is meta-transformer for selecting features based on importance weights. The used model is usually some simple one(s) like *Logistic Regression* or *Linear Support Vector Classifier (SVC)* to lower computational requirements and time complexity. ■

Recursive Feature Elimination is a greedy optimization algorithm aiming to find the best performing feature subset. It creates models iteratively, and stores the best (or the worst) feature during each iteration. Next models are then constructed with the remaining features until all features are examined. The features are then ranked based on the order of their elimination.

■ **Example 4.2** A typical example of wrapper selection methods is *Recursive Feature Elimination (RFE)* method in `scikit-learn`. At first, the estimator is trained with the initial set of features and the importance of each feature. Then, the least important features are pruned from the current set of features. That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached. ■

Embedded methods

Embedded methods check the different training iterations of the machine learning model and evaluate the importance of each feature. Just certain machine learning methods (e.g. linear and logistic regression) that have this option, namely with Lasso (L1), Ridge (L2), and Elastic Net regularization.

4.5.4 Dimensionality reduction

Dimensionality reduction refers to reducing the number of input variables for a dataset. The resulting dataset can then be used as input to train a machine learning model. The popular approach of dimensionality reduction is to use techniques from the field of linear algebra. The resulting dataset is often called the *feature projection* and the algorithms used are referred to as *projection methods*.

> There is no best technique for dimensionality reduction.

Matrix factorization

Matrix factorization methods can be used to reduce a dataset matrix into its constituent parts by Eigen Vector Decomposition or Singular Value Decomposition (SVD). These constituent parts can be ranked and then selected that best captures the salient structure of the matrix, for example, by Principal Components Analysis (PCA).

Singular Value Decomposition (SVD) is a common dimensionality reduction technique. The `scipy.linalg.svd` factorizes the matrix A into two unitary matrices U and Vh and a 1D array s of singular values (real, non-negative) such that

$$A = U \cdot S \cdot Vh \tag{4.6}$$

where s is a suitably shaped matrix of zeros with the main diagonal s.

> Truncated SVD is the implementation of SVD in `scikit-learn`. It works on term count *tf-idf* matrices and it is also known as Latent Semantic Analysis (LSA).

Principal Components Analysis (PCA): linear dimensionality reduction using SVD of the data to project it to a lower dimensional space. The input data is centered but not scaled for each feature before applying the SVD. PCA identifies the combination of attributes that account for the most variance in the data.

Model-based projection

Linear Discriminant Analysis (LDA) is a model-based projection method, which is a classifier with a linear decision boundary, generated by fitting class conditional densities to the data and using Bayes' rule. LDA try to identify the attributes that account for the most variance between classes. The model fits a Gaussian density to each class, assuming that all classes share the same covariance matrix.

Manifold learning

Manifold learning are techniques used to create a low-dimensional projection of high-dimensional data, often for data visualization It can be viewed as a nonlinear version of PCA. The features in the projection often have little relationship with the original columns. Several well-known manifold methods are MultiDimensional Scaling (MDS), Isomap Embedding, and t-distributed Stochastic Neighbor Embedding (t-SNE)

4.6 Imbalanced datasets

The best way to approach machine learning solutions is to start by analyzing and exploring the dataset using EDA. One of the common issues found in datasets is the imbalanced classes issue, which usually reflects an unequal distribution of classes within a dataset. Binary classification models without fixing this problem will be very biased as well as feature correlation need to be improved. Imbalanced classes appear in many domains such as text classification, log data monitoring, fraud detection, spam filtering, disease screening, advertising Click-Through-Rate (CTR), recommender systems. Re-sampling techniques to deal with the imbalanced dataset issue are:

- *Undersampling* of the majority class(es): the process of randomly deleting some of the observations from the majority class to match the numbers in the minority class.
- *Oversampling* the minority class(es): the process of generating synthetic data that tries to randomly generate a sample of attributes from observations in the minority class.
- *Combining* minority classes: one of oversampling technique is Synthetic Minority Over-sampling Technique (SMOTE), which looks at the feature space for the minority class data points and considers its k nearest neighbours.

To deal with performance evaluation, we can apply other metrics such as ROC-AUC or PCC, which works better on imbalanced datasets. Machine learning methods, which can deal with imbalanced datasets are:

- Decision trees often perform well on imbalanced datasets because their hierarchical structure allows them to learn signals from both classes.
- Ensemble learning such as Random Forests and Gradient Boosted Trees uses multiple machine learning models to try to make better predictions. An ensemble model works by training different models on a dataset and having each model make predictions individually. The predictions of these models are then combined in the ensemble model to make a final prediction.

Ensemble learning can be divided into:

- *Bagging* = bootstrap resampling + aggregation. The final prediction is averaged.
- *Boosting*: multiple models are trained sequentially and each model learns from the errors of its predecessors.
- *Stacking/Voting ensemble*: the base models are trained entirely with different machine

algorithms.

The last solution to deal with imbalanced datasets is to *reframe* the solving problem as an anomaly detection problem.

4.7 Software available

`Python` is a programming language created by Guido van Rossum and first released in 1991 [12]. The rationale for using Python for Data Science, machine learning as well as EDA is because Python is a general purpose programming language for research, development, and production at small and large scales. Python features a dynamic system and automatic memory management, with large and comprehensive libraries for scientific computation and data analysis.

- `Pandas` is a Python package providing fast, flexible, and expressive data structures designed to make it easier to work with relational or labelled data [27]. Its two primary data structures, Series (one-dimensional) and DataFrame (two-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering.
- `NumPy` is the fundamental package for scientific computing with Python [25]. Besides its obvious scientific uses, NumPy can also be used as an efficient multidimensional container of generic data. NumPy stack has similar users to MatLab, GNU Octave, and SciLab.
- `SciPy` is an open-source Python library used for scientific computing and technical computing [24]. SciPy builds on the NumPy array object and is part of the NumPy stack which includes tools like Matplotlib, Pandas, and SymPy.
- `Scikit-learn` is widely known as a popular open-source Python tool which contains comprehensive library of machine learning and data analytic algorithms. The library supports supervised and unsupervised learning. It extends the functionality of NumPy and SciPy packages with numerous data mining algorithms and provide functions to perform classification, regression, clustering, dimensionality reduction, model selection, and preprocessing. Scikit-learn also provides various tools for data preprocessing, model selection, and evaluation as well as many other utilities.

Furthermore, within Python ecosystem, interactions with various databases can be realized with various Python SQL modules such as:

- SQLite (`pip install pysqlite3`)
- MySQL (`pip install mysql-connector-python`)
- PostgreSQL (`pip install psycopg2`)

Execution of SQL queries on various databases from a Python application are realized to connect to a database with operations such as connect to the database, create tables, insert records, select records, update table records, and delete table records.

Here are several engines (or ecosystems) for Big Data including machine learning libraries for intelligent data modelling and analysis such as Apache Spark and Apache Flink with Python as one of their programming languages. Such an ecosystem can run efficiently on a specialized cluster or a single machine.

4.8 Remark

Zed A Shaw. *Learn python 3 the hard way: A very simple introduction to the terrifyingly beautiful world of computers and code*. Addison-Wesley Professional, 2017. ISBN: 978-0134692883

5. Intelligent data modeling

Artificial Intelligence (AI) is any technique that aims to enable computers to mimic human behaviour [20], [31], including machine learning, computer vision [32], natural language processing [33], robotics, sensor analysis, optimization, and simulation (Figure 5.1). Machine learning is a subset of AI techniques that enable computer systems to learn from previous experience and improve their behaviour for a given task [4], [34]. They include Support Vector Machine, Naïve Bayes, decision trees, neural network, regression, association rules, regularization, and dimensionality reduction and many more algorithms [35], [36]. Neural networks are subset of machine learning techniques, loosely inspired by biological neural networks. They are usually described as a collection of connected units, called artificial neurons, organized in layers [37], [38], [39]. Deep learning is a subset of neural networks that makes the computational multi-layer network feasible. The most mentioned deep learning architectures are Fully Connected Networks, Convolutional Neural Network, Recurrent Neural Network, Generative Adversarial Networks, and autoencoder [40], [41].

5.1 Machine learning

Machine learning algorithms can be grouped according to the task they intend to solve. Below is one of the widely accepted machine learning algorithm categorizations [42]:

Supervised learning algorithms are used when each example in the training data consists of a pair (X_i, y_i), where X_i is the input to be fed into the predictor and y_i is the ground-truth label of the input. The training routine consists of tuning the predictor's parameters to make its output $f(X_i)$ as close to y_i as possible. Most of the classification and regression problems fall under this umbrella.

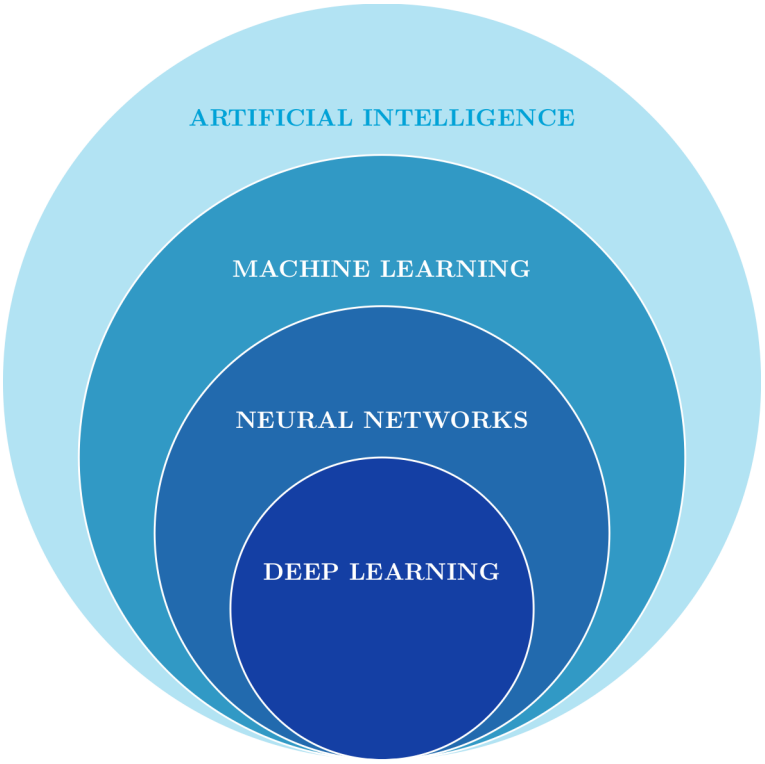


Figure 5.1: AI, machine learning, neural network and deep learning

Unsupervised learning is used to extract information from training data where a ground-truth label y_i is not available. Among the techniques that would belong to this category is clustering, density estimation, dimensionality reduction, generative adversarial networks, or problems where $X_i = y_i$ such as autoencoders.

Reinforcement learning is used to design algorithms that select appropriate actions in a given situation to maximize reward. In contrast to supervised learning where the predictor is provided with ground-truth labels, here the algorithm must learn by trial and error to find the optimal outputs.

It is interesting to note that the number of machine learning algorithms is increasing continually and machine learning algorithms have no strict categorization. It means that some methods can be listed in more categories. For example, neural networks can be used to solve supervised and reinforcement learning problems. Figure 5.2 illustrates one possible grouping of machine learning algorithms, which can be an object for further discussion and justification.

Machine learning algorithms learn from machine learning data

Once the machine learning data is prepared, some machine learning algorithm is chosen to solve the problem. The choice depends on many factors, including

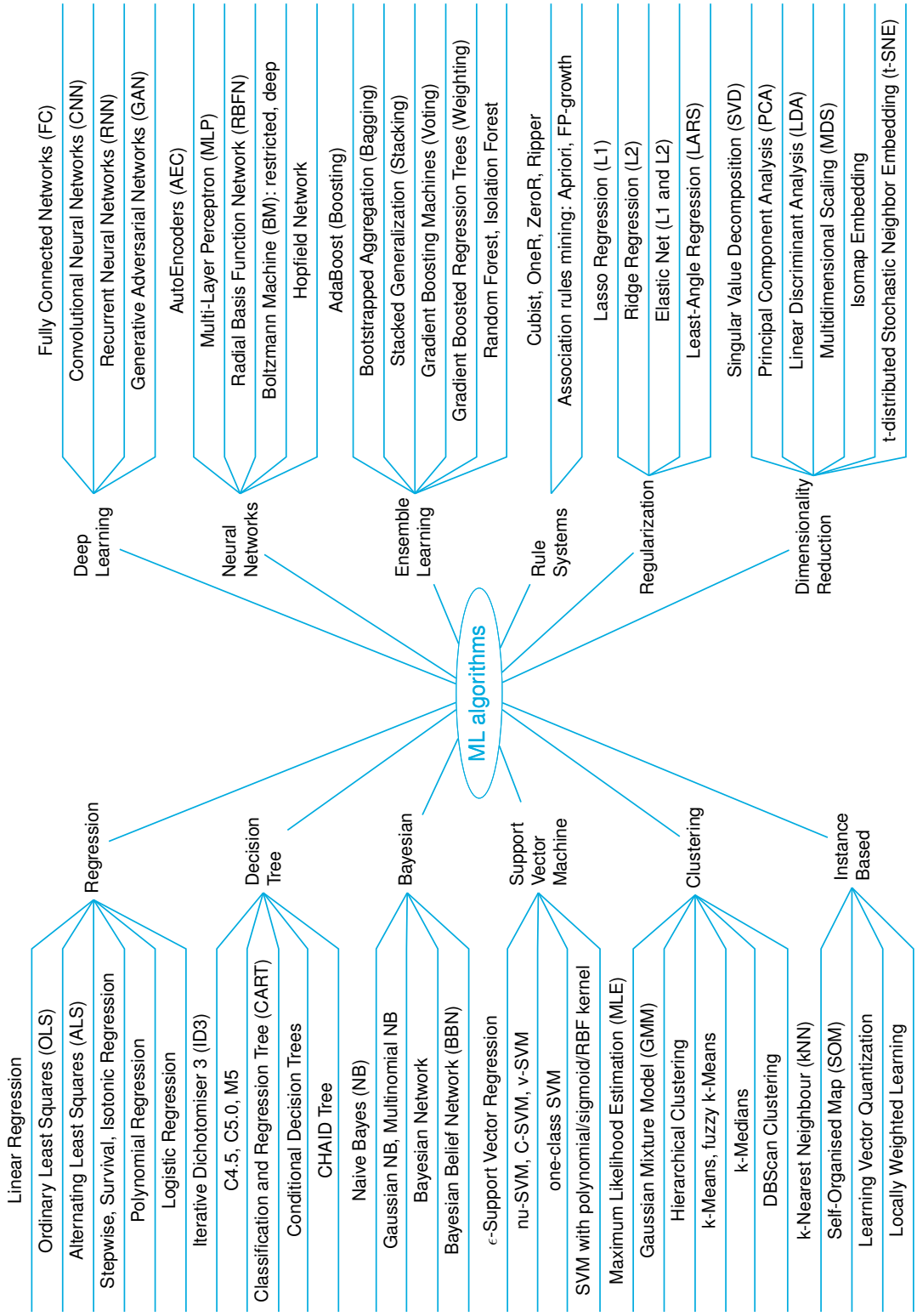


Figure 5.2: Machine learning algorithms

1. size, quality, and nature of the domain data,
2. available computational time,
3. urgency of the task,
4. structure of the desired predictions, and
5. loss to be minimized.

Machine learning algorithm learns from a set of training observations

$$\{X, Y\} = \{\{x_1, x_2, \cdots, x_N\}, \{y_1, y_2, \cdots, y_N\}\} \tag{5.1}$$

where

- x_i is an observation of X in the past,
- y_i is an observation of Y in the past. Y is often called label or response.

After learning, we obtain a model (other names: function, knowledge, experience), which can be used to predict (or to do inference) for future observations.

