POLITECNICO DI TORINO

Master's Degree in Engineering and Management

Master's Degree Thesis

Machine Learning Methods for Business Continuity Prediction

Supervisors

Prof. GUIDO PERBOLI

Prof. STEFANO MUSSO

Candidate

SAIDALIKHON ALIKHONOV

JUNE 2022

Table of Contents

Chapter 1

Introduction

The goal of this research is to use machine learning to predict the bankruptcy status of small and medium-sized Italian innovative enterprises. It was hoped that by evaluating the financial status of Italian SMEs, the impact of this economic crisis could be predicted. At this stage, the literature was investigated step by step, from risk management to Machine learning techniques to predict default.

Thesis structure

In Chapter 2, The notion of risk, its characteristics, and typical risk categories are deconstructed, and risk response tactics are evaluated based on these. We took a look at the risk management concept. Risk management phases and approaches have been addressed, as well as risk assessment, under this topic.

In Chapter 3, Concept and types of AI are discussed. Machine Learning usage and some efficient ML techniques and algorithms also mentioned in this chapter.

In Chapter 4, Programming Language and required libraries to manage datasets, preprocess data, build machine learning and deep learning models are described.

forecasting methods used in the literature has been realized. In addition, the general logic of the machine learning technique used in the application is explained in a main frame.

In Chapter 5, methods to build efficient models to predict risk of default, both simple machine learning and deep learning approaches are discussed. Performance of models are accessed, results is visualized. After assessment, test and conclusion is proceeded.

Chapter 2

Risk and its Management

Risk has been described by numerous individuals in various ways. One cannot assert that there is an universally accepted definition. Risk can be defined as the possibility that an event will occur or the condition of being influenced by an occurrence, if it is essential to consolidate all of these definitions. When we explore the origin of the word risk and its connotations in other languages, we may use them as a starting point for defining risk.

If risk is defined using the conventional framework, it can be represented as the likelihood of not achieving a desired outcome, loss, or damage within a specified time frame. In addition, risk can be defined as the possibility of an undesired circumstance occurring and the intensity of its resulting impact. Future issues and risks are represented by the term "risk."

To describe risk in general terms, it is the likelihood that a random occurrence will occur. Risk is composed of the three components listed below.

- Cause / Source
- Probability that the event could happen
- Impact, as magnitude of the event

2.1 Risk types

Risks an organization may face can be categorized in a variety of ways. Risks differ based on an organization's activity categories, types, objectives, and target criteria. This classification will be considerably influenced by the organization's structure and sectoral characteristics. Every firm should identify and plan for its inherent risks. Consequently, it cannot be argued that there is a universally acknowledged standard for risk classification. At the same time, it will not be feasible to clearly divide the

various sorts of dangers from one another. However, we can categorize the hazards in two distinct frameworks based on their origin and functional consequence. When we categorize risks based on their origin, we may divide them into two distinct groups: internal and external risks, and four groups based on their functional impact: financial, operational, strategic, and external environmental risks.

2.1.1 Source of Risk

- ***Internal Risk*** that company can control
  - *Commercial:* Cost management, Inability to control the flow of money
  - *Technical:* Shortcomings in planning production management
  - *Human:* Changes in management, Shortage in human resource management

- ***External Risk*** that company cannot control
  - *Natural: Natural disasters, Environmental effects*
  - *Economic: Economic Uncertainties, Market risks, Inflation*
  - *Political: Unforeseen government intervention, Taxation, Political instability*

2.1.2 Types of Financial Risks:

Financial risk is one of the most important categories of risk for any businesses. Financial risk is created by market fluctuations, which can involve a variety of reasons. Based on this, numerous categories of financial risk can be distinguished, including Market Risk, Credit Risk, Liquidity Risk, Operational Risk, and Legal Risk. [1]

Market risk: This risk derives from the fluctuation of financial instrument prices. The two categories of market risk are Directional Risk and Non-Directional Risk. Directional risk is produced by fluctuations in stock prices, interest rates, and other variables. Volatility concerns are an example of Non-Directional risk.

Credit Risk: This risk emerges when an individual fails to fulfill their obligations to their counterparty. The two types of credit risk are Sovereign Risk and Settlement Risk. Typically, complex foreign exchange policies are the source of sovereign risk.

Settlement risk, on the other hand, occurs when one party pays while the other fails to fulfill its responsibilities. [1]

Liquidity Risk*:* This risk stems from the inability to conduct transactions. Asset Liquidity Risk and Funding Liquidity Risk are two classifications of liquidity risk. Asset Liquidity risk occurs when there are insufficient buyers or sellers to fulfill sell orders and buy orders, respectively.

Operational Risk: This sort of risk is caused by operational failures, such as poor management or technology malfunctions. Fraud Risk and Model Risk are two classifications for operational risk. Fraud risk results from a lack of controls, while model risk results from improper model application.