In supervised machine learning, algorithms learn a mapping from inputs to outputs. In the ideal case $Y = f(X)$, where $X = (x_1, x_2, \cdots, x_N)$. In practice, we usually have

$$\widehat{Y} = \widehat{f}(X) + e \tag{5.2}$$

Predictive modeling is to learn the mapping function $\widehat{f}$ to make predictions $\widehat{Y}$ for X_{new}. In this way, machine learning learns general concepts from specific examples (inductive learning). Supervised learning requires labeled examples to learn (Equation 5.3).

$$\{X, Y\} = \{features, labels\} \tag{5.3}$$

Supervised learning can be divided into

- *Classification* when the output variable is a category, i.e., Y belongs to a discrete set. An example of classification algorithms is logistic regression.
- *Regression* when the output variable is a real value, i.e., Y is a real number. An example of regression algorithms is linear regression.

Machine learning algorithms can be divided into

- *Generative learning*: generative model learns the joint probability distribution $p(x, y)$. Generative model can also generate data. Examples of generative learning are Naïve Bayes, Bayesian networks and Hidden Markov Models.
- *Discriminative learning*: discriminative model learns the conditional probability distribution $p(y|x)$. It is used to classify data. Examples of discriminative learning are logistic regression, Support Vector Machine, traditional neural network, nearest neighbour (kNN).

Often it is difficult to predict which algorithm would perform the best before trying

different algorithms after a thoughtful data examination. The choice of a concrete algorithm is based on data characteristics and EDA. As it is usual in machine learning, the performance of data models strongly depends on the representativeness of the provided data set. The complementary of methods leads to a selection from a wide spectrum of available modelling methods based on data characteristics and analysis. To reach the maximum performance, in many cases, it is necessary to train each model multiple times with different parameters and options (so-called model ensembling). Sometimes, it is also suitable to combine several independent models of different types, because each type can be strong in fitting different cases. The full data potential can be reached by ensemble learning based on principles such as voting, record weighting, multiple training process, or random selection. Weak learners can also be converted into stronger ones using boosting methods. Hence, a proper combination of several types of models with different advantages and disadvantages can be used to reach the maximum accuracy and stability in prediction.

5.1.1 Linear regression

In the simplest form, linear regression attempts to model the relationship between two variables by fitting a linear equation to the observed data. One variable is considered an independent variable (other names: x, predictor, attribute, feature, explanatory, variate) and the second is considered to be a dependent (other names: y, output variable, response, target) to the first one.

Simple linear regression: If a single independent variable, x is used to predict the value of a numerical dependent variable y. We can denote

$$y_i = b_0 + b_1 x_i \tag{5.4}$$

where

y is estimated (or predicted) variable;
b_0 is the estimation of the regression intercept;
b_1 is the estimation of the regression slope;
x is the independent variable.

Multiple linear regression: If more than one independent variable x is used to predict the value of a numerical dependent variable y.

$$y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n \tag{5.5}$$

Multivariate linear regression: if more than one independent variable x and more than one dependent variable y are linearly related.

Polynomial regression: it is suitable to mention polynomial regression, which is a regression

algorithm that models the relationship between a dependent y and independent variable x as n^{th} degree polynomial.

$$y = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + \ldots + b_n x^n \tag{5.6}$$

Usually, the output variable is a numerical value that exists on a continuous scale, but it can be put in another way as an integer or a floating point value.

Assumptions of linear regression are

- Linear relationship between features and target;
- *Homoscedasticity* meaning: *same variance* is central to the linear regression model;
- Small or no multicollinearity between the features, no autocorrelations;
- The error term should follow the normal distribution pattern.

Linear model

Let the input D has M observations:

$$D = \{(x_1, y_1), (x_2, y_2), \cdots, (x_M, y_M)\} \tag{5.7}$$

The learning function $f(x)$ is assumed to be a linear form:

$$f(x) = w_0 + w_1\, x_1 + \cdots + w_n\, x_n \tag{5.8}$$

where $w_0, w_1, ..., w_n$ are the regression coefficients.

Learning a linear function is equivalent to the learning the coefficient vector:

$$w = (w_0, w_1, \cdots, w_n)^T \tag{5.9}$$

Prediction by the regression model is calculated as:

$$y_x = w_0 + w_1\, x_1 + \cdots + w_n\, x_n \tag{5.10}$$

For each observation x, the true output is denoted as c_x, which is unknown for future data. We often expect:

$$y_x \approx c_x \tag{5.11}$$

Prediction for future observation $z = (z_1, z_2, \cdots, z_n)^T$ uses the previously learned function f to make the prediction is calculated as:

$$f(z) = w_0 + w_1\, z_1 + \cdots + w_n\, z_n \tag{5.12}$$

Regression model

Regression model learns a function $y = f(x)$ from a given training data D such that

$$y_i \approx f(x_i) \ \ \forall i \tag{5.13}$$

Each observation of x is represented by a vector in an n-dimensional space

$$x_i = (x_{i1}, x_{i2}, \cdots, x_{in})^T \tag{5.14}$$

where each dimension represents a feature.

Learning function

The learning goal is to learn a function f^* such that its prediction in the future is the best. It means the error (or expected loss) of

$$|c_z - f^*(z)| \tag{5.15}$$

is as small as possible for future z with the best generalization. The error (or loss) of the prediction for an observation $x = (x_1, x_2, \cdots, x_n)^T$ is

$$r(x) = \Big(c_x - f^*(x)\Big)^2 = (c_x - w_0 - w_1 x_1 - \cdots - w_n x_n)^2 \tag{5.16}$$

The expected loss E over the whole space is:

$$E = E_x(r(x)) = E_x(c_x - f^*(x))^2 \tag{5.17}$$

where E_x is the expectation over x.

Then the goal of learning is to find function $\widehat{f}$ that minimizes

$$\widehat{f} = argmin_{f \in H} E_x(r(x)) \tag{5.18}$$

where H is the space of linear functions.

Ordinary Least Squares (OLS)

OLS is a type of linear least squares method for estimating the unknown parameters in a linear regression model. The empirical loss in OLS is denoted as:

$$RSS(f) = \sum_{i=1}^{M} \Big(y_i - f(x_i)\Big)^2 = \sum_{i=1}^{M} \Big(y_i - w_0 - w_1 x_{i1} - \cdots - w_n x_{in}\Big)^2 \tag{5.19}$$

Residual Sum of Squares (RSS) denotes the sum of the squares of the residual errors. It is a measure of how well the model approximates the data. The next such measure is Residual Standard Error (RSE), which is the average variation of points around the fitted regression

line. The lower the RSE is, the better is the fitted regression model.

Given D, we find $\widehat{f}$ that minimizes RSS:

$$\widehat{f} = argmin_{f \in H} RSS(x) \tag{5.20}$$

That means the coefficient vector w^* is calculated as

$$\widehat{w} = argmin \sum_{i=1}^{M} \left(y_i - w_0 - w_1 x_{i1} - \cdots - w_n x_{in} \right)^2 \tag{5.21}$$

Finding w^* by taking RSS gradient and solving the equation $RSS' = 0$, we have

$$\widehat{w} = (A^T A)^{-1} A^T y \tag{5.22}$$

where

A is the data matrix of size $M(n+1)$;
$A_i = (1, x_{i1}, x_{i2}, \cdots, x_{in})$;
A^{-1} is the inversion of matrix A with the assumption that $A^T A$ is invertible;
$y = (y_1, y_2, \cdots, y_M)^T$.

Prediction of the new x is calculated as:

$$y_x = \widehat{w}_0 + \widehat{w}_1 x_1 + \cdots + \widehat{w}_n x_n \tag{5.23}$$

Ridge regression (L2) and Lasso regression (L1)

Ridge regression (L2) and Lasso regression (L1) use the regularization approach (L1 respectively L2) in machine learning to avoid overfitting. They are modifications over OLS aiming to minimize the sum of the square of residuals by another term equal to the sum of the square of regression parameters multiplied by a tuning parameter. The combination of L1 and L2 leads to Elastic Net, which profits from the tradeoff between variable selection and small coecifients.

Given input data

$$D = \{(x_1, y_1), (x_2, y_2), \cdots, (x_M, y_M)\} \tag{5.24}$$

We find $\widehat{f}$ that minimize RSS in Ridge regression (L2) as follows:

$$\widehat{f} = argmin RSS(f) + \lambda \left\| w \right\|_2^2 \tag{5.25}$$

and the coefficient vector

$$\widehat{w} = argmin \sum_{i=1}^{M} \left(y_i - A_i w\right)^2 + \lambda \sum_{i=1}^{n} w_j^2 \tag{5.26}$$

In the case of Ridge regression (L2), the penalty term is $\lambda \|w\|_2^2$.
In the case of Lasso regression (L1), the penalty term is $\lambda |slope|$.

Finding $\widehat{w}$ by taking RSS gradient and solving the equation $RSS' = 0$, we have

$$\widehat{w} = (A^T A + \lambda I_{n+1})^{-1} A^T y \tag{5.27}$$

where

A is the data matrix of size $M(n+1)$;
$A_i = (1, x_{i1}, x_{i2}, \cdots, x_{in})$;
A^{-1} is the inversion of matrix A with the assumption that $A^T A$ is invertible;
I_{n+1} is identity matrix of size $(n+1)$;
λ is regularization constant ($\lambda > 0$);
$y = (y_1, y_2, \cdots, y_M)^T$.

Prediction of the new x is calculated as:

$$y_x = \widehat{w}_0 + \widehat{w}_1 x_1 + \cdots + \widehat{w}_n x_n \tag{5.28}$$

The advantage of Ridge regression (L2) is it always works and it is used to avoid overfitting. The disadvantage is the error in training data can be bigger than with OLS and the predictiveness of Ridge depends heavily on the choice of the hyperparameter λ.

Linear regression has many variances and many implementations in various software libraries. Concretely, Python implementations of linear regression can be found in statsmodels, scikit-learn as well as in other Python libraries like MLLib (Apache Spark).

5.1.2 Logistic regression

Logistic regression, also known in the literature as logit regression, Maximum Entropy classification (MaxEnt) or the log-linear classifier. In this model, the probabilities describing the possible outcomes of a single trial are modeled using a *logit* function. Logistic regression is similar to linear regression but with two major differences

- It uses a *sigmoid* activation function (Section 5.2, Equation 5.55) on the output neuron to squash the output into the range $[0, 1]$ to represent the output as a probability;
- It uses a loss function called *logloss* to calculate the error instead of Mean Squared Error (MSE) in linear regression.

Logistic regression is a classification algorithm, used when the value of the target variable

is categorical in nature. Logistic regression is most commonly used when the data in question has a binary output, so when it belongs to one class or another, or is either 0 or 1. Logistic regression is the only classification algorithm in combination with a decision rule that makes dichotomous the predicted probabilities of the outcome. Logistic regression is a regression model because it estimates the probability of class membership as a (transformation of a) a multilinear function of the features.

Classification and regression tasks are both types of supervised learning. In a classification task, the outputs of an algorithm fall into one of various prechosen categories. The classification model attempts to predict the output value when given several input variables, placing the example into the correct category.

Input(s)

- Single-variate: if there's and only one input variable denoted with x;
- Multi-variate: for more than one input, the vector notation denoted as $X = (x_1, \cdots, x_r)$, where r is the number of independent features.

Output(s): the output variable is often denoted with y and takes the values 0 or 1.

- Binary or binomial classification: exactly two classes to choose between usually 0 and 1, true and false, or positive and negative;
- Multiclass or multinomial classification: three or more classes of outputs to choose from.

Logistic regression is a linear classifier (Equation 5.29):

$$f(x) = b_0 + b_1 x_1 + \cdots + b_r x_r \tag{5.29}$$

It is also called the *logit*. The variables $b_0, b_1, \cdots, b_r$ are the predicted weights.

The logistic regression function $p(x)$ is the sigmoid function of $f(x)$ or it is also called *inverse-logit* (Equation 5.30).

$$p(x) = \frac{1}{1 + e^{-f(x)}} \tag{5.30}$$

$f(x)$ is calculated in the relation with $p(x)$ as in Equation 5.31:

$$f(x) = log\Big(\frac{p(x)}{1 - p(x)}\Big) \tag{5.31}$$

Logloss function

Logloss or binary crossentropy is an evaluation metric for binary classification (Equation 5.32). It is used also as the optimization objective in logistic regression and neural

networks. For any problem, a lower logloss value means better prediction.

$$logloss_i = -\Big(y_i\ log(p_i) + (1 - y_i)\ log(1 - p_i)\Big) \tag{5.32}$$

where

i is the given observation,
y_i is the actual value,
p_i is the prediction probability.

The average of the logloss values of the N predictions is:

$$logloss = -\frac{1}{N}\sum_{i=1}^{N} logloss_i = -\frac{1}{N}\sum_{i=1}^{N}\Big(y_i\ log(p_i) + (1 - y_i)\ log(1 - p_i)\Big) \tag{5.33}$$

Categorical crossentropy metric is the log loss generalized to the multiclass problem (Equation 5.34). If N samples belonging to C classes, then we have:

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\ p_{ij} \tag{5.34}$$

where

$y_{ij} = 1$ if the $sample_i$ belongs to class j else $y_{ij} = 0$;
p_{ij} is the probability of a prediction of $sample_i$ belonging to class j.

Logloss is closely related to log-likelihood function (LLF). In fact, *logloss* is $(-1) \cdot LLF$.

■ **Example 5.1** Let y be the actual output and p be the predicted probability.

$$y = [1, 1, 0, 1, 0, 0, 0, 1, 0, 0] \tag{5.35}$$
$$p = [0.95, 0.85, 0.15, 0.71, 0.01, 0.12, 0.34, 0.90, 0.25, 0.41] \tag{5.36}$$
$$logloss \approx 0.219 \tag{5.37}$$

The *logloss* value is calculated using Equation 5.33. ■

Maximum Likelihood Estimation (MLE) is the maximization of the *logloss* function (or loss function) for all observations to get the best weights. Logistic regression determines the weights $b_0, b_1, \cdots, b_r$ that maximize the LLF (Section 3.5). In binary classification, for each observation $i = 1, \cdots, n$: the predicted output is 1, if $p(x_i) > threshold$. It is 0 in otherwise cases. The threshold does not have to be always 0.5, but it usually is.

Multiclass classification

Multinomial logistic regression is a form of logistic regression used to predict a target variable that has more than two classes. It is a modification of logistic regression using the *softmax* function instead of the *sigmoid* function for the *cross entropy logloss* function (Equation 5.38).

Cross entropy H is a measure of how different two probability distributions p, q are from each other (see also Section 3.5, Equation 3.42).

$$H(p,q) = -\sum_i p(x)\ log\ q(x) \tag{5.38}$$

This function has a range of $[0, \infty]$, its value equals to 0 when $p = q$ and equals to ∞ when p is very small compared to q or vice versa.

$Softmax$ is often used for multiclass classification (Equation 5.39). The output of $softmax$ is a vector of values in the range $[0, 1]$ with the total sum equals 1. These values represent the probability of classification for every class. The class with the highest probability is selected as the result of the classification.

$$softmax(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{5.39}$$

where $i \in \{1, \cdots, n\}$ and n is the number of classes in multiclass classification.

A number of machine learning algorithms like logistic regression, or Support Vector Machine are designed for binary classification. In general, they do not support classification tasks. A solution to this problem is to split the multiclass classification dataset into multiple binary classification datasets and fit a binary classification model to each of them. Here are two strategies of this approach: One-vs-Rest and One-vs-One.

One-vs-Rest (OVR) or *One-vs-All (OVA) strategy*

For each classifier, the class is fitted against all other classes, i.e., a separate binary classifier for each class fitting n models in n-class classification problem. Then, run all those classifiers on any new example to predict and take the class with the maximum score.

One-vs-One (OVO) strategy

This strategy consists in fitting one classifier per class pair. At the prediction time, the class which received the most votes is selected. It requires to fit $1/2 \cdot n(n-1)$ classifiers. This method is usually slower than One-vs-Rest due to its complexity $O(n^2)$, where n is the number of classes.

Logistic regression and neural network

Neural network are somewhat related to logistic regression. Basically, logistic regression can be seen as the simplest form of neural network, that results in decision boundaries that are a straight line. On the other hand, neural network is a superset that includes logistic regression and other classifiers that can generate more complex decision boundaries.

5.1.3 Naïve Bayes

The Naïve Bayes problem formulation is as follows. Let the input is a set of documents $D = \{d_1, d_2, \cdots, d_n\}$ with a fixed set of classes $C = \{c_1, c_2, \cdots, c_j\}$. The training set of labeled documents is $\{(d_1, c_1), \cdots, (d_m, c_m)\}$. Then the output is a learned classifier

$$\gamma : d \to c \tag{5.40}$$

Naïve Bayes is efficient and robust for problems such as text classification with supervised machine learning. It is based on Bayes' theorem with the *"naïve"* assumption of variable independence:

$$P(class|D) = P(class|d_1, \ldots, d_n) = \frac{P(class)\ P(d_1|class)\ \ldots\ P(d_n|class)}{P(D)} \tag{5.41}$$

$$P(class|D) = P(class|d_1, \ldots, d_n) = \frac{P(class)\ P(d_1|class)\ \ldots\ P(d_n|class)}{P(d_1, \ldots, d_n)} \tag{5.42}$$

giving us that the conditional probability of *class* is proportional to its prior probability and a product of conditional probabilities of items in D:

$$P(class|D) \propto P(class) \prod_{d \in D} P(d|class) \tag{5.43}$$

The classification (or recommendation) rule is defined as:

$$\hat{I} = \underset{class \in I}{\arg\max^{N}} P(class|D) \tag{5.44}$$

where $N \in \mathbb{N}$ indicates the number of top-N results to be returned in a ranked list $\hat{I}$. The most probable class c of a new document $d \in D$ with the maximum a posteriori (MAP) estimation is calculated as

$$c_{MAP} = argmax_{c \in C}\ P(c|d) = argmax_{c \in C}\ P(d|c)\ P(c)\ /\ P(D) \tag{5.45}$$

If $P(D)$ is the same for all d, we have

$$c_{MAP} = argmax_{c \in C}\ P(d|c)\ P(c) = argmax_{c \in C}\ P(x_1, x_2, \cdots, x_n|c)\ P(c) \tag{5.46}$$

We denote

$$c_{NB} = argmax_{c \in C} P(c) \prod_{x \in X} P(x|c) \tag{5.47}$$

where x_i are features.

5.1.4 Decision Tree

Decision trees have been used in many practical applications. It represents a function on a tree, which can be interpreted as a set of rules of the form IF-THEN.

Tree representation contains

- the root node: the first split which decides the entire population or sample data should further get divided into two or more homogeneous sets;
- splitting process is a process of dividing a node into two or more sub-nodes;
- decision nodes decide whether/when a subnode splits into further subnodes or not.
- list nodes (leaves) predict the outcome (categorical or continuous value).

Advantages of decision trees are:

- Decision trees are simple to understand and to interpret. Trees can be visualised.
- Decision trees are interpretable
- Decision trees are white box model, i.e., if a given situation is observable in a model, the explanation for the condition is explained by Boolean logic. It contrasts with the black box model (e.g., neural network), where results may be more difficult to interpret.
- Decision trees require little data preparation. Other techniques often require data normalisation, dummy variables need to be created, and blank values to be removed.
- Decision trees are able to handle multi-output problems.
- The cost of using the tree (predicting data) is logarithmic in the number of data points used to train the tree.
- It is possible to validate the model using statistical tests.
- Decision tree performs well even if its assumptions are violated by the true model from which the data were generated.

Disadvantages of decision trees are:

- Overfitting: decision-tree learners can create very complex trees that do not generalise the data well. Mechanisms such as pruning, setting the minimum number of samples required at a leaf node, or setting the maximum depth of the tree are necessary to avoid this problem.
- Unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.
- NP-complete learning for an optimal decision tree is known to be under several

aspects of optimum and even for simple concepts.
- There are concepts that are hard to learn because decision trees do not express them easily, such as XOR, parity, or multiplexer problems.
- Biased tree if some classes dominate. It is therefore recommended to balance the dataset prior to fitting to the decision tree.

Iterative Dichotomiser 3 (ID3)

ID3 is a greedy algorithm which was proposed by Ross Quinlan in 1986. It uses the top-down scheme.

- At each node N, select a test attribute A which can do best classification for the data in N. Generate a branch for each value of A, and then separate the data into its branches accordingly.
- Grow the tree until it classifies correctly all training data; or all attributes are used.

Entropy values (Section 3.5, Equation 3.38) in a decision tree are as follows:

- Entropy will always belong to $[0, 1]$.
- If classes are equally distributed than $entropy = 1$;
- If one class fully dominated than $entropy = 0$;
- If the sample is completely homogeneous, then $entropy = 0$;
- If the sample is equally divided, then it has an $entropy = 1$;
- For Gaussian distribution, the data set is widely distributed so entropy is maximum but less compare to uniform distribution because all have equal value;
- For uniform distribution, the dataset is equally distributed so maximum entropy;
- For peaked distribution, the dataset is equally distributed so entropy is near 0.

ID3 searches for a tree that fits well with the training data by growing the tree gradually. ID3 just searches for only one tree. ID3 never backtracks, as a consequence, it can find a local optimal solution/tree. Once an attribute has been selected, ID3 never rethinks of this choice. For a training dataset, there might be many trees that fit well with it. ID3 selects the first tree that fits the training data, it never reconsiders its choice when growing a tree.

The searching scheme of ID3 is to prefer simple trees. It prefers the tree in which the attributes with higher Information Gain (IG) (Section 3.5, Equation 3.51) will be and that means Information Gain decides the search direction of ID3. ID3 attribute selection based on Information Gain is to prefers the attribute that has more unique values. These attributes are placed closer to the root than the other ones. There are several problems with Information Gain such as biases towards attributes with many values or it does not work for new data. Therefore, Gain Ratio is used for ID3 attribute selection.

$$Gain\ Ratio = \frac{Information\ Gain}{Intrinsic\ Value} \tag{5.48}$$

Gain Ratio is an IG modification that reduces its bias on high-branch attributes. It takes the number and size of branches into account when choosing an attribute. It corrects the IG by taking the intrinsic information of a split into account. Gain Ratio is also called split ratio. Intrinsic Value (IV) is the entropy of the distribution of instances into branches, i.e., how much information to know which branch an instance belongs to.

ID3 complexity

The runtime cost to construct a balanced binary tree is

$$O\Big(n\ m\ log(n)\Big) \tag{5.49}$$

and the query time is

$$O\Big(log(n)\Big) \tag{5.50}$$

where n is the number of samples and m is the number of features/attributes.

Although the tree construction algorithm attempts to generate balanced trees, they will not always be balanced. Assuming that the subtrees remain approximately balanced. The cost at each node consists of searching through is

$$O(m) \tag{5.51}$$

The cost to find the feature that offers the largest reduction in entropy at each node of

$$O\Big(m\ n\ log(n)\Big) \tag{5.52}$$

leading to a total cost over the entire tree of

$$O\Big(m\ n^2\ log(n)\Big) \tag{5.53}$$

For a large dataset, a decision tree may not be a good option. Decision tree is good in the large dataset but less dimensions. It also good for low latency requirements.

Solutions for overfitting issues in ID3 are as follows:

- Pre-pruning: stop learning early before the tree fits the training data perfectly.
- Post-pruning the full tree: grow the tree to its full size, and then post prune the tree, but it is hard to decide when to stop learning. Post-pruning the tree empirically results in better performance, e.g., reduced-error pruning or rule-post pruning.

ID3 successors: C4.5, C5.0 and CART

- C4.5 is the successor to ID3, it removes the restriction that features must be categorical by dynamically defining a discrete attribute (based on numerical variables). It

partitions the continuous attribute value into a discrete set of intervals.
- C5.0 is Quinlan's latest version release under a proprietary license. It uses less memory and builds smaller rulesets than C4.5 while being more accurate.
- CART (Classification and Regression Trees) is very similar to C4.5. It differs in that it supports numerical target variables and does not compute rule sets. CART constructs binary trees using the features and threshold that yield the largest IG at each node. CART is implemented in `sklearn` and it uses *Gini* (Section 3.5, Equation 3.39) to create split points including Gini index (Gini impurity) and Gini Gain.

Random Forests

Random Forests is a method for both classification and regression. The main idea of Random Forests is

- Prediction is based on the combination of many decision trees, by taking the average of all individual predictions.
- Each tree in Random Forests is simple but random.
- Each tree is grown differently, depending on the choices of the attributes and training data.

Currently, Random Forests is one of the most popular and accurate methods; it is also very general. Random Forests can be implemented easily and efficiently. Random Forests can work with problems of very high dimensions without overfitting.

The basic ingredients of Random Forests are

- Randomization and no pruning:
  - For each tree and at each node, we select randomly a subset of attributes.
  - Find the best split, and then grow the appropriate subtrees.
  - Every tree will be grown to its largest size without pruning.
- Combination: each prediction later is made by taking the average of all predictions of individual trees.
- Bagging: the training set for each tree is generated by sampling with replacement from the original data.

Random Forests is fast, scalable and performance. It is parallelizable, e.g., with the option $n\_jobs = -1$ in scikit-learn to use all available capacity of machines. Random Forests provides quick prediction and training speed, it is robust to outliers and non-linear data. Random Forests can handle unbalanced data with low bias and moderate variance.

The drawbacks of Random Forests is the model interpretability. For very large data sets, the size of the trees can take up a lot of memory. It can tend to overfitting, so hyper-parameter tuning is needed.

5.2 Neural network and deep learning

Neural networks are a subset of machine learning techniques. These networks are not intended to be realistic models of the brain, but rather robust algorithms and data structures able to model difficult problems. Neural networks have units (neurons) organized in layers. There are three main layer categories: input layer, hidden (middle) layers, and output layer. Neural networks can be divided into: shallow (one hidden layer) networks, and deep (more hidden layers) networks.

The predictive capability of neural networks comes from this hierarchical multilayered structure. Through proper training, the network can learn how to optimally represent inputs as features at different scales or resolutions and combine them into higher-order feature representations relating these representations to output variables and therefore learning how to make predictions. Neural networks are effective universal approximators [43], which show promising results in many machine learning tasks [44]. The simplest and widely used formal description of a neural network is the nonlinear weighted sum:

$$y = f_A(x) = f_A\left(b + \sum_i^n w_i x_i\right) \tag{5.54}$$

where

y is the output,
x is the input vector of size n,
b is the bias;
w is the weight vector;
f_A is the activation function such as tanh, sigmoid, softmax or rectifier.

Deep neural network (DNN) are considered to be capable of learning high-level features with more complexity and abstraction due to their larger number of hidden layers than shallow neural networks. In order to achieve high accuracy in prediction, there are two dependent problems that have to be addressed; i.e., network architecture and training routine [40], [41], [45]:

- Defining network architectures involves setting fine-grained details such as activation functions (hyperbolic tangent, Rectified Linear Unit (ReLU), maxout, etc.) and the types of layers (fully connected, dropout, batch normalization, convolutional, pooling, etc.) as well as the overall architecture of the neural network.
- Defining training routines involves setting the learning rate schedules (stepwise, exponential); learning rules such as Stochastic Gradient Descent (SGD), SGD with momentum, Root Mean Square Propagation (RMSprop), Adaptive Moment Estimation Algorithm (Adam); the loss functions like MSE, categorical cross entropy; regularization techniques liek L1, L2 weights decay, early stopping; and hyperparameter optimization such as grid search, random search, bayesian guided search.

In DNN models, the activation functions of the hidden units are used to capture the nonlinearity. Without the nonlinearity, the network would not be more powerful than plain perceptrons without hidden layers. The most frequently used activation functions are sigmoid, tanh, ReLU and softmax.

A *sigmoid* function is a mathematical function having a characteristic *S-shaped* curve or sigmoid curve. *sigmoid* function most often show a return value $\sigma(x)$ in the range $[0,1]$, but the range $[-1,1]$ is also possible. A common example of a sigmoid function is the *logistic function* (Equation 5.55). The sigmoid function is often used in the output layer of neural network for binary classification. The return values are in the range $[0,1]$.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{5.55}$$

tanh or hyperbolic tangent is the next type of sigmoid function. This function returns values in the range $[-1,1]$. It is a proportion of hyperbolic sine and cosine, but it can also be expressed using a sigmoidal function (Equation 5.56).

$$tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{1-e^{-2x}}{1+e^{-2x}} = 2\sigma(2x) - 1 \tag{5.56}$$

Rectified Linear Unit (ReLU) is often used in hidden layers of neural networks, especially of Convolutional Neural Networks.

$$f(x) = max(0,x) \tag{5.57}$$

The function returns 0 if it receives any negative input, but for any positive value x, it returns that value back. Thus it gives an output that has a range from 0 to infinity.

$Softmax$ is often used for multiclass classification (Equation 5.58). The output of $softmax$ is a vector of values in the range $[0,1]$ with the total sum equals 1. These values represent the probability of classification for every class. The class with the highest probability is selected as the result of the classification.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}} \tag{5.58}$$

where $i \in \{1, \cdots, n\}$ and n is the number of classes in multiclass classification.

Many DNN models have been developed over the past two decades. Each of these models has a different network architecture in terms of number of layers, layer types, layer shape, and connections between layers.

Below are some applications of the most popular deep learning architectures. Many

applications are simultaneously composed of several types of architectures.

- Fully Connected Networks (FC) are the most common architecture as they can be used to model a wide variety of problems that use tabular data. It has been a long time since they are used in finance, physics, or biomedicine.
- Convolutional Neural Network (CNN) are networks specially designed to deal with images (or more generally with translation invariant data). Current applications include a wide variety of image classification, image segmentation, autonomous driving or satellite imagery.
- Recurrent Neural Network (RNN) are specially designed to deal with sequential data. They are widely used in NLP like neural machine translation (NMT), language generation, time series analysis, medicine or climatology. If the sequential data are images (like in videos), then RNNs are used in conjunction with CNNs.
- Generative Adversarial Networks (GAN) are networks that compete with themselves to create the most possible realistic data. Current applications include image style transfer, high resolution image synthesi, text-to-image synthesis, image super-resolution, anomaly detection, music generation and many more.

5.2.1 Convolutional Neural Network

Convolutional Neural Network (CNN) were originally developed for working with two-dimensional image data. However, they also show effectiveness in automatic feature extraction and feature learning from sequence data [20], [40]. A common design pattern to create a CNN is done by alternating layers such as convolutional layer (ConV), max pooling layer (MaxPooling) and fully connected layers (FC) as follows

$$Input - ConV - MaxPooling - ConV - MaxPooling - FC - FC(Output) \quad (5.59)$$

Yann LeCun used a similar design in his famous LeNet model. This design is popular in neural-inspired models of visual object recognition and other practical applications.

Convolution layers (ConV) are the core building block of CNN. This layer performs a *dot product* between two matrices, where one matrix is *kernel* (the set of learnable parameters) and the other matrix is the restricted portion of the receptive field. During the forward pass, the kernel slides across the image and produces an *activation map*. The sliding size of the kernel is called a *stride*. Convolution layer has three important ideas motivated from computer vision:

- sparse interaction: achieved by making the kernel smaller than the input;
- parameter sharing: realized by applying the same weights to one input as elsewhere;
- equivariant representation: if the input is changed in some way, the output will also get changed in the same way. It is useful for tasks such as image classification, where the goal is to classify if an object is present at any location.

Pooling layers replaces the output of the network at certain locations by deriving a summary

statistic of the nearby outputs.

- *Max pooling* layer computes the maximum over its incoming values.
- *Average pooling* is an alternative to use, which replaces the maximum by the mean.
- *Global average pooling* is averaging over all locations in a feature map.

Fully connected layers (FC) Neurons in this layer have full connectivity with all neurons in the preceding and succeeding layer. This layer helps to map the representation between the input and the output.

Since convolution is a linear operation, nonlinearity layers are often placed after the convolutional layer to introduce nonlinearity to the activation map. There are several types of nonlinear operations such as $sigmoid$, $tanh$, $ReLU$ and $softmax$ as the last layer for multiclass classification. CNN have also other layer types such as normalization layers. Typical example is *batch normalization* standardizes all the activations within a given feature channel to be zero mean and unit variance. This can significantly help with training. To avoid problems with small batches, *layer* normalization and *instance* normalization are introduced. A natural generalization of the above methods is known as *group* normalization.