Legal Risk: This sort of financial risk is caused by legal restrictions, such as lawsuits. When a business incurs financial losses as a result of legal proceedings, it faces a legal risk.. [1]

2.2 Risk analysis techniques

Risk analysis has a number of applications and can help firms eliminate potential risks and achieve greater performance. Learning how to conduct these studies correctly might help your firm detect process trends and become more proactive. The ability to recognize hazards and maybe prevent them from becoming threats can save a company money and protect its reputation. This document defines risk analysis, addresses why firms conduct them, and provides a list of risk analysis techniques. qualitative and quantitative types of risk analysis exist. When it comes to project management, both quantitative and qualitative analyses reside in the planning phase, although qualitative analysis follows after quantitative analysis if both are performed. [2]

Bow Tie Analysis : This is one of the most applicable strategies for identifying risk mitigations. Bow-tie Analysis begins by examining a risk event before projecting it in two directions. On the left are listed all of the potential causes of the event, while on

the right are mentioned all of the potential effects. Then, it is possible to identify and implement mitigations (or barriers) to each of the causes and effects separately, successfully mitigating both the probability of risk occurrence and the resulting repercussions, should the risk still occur.

The SWIFT : This acronym stands for "Structured What-If Technique" and is a condensed version of a HAZOP. SWIFT employs a systematic, team-based approach in a workshop setting, in which the team analyzes how changes to an authorized design or plan may impact a project through a series of "What if?" questions. This method is very helpful for assessing the viability of Opportunity Risks.

Decision Tree Analysis: Similar to Event Tree Analysis, but without a fully quantified output, Decision Tree Analysis is frequently used to decide the optimal course of action when the outcomes of potential events or suggested plans are uncertain. This is accomplished by beginning with the initial suggested decision and mapping the many paths and outcomes resulting from the initial decision's occurrences. After establishing all possible paths and outcomes and evaluating their respective probabilities, a course of action can be chosen based on a combination of the most desirable outcomes, connected events, and likelihood of success. [2]

Probability/Consequence Matrix: This is now the usual way for determining the severity of a risk in Qualitative Risk Analysis. Risk Matrices might vary in size, but they all perform fundamentally the same function: Provide a practical method for assessing the overall severity of a risk by multiplying the probability of risk occurrence by the impact of risk occurrence. By comparing risk probability to risk consequence, one can establish not only the total severity of the risk, but also the primary driver of risk severity, be it probability or consequence. Based on the risk's prominent drivers, this information is therefore useful for identifying appropriate risk reduction strategies. [2]

2.3 Risk Management

The risk management process is a management instrument that encompasses all the

mechanisms that may have an impact on risk owners' ability to achieve their goals and objectives. Despite being organized under distinct titles, the risk management process follows essentially the same procedures. Managing risk requires four fundamental steps: identification, analysis, response, and monitoring and control.

Risk identification: It is the process of identifying, categorizing, and updating the risks that impede or make it impossible for the administration to achieve its objectives, based on predetermined procedures. After hazards have been discovered, they are either manually or automatically registered in the system so that anyone with access to the system can examine them.

Risk analysis: The risk analysis process is carried out with 3 successive subtitles.

- *Risk Evaluation:* It is the calculation of the probability and impact of each risk. The probability and effects of the risks are shown in figures and the risk score is calculated. Based on this score, it is aimed to measure the severity of the risk.
- *Prioritization of Risks:* It is the ranking of the risks according to the scores they get as a result of the measurement, starting from the highest score according to thedegree of importance. Three risk categories are used to be low, medium and high-risk level.
- *Risk Registration*: Each identified risk is enumerated, approved and recorded by authorized persons

2.4 Risk responses

In reaction to identified hazards, project managers and teams must take action. Concentrating on the most significant risks can increase the likelihood of project success. During the early stages of a project's development, activities and information may appear chaotic, arriving from numerous directions and sources. Risk management provides a structured and disciplined method for documenting, evaluating, and analyzing the information so that a prioritized list of project risks may be compiled. This ranking can be used to properly direct project risk management

resources. To optimize the benefits of project risk management, integrate risk management activities into the Project Management Plan and project work activities. This involves incorporating risk management activities into the Work Breakdown Structure (WBS). WBS is the Master Deliverables List for WSDOT (MDL). The MDL ensures that project work plans are exhaustive, consistent, and exhaustive. Risk Reaction involves work to create and conduct response activities. Plan for this effort in the project management plan and tasks. WSDOT instruments and direction to aid in this task . [3]

Threats	Opportunities
1. Avoid	1. Exploit
2. Transfer	2. Share
3. Mitigate	3. Enhance
4. Accept	

Avoid: Avoidance actions include modifying the Project Management Plan to eliminate a threat, isolating project objectives from the risk's influence, or modifying the in jeopardy project objective by extending the timetable or reducing the scope. By elucidating needs, accumulating data, enhancing communication, and developing specialized knowledge, early-stage project risks can be mitigated. There are two forms of action: (1) removing the cause of the risk (the risk trigger), and (2) executing the project in a different manner while still pursuing the project's objectives. Not all risks can be avoided or eliminated, and for others, this technique may be too costly or time-consuming, but it should be the first solution evaluated for each risk. [4]