5.2.2 Recurrent Neural Network

RNN handles sequences by having a recurrent hidden state, whose activation at each time is dependent on previous time steps [40]. The most famous RNN special units (building blocks) are Long Short-Term Memory (LSTM) (Equations 5.60 to 5.65) and Gated Recurrent Unit (GRU) (Equations 5.66 to 5.69). They are similar in overall structure, but GRU has a simpler form [46], [47].

$$\text{Input gate} \quad i_t = \sigma\left(x_t U^i + h_{t-1}\mathcal{W}^i\right) \tag{5.60}$$

$$\text{Forget gate} \quad f_t = \sigma\left(x_t U^f + h_{t-1}\mathcal{W}^f\right) \tag{5.61}$$

$$\text{Hidden state} \quad \tilde{\mathcal{C}}_t = tanh\left(x_t U^g + h_{t-1}\mathcal{W}^g\right) \tag{5.62}$$

$$\text{Internal memory} \quad \mathcal{C}_t = \sigma\left(f_t * \mathcal{C}_{t-1} + i_t * \tilde{\mathcal{C}}_t\right) \tag{5.63}$$

$$\text{Output hidden state} \quad h_t = tanh(\mathcal{C}_t) * o_t \tag{5.64}$$

$$\text{Output gate} \quad o_t = \sigma\left(x_t U^o + h_{t-1}\mathcal{W}^o\right) \tag{5.65}$$

where

x is the input vector;
$\mathcal{W}$ is the recurrent connection at the previous hidden and current hidden layer;
U is the weight matrix connecting the inputs to the current hidden layer;
$*$ is the element-wise multiplication and ignore bias term;
σ is a logistic sigmoid function;
$\tilde{\mathcal{C}}$ is a hidden state computed based on the current input and the previous hidden state.

GRU has a reset (r) gate and an update gate (z) instead of the input (i), output (o), and forget (f) gates of the LSTM. GRU also has shown to be faster in training than LSTM [47].

$$\text{Update gate } z_t = \sigma\left(x_t U^z + h_{t-1}\mathcal{W}^z\right) \tag{5.66}$$

$$\text{Reset gate } r_t = \sigma\left(x_t U^r + h_{t-1}\mathcal{W}^r\right) \tag{5.67}$$

$$\text{Hidden state } \tilde{h}_t = tanh\left(x_t U^h + (r_t * h_{t-1})\mathcal{W}^h\right) \tag{5.68}$$

$$\text{Output hidden state } h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{5.69}$$

RNN as well as LSTM and GRU come with more complex versions such as bidirectional, autoencoder, or stacked ones with more RNN blocks.

- Bidirectional LSTM combines an LSTM that moves forward through time from the start of the sequence, with another LSTM that moves backward through time from the end of the sequence [48]. This allows o_t to compute a representation that depends on both the past and the future, with most sensitive input gate i_t, without fixed time-step specification around t.
- The idea of autoencoder LSTM is borrowed from natural language processing and video representation as sequence to sequence (seq2seq) modeling [49].

The *attention mechanism* for seq2seq modeling is recently published in [50], [51] for NMT. The mechanism is an improvement to the model that allows it to pay attention to different words in the input sequence as it outputs each word in the output sequence. The attention LSTM model in this work is built by adding one self-attention layer with local attention to bidirectional LSTM block.

Stacked models and stacked hidden layers make the model deeper [52]. This promises more accurate learning but at the cost of model complexity and the cost of computing resources during training. Stacked model consists of multiple hidden layers (or blocks) where each layer contains customizable multiple memory cells, one on top of another.

5.3 Model evaluation and selection

In general, models like neural network depend on the initial weights, which are initialized randomly. Different random seed values can lead to inconsistent prediction performance from one training session to another. For this reason, the evaluation is usually repeated multiple times (if it is possible with time and computational power) to ensure model quality and stability in production. We calculate for the resulting series $r = \{r_1, \ldots, r_n\}$ where n is the number of result values, the following metrics:

Average (Equation 5.70)

$$avg = \bar{r} = \frac{1}{n}\sum_{i=1}^{n} r_i \tag{5.70}$$

Standard deviation (Equation 5.71)

$$std = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(r_i - \bar{r})^2} \tag{5.71}$$

Coefficient of variation (Equation 5.72), which quantifies the dispersion of values in r.

$$cv = \frac{std}{\bar{r}} \tag{5.72}$$

Choosing the right metric is crucial while evaluating models. Various metrics are proposed to evaluate the models in different applications. Looking at a model with a single metric may not give the whole picture of solving problems.

5.3.1 Model performance measurements

Evaluating the performance of intelligent models built using machine learning algorithm is an essential part of any project. Metrics are used to monitor and measure the performance of a model during training and testing. They report the progress in measurable ways with numerical numbers.

Classification metrics

Let True Positives (TP), False Positives (FP), True Negatives (TN), False Negatives (FN) are outcome measurements of p and y where p represents predicted values, y represents label and n is the number of data records. The most used and mentioned classification metrics are Accuracy (ACC), precision, recall, and the F_1 score. Their values are in the range $[0, 1]$, where a higher value indicates a better result.

Accuracy (ACC) is the proportion of true results among the total number of observations (Equation 5.73). It is used for classification problems, in which the data is balanced and not skewed.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.73}$$

Precision (Equation 5.74) answers the question: what proportion of predicted Positives is truly positive. It is used for to show the surety of prediction.

$$Precision = \frac{TP}{TP + FP} \tag{5.74}$$

Recall answers the question: what proportion of actual Positives is correctly classified. It

is used to capture as many positives as possible (Equation 5.75).

$$Recall = \frac{TP}{TP + FN} \tag{5.75}$$

The $F1$ score is the harmonic mean of precision and recall. It gives equal weight to precision and recall (Equation 5.77).

$$F1 = 2\,\frac{Precision \cdot Recall}{Precision + Recall} \tag{5.76}$$

When the weight is β, the formula for F_β becomes:

$$F_\beta = (1 + \beta^2)\frac{Precision \cdot Recall}{\beta^2\ Precision + Recall} \tag{5.77}$$

Pearson Correlation Coefficient (PCC) is a measure of linear correlation between two sets of data. It is the ratio between the covariance of two variables and the product of their standard deviations. Its values are in the range $[-1, 1]$. PCC reflects the strength and direction of correlations (Equation 5.78).

$$PCC = \frac{cov(p,y)}{\sigma_p \sigma_y} = \frac{TP\ \ TN - FP\ \ FN}{\sqrt{(TP + FN)(FP + TN)(TP + FP)(FN + TN)}} \tag{5.78}$$

Generally, no perfect measurement method exists for model evaluation, but if the data is imbalanced then PCC is considered to be better than ACC.

Furthermore, Receiver Operator Characteristic Curve (ROC) is the curve, which is plotting True Positive Rate (TPR) (Equation 5.79) as y-axis against False Positive Rate (FPR) (Equation 5.81) as x-axis and Area Under the Curve (AUC) gives the rate of successful classification by the logistic model.

Sensivity or True Positive Rate (TPR)

$$Sensivity = Recall = \frac{TP}{TP + FN} = TPR \tag{5.79}$$

Specificity or True Negative Rate (TRN)

$$Specificity = \frac{TN}{TN + FP} = TRN \tag{5.80}$$

False Positive Rate (FPR)

$$FPR = 1 - Specificity = \frac{FP}{TN + FP} \tag{5.81}$$

For multiclass classification, micro averaging calculates metrics globally by counting the total TP, FN and FP for C classes (Equation 5.82 and Equation 5.83).

Micro averaging precision

$$precision = \frac{\sum_{i=1}^{C} TP_i}{\sum_{i=1}^{C} (TP_i + FP_i)} \tag{5.82}$$

Micro averaging recall

$$recall = \frac{\sum_{i=1}^{C} TP_i}{\sum_{i=1}^{C} (TP_i + FN_i)} \tag{5.83}$$

Macro averaging calculates the metrics for each label and finds their unweighted mean (Equation 5.84 and Equation 5.85).

Macro averaging precision

$$precision = \frac{\sum_{i=1}^{C} precision_{C_i}}{C} \tag{5.84}$$

Macro averaging recall

$$recall = \frac{\sum_{i=1}^{C} recall_{C_i}}{C} \tag{5.85}$$

Except for micro and macro metrics as above, weighted averaging calculates the metrics for each label and finds their average weighted support, i.e., the number of true instances for each label (Equation 5.86).

$$\overline{x} = \frac{\sum_{i=1}^{C} w_i x_i}{\sum_{i=1}^{n} w_i} \tag{5.86}$$

Logloss or binary crossentropy is also an evaluation metric for binary classification (more

details are in Section 5.1.2).

In the context of classification and metrics, contingency tables and confusion matrices are often mentioned. *Contingency table* (also known as cross tabulation or crosstab) is a type of table in a matrix format. It displays the (multivariate) frequency distribution of the variables. Contingency table is used to describe the data. *Confusion matrix* is used to evaluate the performance of a classifier. It tells how accurate a classifier is in making predictions about classification. Confusion matrix can be seen as a special kind of contingency table, with two dimensions ("actual" and "predicted"), and identical sets of "classes" in both dimensions.

Regression metrics

Regression models have continuous output. Thus, a regression metric calculates the distance between the predicted values $\widehat{y}$ from the regression model and the ground truth values y, where n is the number of observations. The most used and mentioned regression metrics are Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Square Error (RMSE), and Coefficient of Determination (R2).

Mean Squared Error (MSE) is the most popular regression metric. It calculated the average of the squared difference between the ground truth values and the predicted values produced by the regression model (Equation 5.87).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2 \tag{5.87}$$

Mean Absolute Error (MAE) is the average of the difference between the ground truth and the predicted values (Equation 5.88).

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y}_i| \tag{5.88}$$

Root Mean Square Error (RMSE) is $sqrt(MSE)$. It corresponds to the square root of the average of the squared difference between the ground truth values and the predicted values produced by the regression model (Equation 5.89).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2} \tag{5.89}$$

Coefficient of Determination (R2) value of 100% means the model explains all variation of the target variable and a value of 0% measures the zero predictive power of the model. Thus, the higher the R2 value the better the model (Equation 5.90). There are cases where the computational definition of R2 can yield negative values, depending on the definition

used. R2 is bounded above by 1.0, but it is not bounded below. The reason is the evaluation (score) on unseen data, which can lead to results outside the range $[0, 1]$.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(y_i - \widehat{y_i}\right)^2}{\sum_{i=1}^{n}\left(y_i - \frac{1}{n}\sum_{j=1}^{n} y_j\right)^2} \tag{5.90}$$

R2 may suffer from some problems, e.g., giving the false impression that the model is improving when the score is increasing, but in reality, the learning is not happening. This can happen when a model overfits the data, in that case the variance explained will be 100% but the learning has not happened. To rectify this, adjusted R2 is proposed (Equation 5.91) with the number of independent variables[1].

$$R^2_{adjusted} = 1 - \left((\frac{n-1}{n-k-1})(1-R^2)\right) \tag{5.91}$$

where n is the number of observations and k is the number of independent variables. Adjusted R2 is always lower than R2, as it adjusts for the increasing predictors and only shows improvement if there is a real improvement.

The next two metrics are Mean Absolute Percentage Error (MAPE) and Symmetric Mean Absolute Percentage Error (SMAPE), which have the *divide by zero* problem in implementation (Equation 5.92 and Equation 5.93).

$$MAPE = 100\%\frac{1}{n}\sum_{i=1}^{n}\left|\frac{(y_i - \widehat{y_i})}{y_i}\right| \tag{5.92}$$

SMAPE formula provides results in 200% scale interpretation instead of the frequently used 100% scale interpretation. It is less sensitive to low-volume items (i.e., near-zero) in comparison with other similar metrics such as MAPE.

$$SMAPE = 100\%\frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \widehat{y_i}|}{\frac{1}{2}\left(|y_i| + |\widehat{y_i}|\right)} \tag{5.93}$$

cosine similarity is the cosine of the angle between two n-dimensional vectors in an n-dimensional space. It is the dot product of two vectors divided by the product of their lengths. The result values are in the range $[-1, 1]$, where -1 meaning exactly opposite,

[1] https://neptune.ai/blog/performance-metrics-in-machine-learning-complete-guide

1 meaning the same, and 0 indicating orthogonality or noncorrelation.

$$cosine = \frac{y\,\widehat{y}}{\|y\|\,\|\widehat{y}\|} = \frac{\sum_{i=1}^{n} y_i\,\widehat{y}_i}{\sqrt{\sum_{i=1}^{n} y_i^2}\sqrt{\sum_{i=1}^{n} \widehat{y}_i{}^2}} \tag{5.94}$$

In machine learning, loss functions measure the inconsistency between the ground truth vector y and the forecasting vector $\widehat{y}$ of n observations. Robustness of a model increases with the decrease of the loss function value. Frequently used regression metrics are MAE, MSE, and RMSE.

Here are more distance metrics, which measure the similarity between two vectors such as Manhattan distance, Euclidean distance, Levenshtein distance. Chebyshev distance, Minkowski distance, Canberra distance, Sokal-Sneath distance, and many more distance metrics can also be listed here[2].

Manhattan distance (Equation 5.95):

$$d_1(p,q) = \|p-q\|_1 = \sum_{i=1}^{n} |p_i - q_i| \tag{5.95}$$

Euclidean distance (Equation 5.96):

$$d(p,q) = d(q,p) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \tag{5.96}$$

where $p = (p_1, p_2, \cdots, p_n)$, and $q = (q_1, q_2, \cdots, q_n)$.

Levenshtein distance (Equation 5.97) between two strings a, b is calculated as follows:

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min\big(lev_{a,b}(i-1,j)+1, & \\ \quad lev_{a,b}(i,j-1)+1, & \\ \quad lev_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)}\big) & \text{otherwise.} \end{cases} \tag{5.97}$$

where

i is the terminal character position of a,
j is the terminal character position of b.
All positions are indexed from 1.

[2] https://www.sequentix.de/gelquest/help/distance_measures.htm

5.3.2 Generalization, overfitting and underfitting

Generalization refers to how well the concepts learned by a machine learning model apply to specific examples not seen by the model when it was learning. Concretely, inductive learning refers to learning general concepts from specific examples and statistical fit refers to how well is the approximation done for the target function.

Bias and variance

The prediction error for any machine learning model can be divided into bias error, variance error, and irreducible error. The last one can not be reduced regardless of machine learning algorithms.

Machine learning algorithms like Decision Trees, k-Nearest Neighbors algorithm, and Support Vector Machine provide *low bias*, which suggests fewer assumptions about the form of the target function. Machine learning algorithms like linear regression, logistic regression and Linear Discriminant Analysis provide *high bias*, which suggests more assumptions about the form of the target function.

On the other hand, machine learning algorithms like linear regression, logistic regression and Linear Discriminant Analysis have *low variance*, which suggests small changes to the estimate of the target function with changes to the training dataset Machine learning algorithms like Decision Trees, k-Nearest Neighbors algorithm, Support Vector Machine, and neural network have *high variance*, which suggests large changes to the estimate of the target function with changes to the training dataset.

Overfitting and underfitting

The cause of poor performance in machine learning is either overfitting or underfitting the data.

Underfitting refers to failing to learn the problem from the training data sufficiently. It does provide good contrast to the overfitting problem. A solution for the underfitting problem can be:

- increasing the training time of the model or
- increasing the number of features.

Overfitting refers to learning the training data too well at the expense of not generalizing well to new data. Overfitting happens when a model learns the detail and noise in the training data. This has negatively impacted the performance of the model on new data. Overfitting is the most common problem in practice and can be addressed by:

- cross-validation (holdout and k-fold),
- training with more data,
- removing features,
- early stopping the training,
- regularization
- ensembler learning.

5.3.3 Cross-validation and regularisation

Cross-validation is a useful technique for assessing the model effectiveness and to prevent overfitting. It uses different portions of the data to test and train a model at different iterations.

Holdout validation

The basic idea is to partition the data into two disjoint sets, the training set D_{train} and the validation set D_{valid}. The often used percents is about 80% of the data for the training set, and 20% for the validation set. The model then is fit on D_{train} and then evaluate its performance on D_{valid}.

K-fold validation

If the size of the training set is small, just leaving aside 20% for a validation set can result in an unreliable estimate of the model parameters. A popular solution is to use cross-validation with the following idea:

- the training data is splitted into K folds;
- for each fold $k \in \{1, ..., K\}$, the train is done on all the folds except the k^{th}, and test on the k^{th} in a round-robin fashion.

Formally D_k is the data in the k^{th} fold, and D_{-k} is all other data. If $K = N$, the method is known as *leave-one-out* cross validation. As the amount of data increases, the chance of overfitting (for a model of fixed complexity) decreases assuming the data contains suitably informative examples and is not too redundant.

Regularisation

In mathematics, statistics, finance, computer science, and particularly in machine learning and inverse problems, regularization is the process of adding information to solve an ill-posed problem (underfitting) or to prevent overfitting.

L1 regularization (Lasso)
The regularization term Ω is the sum of the absolute values of the weight parameters in a weight matrix (Equation 5.98).

$$\Omega(W) = \sum_i \sum_j |w_{ij}| \tag{5.98}$$

Regularisation techniques applied to logistic regression mostly tend to penalize large coefficients $b_0, b_1, \cdots, b_r$. L1 regularization (Lasso) penalizes the log-likelihood function (LLF) with the scaled sum of the absolute values of the weights:

$$|b_0| + |b_1| + \cdots + |b_r| \tag{5.99}$$

L2 regularization (Ridge)

The regularization term Ω is defined as the Euclidean Norm (or L2 norm) of the weight matrix (Equation 5.100).

$$\Omega(W) = \sum_i \sum_j w_{ij}^2 \tag{5.100}$$

L2 regularization (Ridge) penalizes the log-likelihood function (LLF) with the scaled sum of the squares of the weights:

$$b_0^2 + b_1^2 + \cdots + b_r^2 \tag{5.101}$$

Elastic Net regularisation is a linear combination of L1 and L2. Regularization can significantly improve the model performance on unseen data.

Early stopping

Perhaps the simplest way to prevent overfitting is called early stopping, which refers to the heuristic of stopping the training procedure when the error on the validation set first starts to increase. Early stopping works are due to restricting the ability of the optimization algorithm to transfer information from the training examples to the parameters. Early stopping can be viewed as a regularization in time. A training procedure like gradient descent will tend to learn increasingly complex functions as the number of iterations increases. By regularizing on time, the complexity of the model can be controlled, improving the generalization.

Dropout

Dropout can dramatically reduce overfitting and is very widely used in DNN. The reason dropout works well is that it prevents complex co-adaptation of the hidden units. Thus, each unit must learn to perform well even if some of the other units are missing at random. This prevents the units from learning complex, but fragile dependencies on each other.

5.3.4 Model selection and optimization

Model selection and model performance optimization contains a large number of approaches for hyper-parameter tuning such as grid search, random search, Bayesian optimization, and nature-inspired optimization. Model evaluation can be done with various performance metrics such as ACC, precision, recall, F1 score, MCC, PCC, ROC, AUC, MAE, MSE and RMSE.

Hyperparameter optimization

Every machine learning model has a set of parameters (like weights of neural network) and a set of hyper-parameters. A hyperparameter is a parameter whose value is set before the learning process begins. Hyper-parameters decide the structure of the model and the learning strategy. Because of the flexibility in model creation, they have to be

tuned carefully to achieve the best performance of the model and to avoid overfitting. Examples of hyper-parameters are: weight decay coefficient (L1, L2 regularization), batch size, learning rate, number and width of hidden layers.

Hyperparameter tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. Hyperparameter tuning is also called hyperparameter optimization. The hyperparameter tuning and model selection can be more generalized with more criteria on performance measurement such as dataset size(s), the model train complexity, and outside or environment setting.

In general, this can lead to the requirements of higher computational power and time lost, then we need optimization to reduce such effects. Hyperparameter optimization is a common problem in machine learning, where algorithms, in logistic regression to neural network, depend on well-tuned hyperparameters to reach maximum effectiveness. Different hyperparameter optimization strategies have varied performance and cost in time, money, and compute cycles, while evaluating optimization strategies is quite nonintuitive.

Model selection

It is common in machine learning to try multiple models and find one that works best for a particular problem. Given a dataset D, we need to choose a good model A^* from a set of potencial models applicable in the domain such that the function learned by A^* generalizes well. Each model A_i often has a set of hyperparameters, which can be set in a priori. A validation dataset is often used to find a good setting. In this context, the two simplest optimization strategies are grid search and random search.

Grid search

Hyper-parameters are parameters that are not directly learnt within model training. They are passed as training arguments. Grid search exhaustively considers all parameter combinations. It suggests parameter configurations deterministically, by laying down a grid of all possible configurations inside the parameter space. This strategy can lead to the curse of dimensionality (disadvantage). However, it is embarrassingly parallel (advantage).

Random search

Random search replaces the exhaustive search by selecting them randomly. It suggests configurations randomly from the parameter space. Random search has been shown to find equal or better values than grid search within fewer function evaluations for certain types of problems. Random search is embarrassingly parallel (advantage).

Bayersian optimization

In real world problems with large-scale data, we can have a problem to optimize a function $f(x)$, which is computational expensive to calculate and its derivative is also unknown. The quest is to find the global minima by an adaptive approach to parameter optimization. A trading off between exploring new areas of the parameter space and exploiting historical

information need to be done to find the parameters that maximize the function quickly.

Bayesian optimization builds a probabilistic model of the function mapping from hyper-parameters to the objective. Like random search, Bayesian optimization is stochastic. Its surrogate function $c(x)$ is formed based on sampled points. Based on the surrogate function, Bayesian optimization identifies which points are promising minima. The algorithm samples more from these promising regions and updates $c(x)$ accordingly. $c(x)$ is be mathematically expressed in a way that is significantly cheaper to evaluate $c(x)$ is usually represented by Gaussian process (probability distribution over possible functions).

5.4 Nature-inspired methods

An impressive variety of nature-inspired (metaheuristic) algorithms has been investigated and reported. For this many real-world optimization problems, no exact optimizer can be applied to solve them at an affordable computational cost or within a reasonable time, due to their complexity or the amount of data to use. In such cases, the use of traditional techniques has been widely proven to be unsuccessful, thereby calling for the consideration of alternative optimization approaches [53], [54]. The majority of the classical metaheuristic algorithms have been developed a long time ago such as Simulated Annealing (SA) [55], Genetic Algorithm (GA) [56], Differential Evolution (DE) [57], Particle Swarm Optimization (PSO) [58].

Despite the achievements of them, novel and improved evolutionary approaches have also emerged successfully in the last two decades with a great number of new metaheuristics inspired by evolutionary or behavioral processes. These new generation algorithms are often called nature-inspired or bio-inspired algorithms [59]. The whole of them can be classified into 2 main categories [60]:

- Single-solution-based algorithms (like SA) randomly generate a certain solution and gradually improve the solution until the best results are obtained. The main drawback of this technique is that it may be trapped into a local optimum, which prevents to determine the global optimum.
- Population-based algorithms (like GA and PSO) generate a set of random solutions within a given search space. These solutions are updated continuously until the best one is found out.

These algorithms can also be divided in another way into 4 main groups [61]:

- Evolutionary algorithms with the most mentioned GA, which mathematically mimics Darwinian's evolutionary laws with evolution strategies. DE also belongs to this group with its adaptive variants. The search process starts with randomly generated solutions that are continuously evolved throughout generations [62].
- Swarm-based algorithms or swarm intelligence refers to the collective behaviors of wild animals, e.g., birds, cats, bacteria, and mimics their social interactions. The optimizing process in these algorithms is mainly characterized by the ability to

explore based on the diversity of platforms and develop an exploitation based on searching around the best solution. The typical examples are PSO and its modern versions such as Ant Colony Optimization [63], Whale Optimization Algorithm [64]
- Physics-inspired algorithms mainly simulate physical phenomena that occur in nature by mathematical formulas or imitation of physical principles in the universe such as Galactic Swarm Optimization [65], Multi-Verse Optimization [66].
- Human-inspired algorithms are unique since they draw inspiration from several phenomena commonly associated with human behaviors, lifestyle, or perception such as Harmony Search [67], Firework Algorithm [68].

The optimization problems that attracted the attention of these approaches have a large variance, ranging from single-objective to multiobjective, continuous to discrete, constrained to unconstrained. Solving these problems is not a straightforward task due to their complex behavior. Nature-inspired algorithms provide a practical and elegant solution to many problems. They are designed to achieve approximately optimal solutions in practical execution times for NP-hard optimization problems [69].

5.5 Considerations about learning system design

Learning considerations have to be done about a training dataset, about the type of model, and about learning algorithms. Next considerations are to avoid overfitting and underfitting.

The training dataset plays a key role in the effectiveness of the system. The first consideration has to be about labelling such as do the observations have any labels. Furthermore, the training observations should characterize the whole data space for future predictions.
- How many observations are enough for learning?
- Does the size of the training dataset affect performance of the ML system?
- What is the effect of the disrupted or noisy observations?

Considerations about the type of a model have to be done towards the model representation and its complexity in the training and inference phase such as:
- Which algorithm performs best for a given application (or domain) and a given objective function. What is the convergence condition of the chosen algorithm?
- There's nothing like one size fits all. There is no one model that works best for every problem. The assumptions of a great model for one problem may not hold for another problem. It is common in machine learning to try multiple models and find one that works best for a particular problem.
- What about the generalization of the trained model? It has to predict well for new observations, not only the training data

The learning system design has to be evaluated with theoretical evaluation from the following viewpoints:

- the complexity of algorithms,
- requirements on input data,
- convergence conditions,
- expected output,
- noise and missing data tolerance of methods/algorithms.

The learning system design has to be evaluated with practical or experimental evaluation to have observation of the method performance in practical situations using prepared datasets and performed measurements. The aim is to make a summary of those experiments and to drive a conclusion.

Performance measurements are

- Accuracy: percentage of correct predictions on testing data.
- Efficiency : the cost in time and storage when learning/prediction.
- Robustness: the ability to reduce possible affected by noise/errors/missing.
- Scalability: the relation between performance and training size.
- Complexity: the complexity of the learned function.

5.6 Remark

Kevin P Murphy. *Probabilistic Machine Learning: An introduction*. Adaptive Computation and Machine Learning series, Thomas Dietterich (Editor), Christopher Bishop, David Heckerman, Michael Jordan, and Michael Kearns (Associate Editors). MIT Press, 2021. URL: `probml.ai`

Ian Goodfellow et al. *Deep Learning*. Adaptive Computation and Machine Learning series. MIT press, 2016. ISBN: 0262035618. URL: `https://www.deeplearningbook.org`

AIS ID: 103066

6. Intelligent software for data modeling

A number of machine learning algorithms, as well as their different software implementations, is quite large. Many software tools for data analytic using machine learning techniques have been in development for the past 25 years. Their common goal is to facilitate the complicated data analysis process and to propose integrated environments on top of standard programming languages. These tools are designed for various purposes: analytical platforms, predictive systems, recommender systems, processors (for images, sound, or language). A number of them are oriented to fast processing and streaming of large-scale data, while others are specialized in implementing machine learning algorithms including neural networks and deep learning.

It is important to emphasize that *there is no single tool suitable for every problem* and often a combination of them is needed to succeed. Figure 6.1 provides a comprehensive grouped overview of machine learning frameworks and libraries.

6.1 Machine learning tools

6.1.1 Scikit-learn

Scikit-learn is widely known as a popular open-source Python tool which contains a comprehensive library of machine learning and data analytic algorithms [30], [70], [71]. The scikit-learn project started as a Google Summer of Code project by David Cournapeau. Since 2015, it is under active development sponsored by INRIA, Telecom ParisTech and other companies like Google through the Google Summer of Code. It is

- Simple and efficient tools for predictive data analysis
- Well-updated and comprehensive set of algorithms and implementations.
- Accessible to everybody, and reusable in various contexts

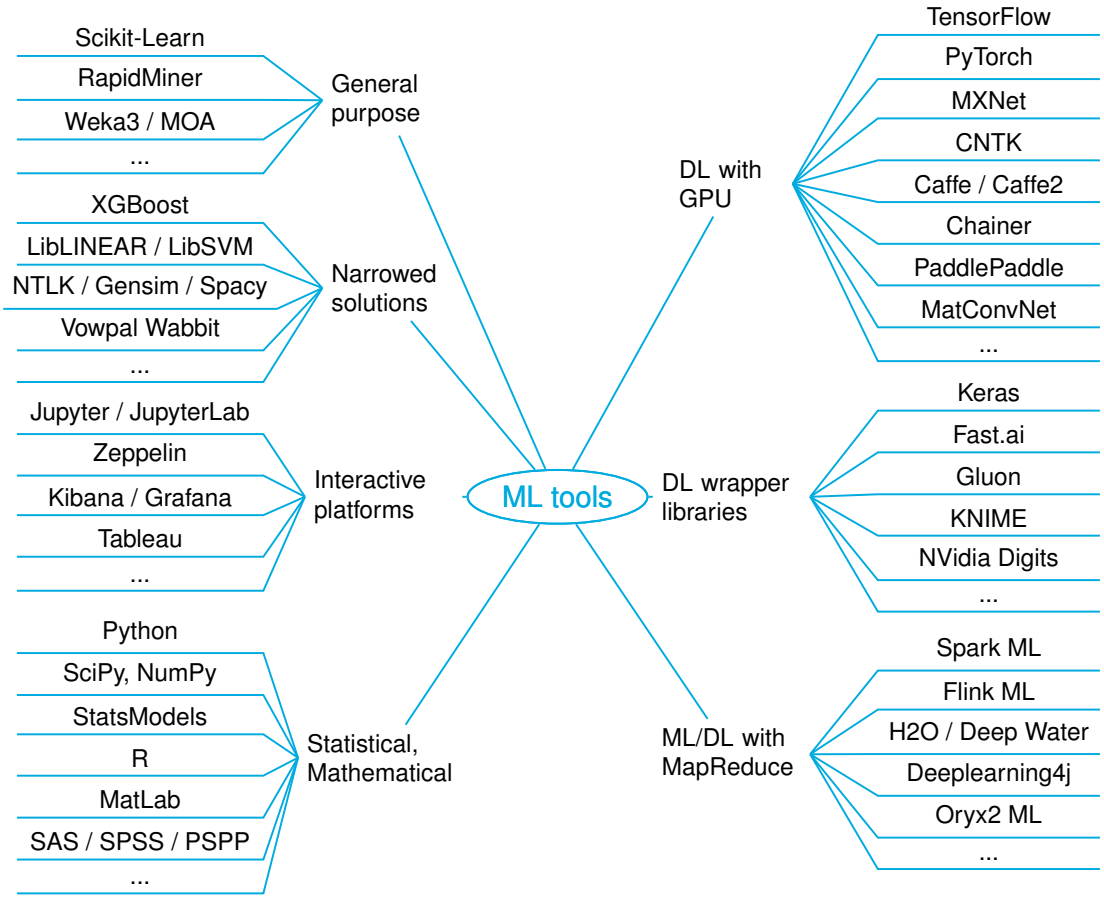


Figure 6.1: Overview of machine learning software frameworks and libraries.

- Built on NumPy, SciPy, and matplotlib
- It is a part of many ecosystems; it is closely coupled with statistics and scientific Python packages.
- Open source, commercially usable - BSD license

Scikit-learn supports supervised and unsupervised learning. It extends the functionality of NumPy and SciPy packages to numerous data mining algorithms and provides functions to perform classification, regression, clustering, dimensionality reduction, model selection, and preprocessing.

Scikit-learn also provides various tools for model fitting, data preprocessing, model selection and evaluation, and many other utilities. It uses the MatplotLib package for plotting charts.

6.1.2 Apache Spark and MLlib

Apache Spark [72] provides *Machine Learning Library (MLlib)* [73] built on top of Spark ecosystem. It provides common machine learning algorithms such as classification, regression, clustering, and collaborative filtering accompanied with featurization tools such as feature extraction, transformation, dimensionality reduction, and selection. Furthermore, pipeline tools for constructing, evaluating, and tuning are included. Model persistence is supported by tools for saving and loading algorithms, models, and pipelines. Utility tools are, for example, linear algebra, statistics, and data handling.

MLlib has strong features such as:

- Machine learning tools for large-scale data, which are already integrated in Apache Spark ecosystem, are convenient to use in development and production.
- Optimized selected algorithms with optimized implementations for Hadoop included preprocessing methods.
- Pipeline (workflow) building for Big Data processing included a set of feature engineering functions for data analytics also with stream data.
- Scalability with SQL support and very fast because of the in-memory processing.

MLlib is mainly focused to work with tabular data. The memory consumption is high because of the in-memory processing. MLlib is a part of the Apache Spark ecosystem, which is its strong as well as wide side.

6.2 Deep learning tools

6.2.1 Tensorflow

Tensorflow is an open-source software library for numerical computation using data flow graphs [74]. TensorFlow was created and is maintained by the Google Brain team within Google's Machine Intelligence research organization for machine learning and deep learning. It is currently released under the Apache 2.0 open-source license. Tensorflow is

- By far the most popular deep learning tool, open-source, fast evolving, supported by a strong industrial company (Google).
- Numerical library for dataflow programming that provides the basis for deep learning research and development.
- Efficiently works with mathematical expressions involving multidimensional arrays.
- GPU/CPU computing, efficient in multi-GPU settings, mobile computing, high scalability of computation across machines, and large data sets.

Many popular neural network and deep learning frameworks and libraries already offer the possibility to use GPU accelerators to speed up the learning process with supported interfaces. The software development in this direction is highly dynamic and has various layers of abstraction as depicted in Figure 6.2.

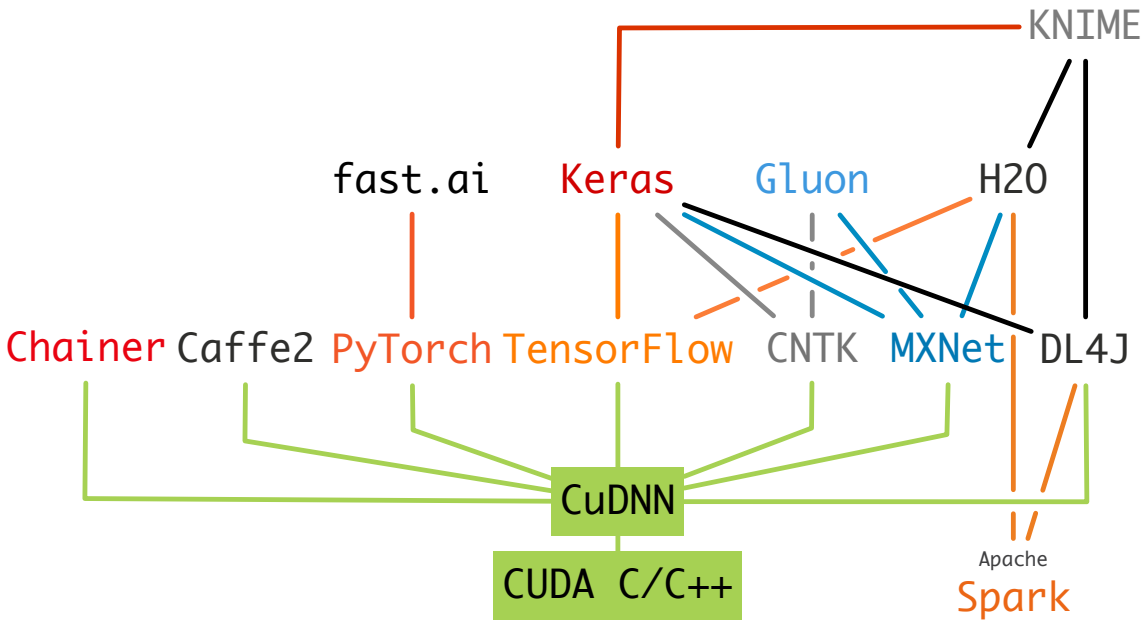


Figure 6.2: Deep learning frameworks and libraries.

6.2.2 Keras

Keras is Python wrapper library [75] that provides bindings to other DL tools such as TensorFlow and CNTK. It was developed with a focus on enabling fast experimentation and is released under the MIT license. Keras can seamlessly execute on GPUs and CPUs given the underlying framework. Keras is developed and maintained by Francois Chollet using four guiding principles:

1. *User friendliness and minimalism*. Keras is an API designed with user experience in mind. Keras follows best practices for reducing cognitive load by offering consistent and simple APIs.
2. *Modularity*. A model is understood as a sequence or a graph of standalone, fully-configurable modules that can be plugged together with as little restrictions as possible. In particular, neural layers, cost functions, optimizers, initialization schemes, activation functions, regularization schemes are all standalone modules to combine and to create new models.
3. *Easy extensibility*. New modules are simple to add, and existing modules provide ample examples allowing to reduce expressiveness.
4. *Work with Python*. Models are described in Python code, which is compact, easier to debug, and allows for ease of extensibility.

Keras Applications are deep learning models[1] that are made available alongside pretrained weights. These models can be used for prediction, feature extraction, and fine-tuning.

[1] https://keras.io/api/applications/

6.2.3 Pytorch

PyTorch is a Python library for GPU-accelerated deep learning. It has been developed by Facebook's AI Research lab (FAIR) since 2016. It declares itself as an open source machine learning framework that accelerates the path from research prototyping to production deployment [76]. PyTorch supports tensor computation with strong GPU acceleration, and DNNs built on a tape-based autograd system. It has become popular by allowing complex architectures to be built easily. Typically, changing the way a network behaves means to start from scratch. PyTorch uses a technique called *reverse-mode auto-differentiation*, which allows to change the way a network behaves with small effort, i.e. Dynamic Computational Graph (DCG).

Pytorch key features include:

- Production ready: transition seamlessly between eager and graph modes with TorchScript, and accelerate the path to production with TorchServe.
- Distributed training: scalable distributed training and performance optimization in research and production is enabled by the torch.distributed backend.
- Robust ecosystem: a rich ecosystem of tools and libraries, extends PyTorch and supports development in computer vision, NLP and more.
- Cloud support: PyTorch is well supported on major cloud platforms, providing frictionless development and easy scaling.

6.2.4 Fast.ai

In the context of Pytorch, Fast.ai library is often mentioned. Fast.ai is a deep learning library which provides practitioners with high-level components that can quickly and easily provide state-of-the-art results in standard deep learning domains, and provides researchers with low-level components that can be mixed and matched to build new approaches [77]. It aims to do both things without substantial compromises in ease of use, flexibility, or performance achieved by leveraging the dynamism of the underlying Python language and the flexibility of the PyTorch library.

The fast.ai library key features include:

- A new type dispatch system for Python along with a semantic type hierarchy for tensors with a new data block API
- A GPU-optimized computer vision library which can be extended in pure Python
- An optimizer which refactors the common functionality of modern optimizers into two basic pieces, allowing optimizations to be implemented in 4–5 lines of code
- A novel 2-way callback system that can access any part of the data, model, or optimizer and change it at any point during training

Fast.ai is organized around two main design goals: to be approachable and rapidly productive, while also being deeply hackable and configurable. It is built on top of a hierarchy of lower-level APIs which provide composable building blocks.

6.3 Accelerated computing

It is suitable to mention that deep learning makes profit of using a form of specialized hardware present in accelerated computing environments. The current mainstream solution has been to use GPU as general purpose processors. GPUs provide massive parallelism for large-scale DM problems, allowing scaling algorithms vertically to data of volumes that are not computable by traditional approaches. GPUs are effective solutions for real-world and real-time systems requiring very fast decision and learning, such as deep learning. Popular alternatives to GPUs include Field Programmable Gate Array (FPGA) and Tensor Processing Unit (TPU).

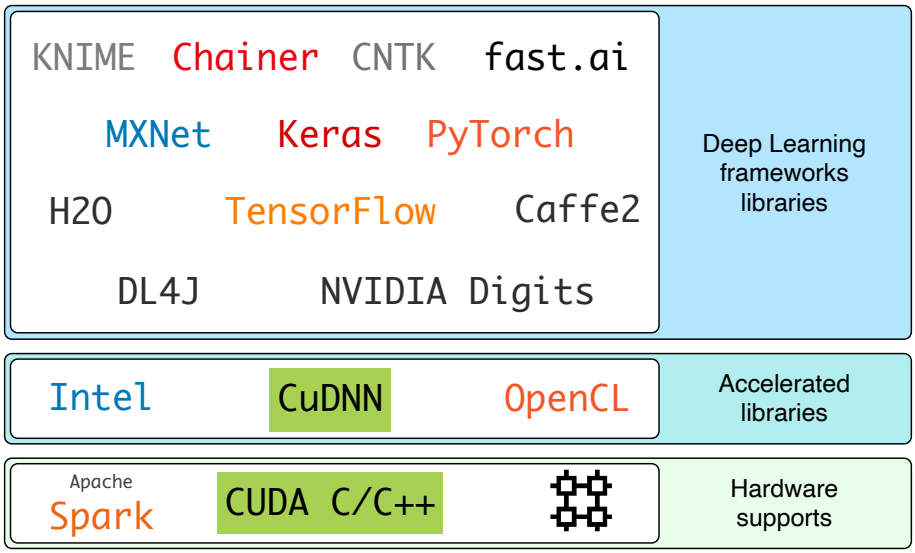


Figure 6.3: Accelerated computing and deep learning.

Other IT companies are starting to offer dedicated hardware for deep learning acceleration; e.g., Kalray with their second generation of deep learning acceleration device MPAA2-256 Bostan, oriented to mobile devices such as autonomous cars. The list of deep learning accelerator technology providers is quite long and includes worldwide IT companies and startups such as IBM TrueNorth Neuromorphic chip, Microsoft BrainWave, Intel Nervana Neural Network Processor, AMD Radeon Instinct.

The main feature of many-core accelerators such as GPU is their massively parallel architecture, allowing them to speed up computations that involve matrix-based operations, which are at heart of many deep learning implementations [78]. Manufacturers often offer the possibility to enhance hardware configuration with many-core accelerators to improve machine/cluster performance as well as *accelerated libraries*, which provide highly optimized primitives, algorithms and functions to access the massively parallel power of GPUs. The most frequently mentioned accelerated libraries are NVIDIA Compute Unified Device Architecture (CUDA), cuDNN, Intel Math Kernel Library (MKL), and

Open Computing Language (OpenCL)[2].

The NVIDIA CUDA is a parallel computing platform and programming model developed by NVIDIA for general computing on GPUs. GPU-accelerated CUDA libraries enable drop-in acceleration across multiple domains such as linear algebra, image and video processing, deep learning and graph analytic. The NVIDIA CUDA Toolkit provides a development environment for creating high performance GPU-accelerated applications. The NVIDIA CUDA DNN library (cuDNN), which is a GPU-accelerated library of DNNs primitives. The cuDNN provides highly tuned implementations for standard routines such as forward and backward convolution, pooling, normalization, and activation layers. It allows deep learning users to focus on training neural networks and developing software applications rather than spending time on low-level GPU performance tuning.

Intel MKL optimizes code with minimal effort for future generations of Intel processors. It is compatible with many compilers, languages, operating systems, and linking and threading models. It accelerates math processing routines, increases application performance, and reduces development time. This ready-to-use core math library includes: linear algebra (BLAS/OpenBLAS, LAPACK, ScaLAPACK), sparse solvers, Fast Fourier Transforms (FFT), vector statistics and data fitting, vector math and miscellaneous solvers.

It is suitable to mention the AMD ROCm (Radeon Open Compute platforM) open ecosystem. It is the AMD answer to the widespread NVIDIA CUDA. It features peer-to-peer multi-GPU operation with Remote Direct Memory Access (RDMA) support and the Heterogeneous System Architecture (HSA) and the Heterogeneous-compute Interface for Portability (HIP) which allows developers to convert CUDA code to common C++. HIP is very thin and has little or no performance impact over coding directly in CUDA or HCC. Because both CUDA and HIP are C++ languages, porting from CUDA to HIP is easier than porting from CUDA to OpenCL.

In comparison and compatible with cuDNN, MIOpen is AMD's open-source accelerated library for high performance machine learning primitives. Other programming libraries, which support computational speed-up and parallel computing, are OpenMP and Open-MPI (MPI). Applications built with the hybrid model of parallel programming can run on a computer cluster using both OpenMP and MPI, such that OpenMP is used for parallelism within a multicore node while MPI is used for parallelism between nodes.

Accelerators have to be adapted also to the new trends that emerge in the intelligent software development community. Several schemes that greatly benefit from specialized hardware have been devised for improving the speed and memory consumption of deep learning algorithms like sparse computation and low precision data types at the cost of a slightly less accurate model [79], [80]. Sparse computation imposes the use of sparse representations along the neural network. Benefits include lower memory requirements

[2]OpenCL provides compatibility across heterogeneous hardware from any vendor.

and faster computation with low precision data types [81], smaller than 32-bits (e.g. half-precision or integer). In these recent years, most deep learning frameworks are starting to support 16-bit and 8-bit computation [82], [83]. However, the vertical scalability of large-scale deep learning is still limited due to the GPU memory capacity, which is up to 80 GiB on the NVIDIA Ampere architecture at the end of 2021.

6.4 Distributed data processing and analytic

Here are several engines (or ecosystems) for Big Data distributed processing including ML libraries for data analysis such as Apache Spark and Apache Flink with Python as one of their programming languages. Such an ecosystem can run efficiently on a specialized cluster or a single machine. The installation of the Python software package in one machine is simple (`pip install pyspark`).

The advantages of Apache Spark are that it is applied to very large datasets, which do not fit into the memory of one machine. Usually, Apache Spark is utilized also as a whole ecosystem of data processing, data integration, and data management accompanied with built-in ML libraries. The disadvantages of Apache Spark are high cost, because it is typically coupled with infrastructure. Furthermore, network communication (including InfiniBand) is slower than GPUDirect RDMA, which provides direct communication between NVIDIA GPUs in remote systems. This eliminates the system CPU and the required buffer copies of data via the system memory, resulting in 10 times better performance. Apache Spark is mainly for text and numerical data, not for multimedia processing.

The next engine in this context is Apache Flink, which processes streaming data (stream of events) such as credit card transactions, sensor measurements, machine logs, or user interactions on a website or mobile application. It is less popular than Apache Spark. The installation in one machine is simple (`pip install apache-flink`).

At first sight, the general audience relates Big Data processing on distributed platforms like above-mentioned engines in this subsection. The contiguity goes there for Volume in conformity with Veracity as well as high speeds in processing and inference in background. In the new era of Big Data, data analysis is expected to change. The nature of large-scale data requires new approaches and new tools that can accommodate them with different data structures, different spatial and temporal scales. The surge of large Volume of information, especially with the Variety characteristic, to be processed by data mining and machine learning algorithms demand new transformative parallel and distributed computing solutions capable to scale computation effectively and efficiently. That is the reason of accelerated computing rising with GPU as the specialized hardware for speeding up general purpose computation in the last decade (Section 6.3) [78], [84].

6.5 Interactive data analytic and visualization tools

There are also open-source tools for data analytics, reporting and integration platforms, which are designed for different purposes such as:

- `Kibana` is the data visualisation front end for the Elastic Stack, complementing the rest of the stack that includes Beats, Logstash, and Elasticsearch.
- `Grafana` is the DevOps tool for many real-time monitoring dashboards of time series metrics. It offers visualisation and supports multiple backend data sources including InfluxDB, Graphite, Elasticsearch, and many others, which can be added via plugins.
- `Tableau` is a universal analytic tool, which can extract data from different small data sources like csv, excel, and Structured Query Language (SQL) as well as from enterprise resources or connected Big Data frameworks and cloud-based sources.

In recent years, web-based notebooks/applications have gained in popularity. They are integrated with data analytic environments to create and share documents that contain data-driven live code, equations, visualizations, and narrative text. The most well-known is `Jupyter` notebook [85] and its variances `JupyterLab`, `JupyterHub` or JetBrain `DataSpell` integrated development environment (IDE) for data scientists. It is an open-source application supporting; e.g., creation and sharing documents (notebooks), code, source equations, visualization and text descriptions for data transformation, numerical simulations, statistical modeling, data visualization and machine learning.

6.6 Remark

Alberto Cano. "A survey on graphic processing unit computing for large-scale data mining". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8.1 (2018). DOI: 10.1002/widm.1232

Giang Nguyen et al. "Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey". In: *Artificial Intelligence Review* 52.1 (2019). Springer Nature research highlights in Computer Science., pp. 77–124. ISSN: 0269-2821. DOI: 10.1007/s10462-018-09679-z

AIS ID: 103066

7. Advanced topics in Data Science and AI

7.1 Responsible AI and Ethics in AI development

In the last decade, AI has achieved a notable momentum over the best expectations over many applications in many sectors. Accompanying with this success, here is also the rising barrier of explainability of the latest techniques such as deep neural network. While the very first AI systems were easily interpretable (e.g., decision trees), the recent decision systems built with deep learning in core is not so easy to explain. The empirical success of deep learning comes from a combination of efficient learning algorithms and their huge parametric space of hundreds of layers and millions of neurons and parameters. This situation makes deep neural network models considered as black-box models. The opposite of the black-box-ness way is transparency, i.e., the search for a direct understanding of the mechanism by which a model works. The desired models have to be *glass-box* models, which are built based on the concept of responsible AI [86].

> Responsible AI is a methodology for the large-scale implementation of AI methods in real organizations with fairness, model explainability, and accountability at its core.

In this context, Explainable AI (XAI) is widely acknowledged as a crucial feature for the practical deployment of AI models. It aims to produce more explainable models while maintaining a high level of learning performance and to enable humans to understand, appropriately trust, and effectively manage the emerging generation of AI partners.

In 2017, the initiative of the Université de Montréal lead to the Montreal Declaration for a Responsible Development of AI. The declaration aims to spark public debate and encourage a progressive and inclusive orientation to the development of AI [87].

> Every person involved in AI development must exercise caution by anticipating, as far as possible, the adverse consequences of AI Systems use and by taking appropriate measures to avoid them.

In 2018, IBM's Principles for Trust and Transparency[1] states that the purpose of AI is to augment human intelligence. The IEEE Ethically Aligned Design[2] series was created to highlight specific aspects of prioritizing human wellbeing with autonomous and intelligent systems (A/IS). The document was created by global experts focused on the pragmatic instantiation of human-centric, value-driven design since 2015 to 2018 [87], [88]. The IEEE report also lays down principles to guide the ethical and value-based design, development, and implementation of A/IS. A/IS creators shall adopt increased human well-being as a primary success criterion for development. It emphasises the necessity of involving actors from different levels of the AI ecosystem and neighboring areas to ensure that experts in policy-making work in tandem with experts in coding and engineering to create tools and frameworks that are coherent and usable by all.

Finally, the *European Union guidelines on ethics in AI: Context and implementation* [88] states:

> The human-centric approach to AI strives to ensure that human values are central to the way in which AI systems are developed, deployed, used, and monitored, by ensuring respect for fundamental rights, including those set out in the Treaties of the European Union and Charter of Fundamental Rights of the European Union, all of which are united by reference to a common foundation rooted in respect for human dignity, in which the human being enjoys a unique and inalienable moral status. This also entails consideration of the natural environment and of other living beings that are part of the human ecosystem, as well as a sustainable approach enabling the flourishing of future generations to come.

The guideline specifies the following key ethical requirements: 1) Ethical requirements, 2) Human agency and oversight, 3) Technical robustness and safety, 4) Privacy and data protection (GDPR), 5) Transparency, 6) Diversity, non-discrimination and fairness, 7) Societal and environmental well-being, and 8) Accountability.

The European Commission's report *"Ethics guidelines for trustworthy AI"* provides a clear benchmark to evaluate the responsible development of AI systems, and facilitates international support for AI solutions that are good for humanity and the environment [89]. AI is revolutionizing everyone's life, and it is crucial that it does so in the right way. AI's profound and far-reaching potential for transformation concerns the engineering of systems that have some degree of autonomous agency. This is epochal and requires establishing a new ethical balance between human and artificial autonomy [90].

[1] https://www.ibm.com/blogs/policy/trust-principles/
[2] https://ethicsinaction.ieee.org/

European Commission. *EU guidelines on ethics in artificial intelligence: Context and implementation*. Accessed 1 Oct 2021. 2019. URL: https://www.europarl.europa.eu/thinktank/en/document.html?reference=EPRS_BRI(2019)640163

European Commission. *Ethics guidelines for trustworthy AI*. Accessed 1 Oct 2021. 2019. URL: https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai

7.2 Beyond supervised learning

Today, the majority of mentioned AI transformative potential is refering to machine learning and the most of the groundbreaking progress in machine learning is probably referring to deep learning, particularly supervised deep learning. In general, AI is not a magic, but its core is clever algorithm(s) built based on applied statistics and algebra. Since manual feature generation tends to be hard and brittle, the AI community is increasingly embraced deep learning, which can learn distinctive properties from data by themselves from a vast amount of usually labelled/annotated data. This effect leads to a significant worldwide issue of high-quality data for machine learning and deep learning as well as data annotation, which are proven to be costly and undertaken even for large organizations. The unsupervised learning paradigm, i.e., using data without labels, should help with the issue, but however, unsupervised machine learning hasn't been able to extract meaningful information from many real-life scenarios with unsupervised methods.

Self-supervised learning is one of those recent machine learning techniques that have made waves in the Data Science community. This way of machine learning may be the future of AI, according to several of the most prominent AI research leaders.

> "I think self-supervised learning is the future. This is what's going to allow to our Artificial Intelligence systems, deep learning system to go to the next level, perhaps learn enough background knowledge about the world by observation, so that some sort of common sense may emerge." *(Geoff Hinton, **Yann Lecun**, Yoshua Bengio: Keynotes Turing Award Winners Event, AAAI conference, 2020.)*

This fairly technique attempts to extract supervisory information directly from the input data and thus it requires no human involvement or intervention. Self-supervised learning works well where algorithms can rely on context for supervision while creating representations based on input data. Another technique, which can also alleviate the need for labeled data by taking advantage of unlabeled data is semi-supervised learning, transfer learning, fine-tuning and domain adaptation.

Stuart Russell et al. *Artificial intelligence: a modern approach, Fourth Edition*. Pearson; 4th edition (April 14, 2021), 2021. ISBN: 978-0134610993

AIS ID: 103066

Abbreviations

A/IS	autonomous and intelligent systems
AAAI	Association for the Advancement of Artificial Intelligence
ACC	Accuracy
Adam	Adaptive Moment Estimation Algorithm
AI	Artificial Intelligence
ANOVA	Analysis of Variance
ASUM	Analytics Solutions Unified Method
AUC	Area Under the Curve
BI	Business Intelligence
BoW	Bag-of-Words
CCPA	California Consumer Privacy Act
CI/CD	continuous integration and continuous delivery
CNN	Convolutional Neural Network
CRISP-DM	Cross-Industry Process for Data Mining
CTR	Click-Through-Rate
CUDA	Compute Unified Device Architecture
DCG	Dynamic Computational Graph
DE	Differential Evolution
DevOps	development and operations
DL	deep learning
DM	data mining
DNN	deep neural network
DS	Data Science
EC	European Commission

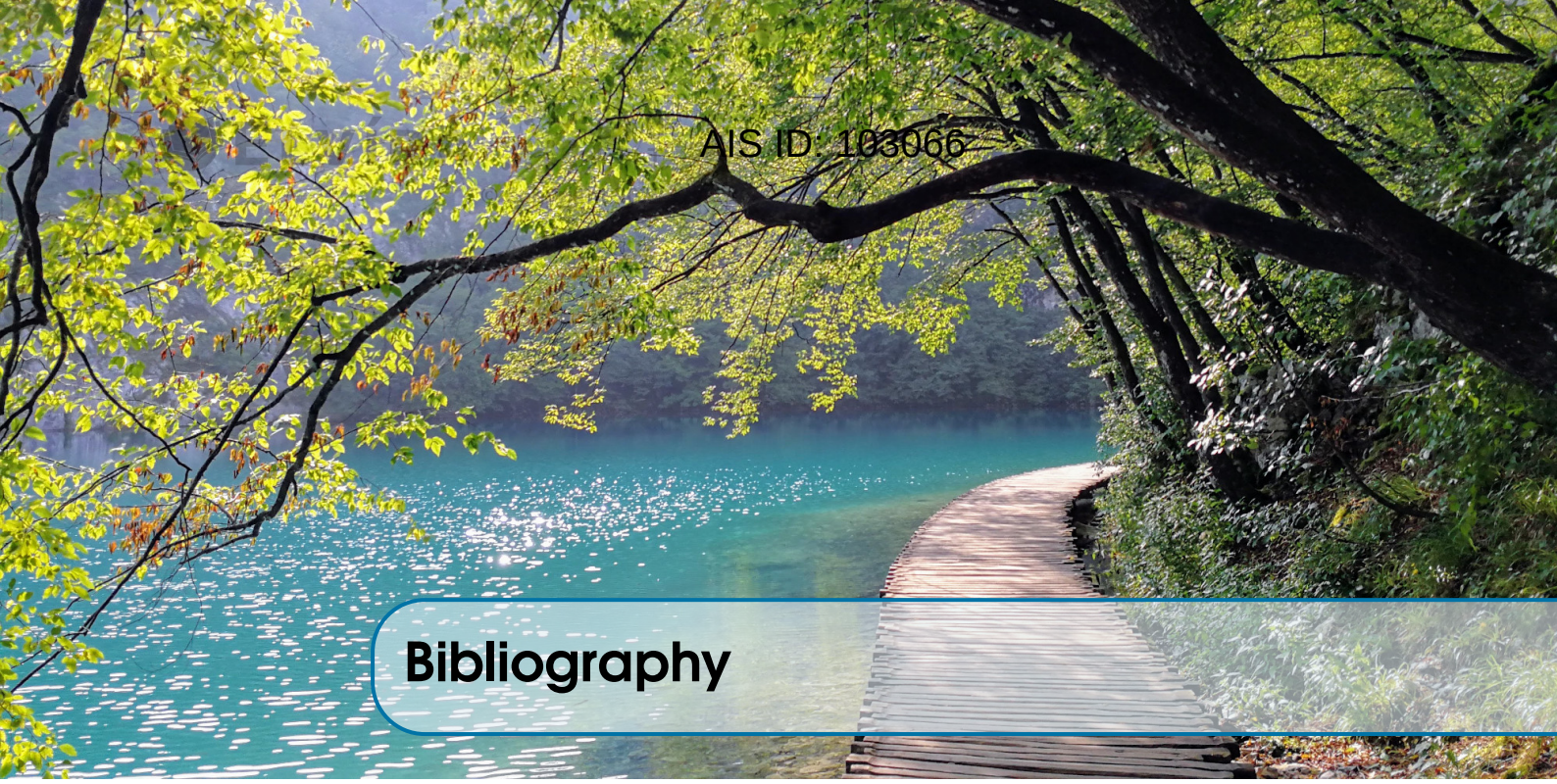
EDA	Exploratory Data Analysis
EU	European Union
FC	Fully Connected Networks
FFT	Fast Fourier Transforms
FN	False Negatives
FP	False Positives
FPGA	Field Programmable Gate Array
FPR	False Positive Rate
GA	Genetic Algorithm
GAN	Generative Adversarial Networks
GDPR	General Data Protection Regulation
GPS	the global positioning system
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HMM	Hidden Markov Models
ID3	Iterative Dichotomiser 3
IDE	integrated development environment
IG	Information Gain
IID	independent and identically distributed
IoT	Internet of Things
IQR	interquartile range
KDD	knowledge discovery in databases
kNN	k-Nearest Neighbors algorithm
Lasso	Least Absolute and Selection Operator
LDA	Linear Discriminant Analysis
LLF	log-likelihood function
LR	Logistic regression
LSA	Latent Semantic Analysis
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MCC	Matthews Correlation Coefficient
MDS	MultiDimensional Scaling
MI	Mutual Information
MKL	Math Kernel Library
ML	machine learning
MLE	Maximum Likelihood Estimation
MLOps	machine learning operations
MSE	Mean Squared Error
NaN	Not A Number
NB	Naïve Bayes

NLP	natural language processing
NLTK	Natural Language Toolkit
NMT	neural machine translation
NN	neural network
NP-hard	non-deterministic polynomial-time hardness
OLS	Ordinary Least Squares
OOP	Object-Oriented Programming
OpenCL	Open Computing Language
PCA	Principal Components Analysis
PCC	Pearson Correlation Coefficient
PII	Personally Identifiable Information
PSO	Particle Swarm Optimization
QA	quality assurance
R2	Coefficient of Determination
ReLU	Rectified Linear Unit
RF	Random Forests
RFE	Recursive Feature Elimination
RMSE	Root Mean Square Error
RMSprop	Root Mean Square Propagation
RNN	Recurrent Neural Network
ROC	Receiver Operator Characteristic Curve
RSE	Residual Standard Error
RSS	Residual Sum of Squares
SA	Simulated Annealing
SAS	Statistical Analysis System
seq2seq	sequence to sequence
SGD	Stochastic Gradient Descent
SMAPE	Symmetric Mean Absolute Percentage Error
SMOTE	Synthetic Minority Over-sampling Technique
SPSS	Statistical Package for the Social Sciences
SQL	Structured Query Language
SVD	Singular Value Decomposition
SVM	Support Vector Machine
t-SNE	t-distributed Stochastic Neighbor Embedding
TN	True Negatives
TP	True Positives
TPR	True Positive Rate
TPU	Tensor Processing Unit
TRN	True Negative Rate
XAI	Explainable AI

AIS ID: 103066

List of Figures

AIS ID: 103066



Bibliography

[1] Anthony JG Hey, Stewart Tansley, Kristin Michele Tolle, et al. *The fourth paradigm: data-intensive scientific discovery*. Vol. 1. Microsoft research Redmond, WA, 2009. ISBN: 978-0-9825442-0-4.

[2] Rachel Schutt and Cathy O'Neil. *Doing data science - Straight talk from the frontline*. " O'Reilly Media, Inc.", 2014. ISBN: 978-1449358655.

[3] Gabriel Peyre. *Mathematical Foundations of Data Sciences*. https://mathematical-tours.github.io. CNRS and DMA, Ecole Normale Superieure, 2019. URL: `https://mathematical-tours.github.io`.

[4] Stuart Russell and Peter Norvig. *Artificial intelligence: a modern approach, Fourth Edition*. Pearson; 4th edition (April 14, 2021), 2021. ISBN: 978-0134610993.

[5] Foster Provost and Tom Fawcett. *Data Science for Business: What you need to know about data mining and data-analytic thinking*. " O'Reilly Media, Inc.", 2013. ISBN: 978-1-449-36132-7.

[6] *CRISP-DM Cross Industry Standard Process for Data Mining*. Accessed 10 Oct 2021. 1999. URL: `http://cordis.europa.eu/project/rcn/37679_en.html`.

[7] *Analytics Solutions Unified Method (ASUM) for Data Mining/Predictive Analytics*. Accessed 10 Oct 2021. 2016. URL: `ftp://ftp.software.ibm.com/software/data/sw-library/services/ASUM.pdf`.

[8] *What is Data Privacy?* Accessed 10 Oct 2021. 2021. URL: `https://www.snia.org/education/what-is-data-privacy`.

[9] *General Data Protection Regulation*. Accessed 10 Oct 2021. 2016. URL: `https://eur-lex.europa.eu/eli/reg/2016/679/oj`.

[10] *What is GDPR?* Accessed 10 Oct 2021. 2020. URL: `https://www.wired.co.uk/article/what-is-gdpr-uk-eu-legislation-compliance-summary-fines-2018`.

[11] *California Consumer Privacy Act*. Accessed 10 Oct 2021. 2018. URL: https://oag.ca.gov/privacy/ccpa.

[12] *Python Software Foundation - The official home of the Python Programming Language*. Accessed 10 Oct 2021. 2021. URL: https://www.python.org.

[13] Zed A Shaw. *Learn python 3 the hard way: A very simple introduction to the terrifyingly beautiful world of computers and code*. Addison-Wesley Professional, 2017. ISBN: 978-0134692883.

[14] Tim Peters. *The Zen of Python*. Accessed 10 Oct 2021. 2004. URL: https://www.python.org/dev/peps/pep-0020/.

[15] *DevelopIntelligence - Python ecosystem in 2020*. Accessed 10 Oct 2021. 2021. URL: https://www.developintelligence.com/the-python-ecosystem-of-2020/.

[16] Ján Paralič, K Furdík, G Tutoky, P Bednár, M Sarnovskỳ, P Butka, and F Babič. *Dolovanie znalostí z textov*. Equilibria, Košice, 2010. URL: http://people.tuke.sk/jan.paralic/knihy/DolovanieZnalostizTextov.pdf.

[17] Giang Nguyen, Stefan Dlugolinsky, Viet Tran, and Álvaro López García. "Deep learning for proactive network monitoring and security protection". In: *IEEE Access* 8 (2020). Special section on Deep Learning: Security and Forensics Research Advances and Challenges, pp. 1–21. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2968718.

[18] George Casella and Roger L Berger. *Statistical inference*. Cengage Learning, 2002. ISBN: 0-534-24312-6.

[19] Martina Litschmannová. *Úvod do statistiky*. Vysoká škola báňská-Technická univerzita Ostrava, 2011. URL: https://dspace.vsb.cz/handle/10084/139339.

[20] Kevin P Murphy. *Probabilistic Machine Learning: An introduction*. Adaptive Computation and Machine Learning series, Thomas Dietterich (Editor), Christopher Bishop, David Heckerman, Michael Jordan, and Michael Kearns (Associate Editors). MIT Press, 2021. URL: probml.ai.

[21] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. "Estimating mutual information". In: *Physical Review E* 69.6 (2004). ISSN: 1539-3755. DOI: 10.1103/PhysRevE.69.066138.

[22] Brian C. Ross. "Mutual information between discrete and continuous data sets". In: *PLoS ONE* 9.2 (2014), e87357. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0087357.

[23] Jorge Gonzalez-Lopez, Sebastián Ventura, and Alberto Cano. "Distributed Selection of Continuous Features in Multilabel Classification Using Mutual Information". In: *IEEE Transactions on Neural Networks and Learning Systems* (2019). DOI: 10.1109/TNNLS.2019.2944298.

[24] *SciPy: open-source software for mathematics, science, and engineering with Python*. Accessed 10 Oct 2021. 2021. URL: https://www.scipy.org/.

[25] *NumPy: The fundamental package for scientific computing with Python*. Accessed 10 Oct 2021. 2021. URL: https://numpy.org/.

[26] *StatsModels: statistical models, hypothesis tests, and data exploration with Python*. Accessed 10 Oct 2021. 2021. URL: https://www.statsmodels.org/stable/index.html.

[27] *Pandas: open source data analysis and manipulation tool with Python*. Accessed 10 Oct 2021. 2021. URL: https://pandas.pydata.org/.

[28] *Matplotlib: Visualization with Python*. Accessed 10 Oct 2021. 2021. URL: https://pandas.pydata.org/.

[29] *Seaborn: statistical data visualization*. Accessed 10 Oct 2021. 2021. URL: https://seaborn.pydata.org/.

[30] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. "Scikit-learn: Machine learning in Python". In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[31] Pavol Návrat, Ľubica Beňušková, Mária Bieliková, I Kapustník, Gabriela Kosková, and Jiří Pospíchal. *Umelá inteligencia*. Bratislava: Vydavatel'stvo STU, 2015. ISBN: 978-80-227-4344-0.

[32] Elena Šikudová, Zuzana Černeková, Wanda Benešová, Zuzana Haladová, and Júlia Kučerová. *Počítačové videnie Detekcia a rozpoznávanie objektov*. Praha: Wikina,Praha, 2013, p. 397. ISBN: 978-80-87925-06-5.

[33] Dan Jurafsky and James H Martin. *Speech and language processing. Vol. 3*. US: Prentice Hall, 2014. ISBN: 978-0131873216. URL: https://web.stanford.edu/~jurafsky/slp3/.

[34] Ján Paralič. *Objavovanie znalostí v databázach*. Elfa, Košice, 2003. URL: http://people.tuke.sk/jan.paralic/knihy/ObjavovanieZnalostivDB.pdf.

[35] Kristína Machová. *Strojové učenie v systémoch spracovania informácií*. Elfa, Košice, 2010. URL: http://www.tuke.sk/machova/pdf/monoSUvSSI.pdf.

[36] Kristína Machová. *Strojové učenie: Princípy a algoritmy*. Elfa, Košice, 2002. URL: http://www.tuke.sk/machova/pdf/SU4.pdf.

[37] Vladimír Kvasnička, Ľubica Beňušková, Jiří Pospíchal, Igor Farkaš, Peter Tiňo, and Andrej Král'. *Úvod do teórie neurónovỳch sietí*. 1997.

[38] Peter Sinčák and Gabriela Andrejková. *Neurónové siete Inžiniersky prístup, 1. diel*. Elfa, Košice, 1996. URL: http://kuzo.szm.com/neuronky1.pdf.

[39] Peter Sinčák and Gabriela Andrejková. *Neurónové siete Inžiniersky prístup, 2.diel*. Elfa, Košice, 1996. URL: https://github.com/ianmagyar/neural-networks-course/blob/master/lectures/Neuronove_siete_2.pdf.

[40] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Adaptive Computation and Machine Learning series. MIT press, 2016. ISBN: 0262035618. URL: https://www.deeplearningbook.org.

[41] *Deep Learning Tutorial*. University of Montreal, LISA lab, 2015.

[42] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[43] George Cybenko. "Approximation by superposition of sigmoidal functions". In: *Mathematics of Control, Signals and Systems* 2.4 (1989), pp. 303–314.

[44] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555* (2014). URL: https://arxiv.org/abs/1412.3555.

[45] Jurgen Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural networks* 61 (2015), pp. 85–117.

[46] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[47] Christopher Olah. *Understanding LSTM Networks*. Accessed 9 Sept 2019. 2015. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[48] Mike Schuster and Kuldip K Paliwal. "Bidirectional recurrent neural networks". In: *IEEE Transactions on Signal Processing* 45.11 (1997), pp. 2673–2681. ISSN: 1053587X. DOI: 10.1109/78.650093.

[49] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. "Unsupervised learning of video representations using lstms". In: *International conference on machine learning*. 2015, pp. 843–852. URL: http://www.jmlr.org/proceedings/papers/v37/srivastava15.pdf.

[50] Thang Luong, Hieu Pham, and Christopher D. Manning. "Effective Approaches to Attention-based Neural Machine Translation". In: (2015), pp. 1412–1421. DOI: 10.18653/v1/D15-1166.

[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008. URL: http://papers.nips.cc/paper/7181-attention-is-all-you-need.

[52] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion". In: *Journal of machine learning research* 11.Dec (2010), pp. 3371–3408. URL: http://www.jmlr.org/papers/v11/vincent10a.html.

[53] Ilhem Boussaïd, Julien Lepagnot, and Patrick Siarry. "A survey on optimization metaheuristics". In: *Information sciences* 237 (2013), pp. 82–117. DOI: 10.1016/j.ins.2013.02.041.

[54] Sedigheh Mahdavi, Mohammad Ebrahim Shiri, and Shahryar Rahnamayan. "Metaheuristics in large-scale global continues optimization: A survey". In: *Information Sciences* 295 (2015), pp. 407–428. DOI: 10.1016/j.ins.2014.10.042.

[55] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. "Optimization by simulated annealing". In: *science* 220.4598 (1983), pp. 671–680. DOI: 10.1126/science.220.4598.671.

[56] Srinivas Mandavilli and Lalit M Patnaik. "Genetic algorithms: A survey". In: *computer* 27.6 (1994), pp. 17–26. DOI: 10.1109/2.294849.

[57] Rainer Storn and Kenneth Price. "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces". In: *Journal of global optimization* 11.4 (1997), pp. 341–359. DOI: 10.1023/A:1008202821328.

[58] Federico Marini and Beata Walczak. "Particle swarm optimization (PSO). A tutorial". In: *Chemometrics and Intelligent Laboratory Systems* 149 (2015), pp. 153–165.

[59] Daniel Molina, Javier Poyatos, Javier Del Ser, Salvador García, Amir Hussain, and Francisco Herrera. "Comprehensive Taxonomies of Nature-and Bio-inspired Optimization: Inspiration Versus Algorithmic Behavior, Critical Analysis Recommendations". In: *Cognitive Computation* 12.5 (2020), pp. 897–939. DOI: 10.1007/s12559-020-09730-8.

[60] El-Ghazali Talbi. *Metaheuristics: from design to implementation*. Vol. 74. John Wiley and Sons, 2009. ISBN: 978-0-470-27858-1.

[61] Tansel Dokeroglu, Ender Sevinc, Tayfun Kucukyilmaz, and Ahmet Cosar. "A survey on new generation metaheuristic algorithms". In: *Computers and Industrial Engineering* 137 (2019), p. 106040. DOI: 10.1016/j.cie.2019.106040.

[62] Adam P Piotrowski. "Review of differential evolution population size". In: *Swarm and Evolutionary Computation* 32 (2017), pp. 1–24.

[63] Marco Dorigo and Thomas Stützle. "Ant colony optimization: overview and recent advances". In: *Handbook of metaheuristics*. Springer, 2019, pp. 311–351. DOI: 10.1007/978-3-319-91086-4_10.

[64] Seyedali Mirjalili and Andrew Lewis. "The whale optimization algorithm". In: *Advances in engineering software* 95 (2016), pp. 51–67.

[65] Venkataraman Muthiah-Nakarajan and Mathew Mithra Noel. "Galactic Swarm Optimization: A new global optimization metaheuristic inspired by galactic motion". In: *Applied Soft Computing* 38 (2016), pp. 771–787. DOI: 10.1016/j.asoc.2015.10.034.

[66] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Abdolreza Hatamlou. "Multiverse optimizer: a nature-inspired algorithm for global optimization". In: *Neural Computing and Applications* 27.2 (2016), pp. 495–513.

[67] Zong Woo Geem, Joong Hoon Kim, and Gobichettipalayam Vasudevan Loganathan. "A new heuristic optimization algorithm: harmony search". In: *simulation* 76.2 (2001), pp. 60–68. DOI: 10.1177/003754970107600201.

[68] Ying Tan and Yuanchun Zhu. "Fireworks algorithm for optimization". In: *International conference in swarm intelligence*. Springer. 2010, pp. 355–364. DOI: 10.1007/978-3-642-13495-1_44.

[69] Frank Neumann and Carsten Witt. "Combinatorial optimization and computational complexity". In: *Bioinspired computation in combinatorial optimization*. Springer, 2010, pp. 9–19. DOI: 10.1007/978-3-642-16544-3_2.

[70] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. "API

design for machine learning software: experiences from the scikit-learn project". In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, pp. 108–122.

[71] *scikit-learn: machine learning in Python*. Accessed 10 Oct 2021. 2021. URL: https://scikit-learn.org/stable/.

[72] *Apache Spark | fast and general engine for large-scale data processing*. Accessed 10 Oct 2021. 2021. URL: https://spark.apache.org/.

[73] *Machine Learning Library (MLlib) Guide*. Accessed 10 Oct 2021. 2021. URL: https://spark.apache.org/docs/latest/ml-guide.html.

[74] *TensorFlow | An open-source software library for Machine Intelligence*. Accessed 10 Oct 2021. 2021. URL: https://www.tensorflow.org/.

[75] *Keras | High-level neural networks API*. Accessed 10 Oct 2021. 2021. URL: https://keras.io/.

[76] *PyTorch | from research to production*. Accessed 10 Oct 2021. 2021. URL: http://pytorch.org/.

[77] *fast.ai · Making neural nets uncool again*. Accessed 10 Oct 2021. 2021. URL: https://www.fast.ai/.

[78] Alberto Cano. "A survey on graphic processing unit computing for large-scale data mining". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8.1 (2018). DOI: 10.1002/widm.1232.

[79] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and less than 0.5 MB model size". In: *arXiv preprint arXiv:1602.07360* (2016).

[80] Stefano Markidis, Steven Wei Der Chien, Erwin Laure, Ivy Bo Peng, and Jeffrey S Vetter. "NVIDIA Tensor Core Programmability, Performance and Precision". In: *arXiv preprint arXiv:1803.04014* (2018).

[81] Patrick Konsor. *Intel software | Developer zone | Performance Benefits of Half Precision Floats, Intel Software Development Zone*. Accessed 10 Oct 2021. Aug. 2012. URL: https://software.intel.com/en-us/articles/performance-benefits-of-half-precision-floats.

[82] Mark Harris. *Mixed-Precision Programming with CUDA 8*. Accessed 10 Oct 2021. 2016. URL: https://devblogs.nvidia.com/mixed-precision-programming-cuda-8/.

[83] Andres Rodriguez, Eden Segal, Etay Meiri, Evarist Fomenko, Young Jin Kim, Haihao Shen, and Barukh Ziv. *Lower Numerical Precision Deep Learning Inference and Training*. Accessed 10 Oct 2021. 2018. URL: https://software.intel.com/en-us/articles/lower-numerical-precision-deep-learning-inference-and-training.

[84] Giang Nguyen, Stefan Dlugolinsky, Martin Bobák, Viet Tran, Álvaro López García, Ignacio Heredia, Peter Malík, and Ladislav Hluchý. "Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey". In: *Artifi-*

cial Intelligence Review 52.1 (2019). Springer Nature research highlights in Computer Science., pp. 77–124. ISSN: 0269-2821. DOI: 10.1007/s10462-018-09679-z.

[85] *Project Jupyter*. Accessed 10 Oct 2021. 2021. URL: https://jupyter.org/.

[86] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI". In: *Information Fusion* 58 (2020), pp. 82–115. DOI: 10.1016/j.inffus.2019.12.012.

[87] Alexandra Luccioni and Yoshua Bengio. "On the morality of artificial intelligence [Commentary]". In: *IEEE Technology and Society Magazine* 39.1 (2020), pp. 16–25. URL: https://technologyandsociety.org/on-the-morality-of-artificial-intelligence/.

[88] European Commission. *EU guidelines on ethics in artificial intelligence: Context and implementation*. Accessed 1 Oct 2021. 2019. URL: https://www.europarl.europa.eu/thinktank/en/document.html?reference=EPRS_BRI(2019)640163.

[89] Luciano Floridi. "Establishing the rules for building trustworthy AI". In: *Nature Machine Intelligence* 1.6 (2019), pp. 261–262. DOI: 10.1038/s42256-019-0055-y.

[90] European Commission. *Ethics guidelines for trustworthy AI*. Accessed 1 Oct 2021. 2019. URL: https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai.

Giang Nguyen

Introduction to Data Science

First edition, 6.5 AH, 2022

The edition of university textbooks in Informatics and Information Technologies

Published in Slovak University of Technology in Bratislava in SPEKTRUM STU Publishing

SPEKTRUM STU

ISBN 978-80-227-5193-3

This publication has been written thanks to the support of the Operational Programme Integrated Infrastructure for the project: International Center of Excellence for Research on Intelligent and Secure Information and Communication Technologies and Systems (ITMS code: 313021W404), co-funded by the European Regional Development Fund (ERDF).

Giang Nguyen

Introduction to Data Science

The edition of university textbooks in Informatics and Information Technologies

Published in Slovak University of Technology in Bratislava in SPEKTRUM STU Publishing

SPEKTRUM STU

ISBN 978-80-227-5193-3