Transfer: Transferring a danger does not erase it; nevertheless, it is now owned and handled by a different entity. Risk transfer can be an effective method for mitigating financial risk exposure. A risk premium, such as insurance, performance bonds, or warranties, is virtually always required when transferring project risk to another

party. Contracts can be used to transfer risks to a third party. (PMBOK) Transferring risk requires identifying a third party ready to assume responsibility for its management and liabilities should the risk materialize. The objective is to guarantee that the risk is owned and managed by the party most capable of effectively addressing it. When considering whether or not to pursue a transfer strategy, the cost-effectiveness of risk transfer must be evaluated. [4]

Mitigate: Risk mitigation entails reducing the likelihood and/or severity of a negative risk occurrence to an acceptable level. Taking preventative measures is frequently more successful than attempting to restore the harm after the danger has happened. Adopting less complicated methods, doing further testing and/or field research, and producing a prototype are examples of mitigation techniques. Targeting the links that determine the degree of an effect, such as incorporating redundancy into a subsystem, can decrease the impact of a failed original component. The most common tactics are mitigation and acceptance, as the number of dangers that may be avoided or transferred is often limited. Preventive reactions are superior than curative answers because they are proactive and, if effective, can result in the avoidance of risk. Preventive responses target the risk's root causes; if it is not possible to lower the risk's likelihood, a mitigation reaction should address the risk's negative effect by addressing the factors that influence its severity. [4]

Accept: The phrase "accept" refers to risks that persist after response activities have been taken and/or for which response is not cost-effective; hazards that are uncontrolled (no reaction measures are feasible) are also accepted. In the end, it is not feasible to eliminate all hazards or seize all opportunities; nevertheless, we can document them and at least raise awareness that they exist and have been discovered; this is referred to as "passive acceptance" by some. In certain instances and sectors, a contingency reserve is developed to address the aggregate residual risk that has been accepted; this is also referred to as "active acceptance." As development of the project progresses, the risk profile will evolve. As we get more adept at mitigating risks and

as our project expertise expands, our risk exposure will decrease. Effectively, we may retire risk reserve when risk occurrences are avoided or managed, or as the risk's active period expires, and it becomes retired.

2.5 Risk monitoring and control

All previous steps are evaluated and documented at this stage. Risks are maintained at a tolerable level by using control measures. On the other hand, certain hazards are uncontrollable and must be checked continually. Examples include environmental and market hazards. At this time, professionals should closely monitor the controls.

2.6 Risk Assessment

Risk assessment is the full process of formulating a plan of action and tactics to be adopted in order to limit risks and the frequency of exposure to possible hazards, as well as determining the significance level of potential hazards. The risk assessment procedure includes the phases of risk identification, risk measurement, calculation of an institution's risk appetite, and determination of risk responses.

Evaluating the risks encountered when attempting to achieve the business purpose, vision, and goals, once the goals and objectives have been clearly established, creates an atmosphere where the most effective reaction can be given to these risks. Risk assessment is the process of identifying and assessing significant risks that might impede the organization's ability to achieve its objectives, as well as deciding the most effective solutions to them. Due to the ever-changing nature of an institution's environment, risk assessment should be an ongoing and recurring activity. Moreover, risk assessment entails modifying internal control in order to recognize and analyze changing circumstances, opportunities, and hazards.

After defining the hazards, a risk assessment is conducted to determine how to effectively manage each risk. Risk assessment is the computation and evaluation of the value of the risks, i.e., their potential impacts and likelihood of occurrence. While the effect indicates the significance of the risk to the organization's capacity to fulfill

its goals, probability refers to the likelihood of the risk occurring within a specified time frame.

After calculating the effect of the risk and its likelihood of occurrence, the risk capacity of the organization is established. At this stage, it is crucial for senior management to guide managers throughout the business regarding acceptable and unacceptable internal risk kinds and levels.

What is meant by acceptable risk is how much danger an organization is willing to tolerate and how much risk it is willing to undertake in order to complete important tasks. Acceptable risk varies based on the organization's operations, its size, and the features of the industry. In other words, the acceptable threshold of risk for each firm should vary. After evaluating the types and degrees of risks based on their potential consequences and occurrence likelihood, the company determines how to manage the risks.

There are several risk assessment methods. Various tools and strategies can be useful in various circumstances and contexts. The purpose of risk assessment is to determine the causes, probabilities, and effects of hazards.

One of these is brainstorming, which is the free-flowing conversation of a group of specialists in order to generate answers to potential threats, mistakes, and risks. Auditing is a further method. It can do both process analysis and audits. By developing a survey approach, it is possible to examine the viewpoints on risk situations and sources.

In addition to this, the SWOT (Strength, Weakness, Opportunity, and Threat) and PESTLE (Political, Economic, Sociological, Technological, Ethical, Legal, and Environmental) approaches might be utilized. The SWOT analysis enables impartial evaluation of the possibilities and risks caused by dangerous conditions inside the firm and the industry. With PESTLE analysis, it is feasible to analyze external environment-related opportunities and risks.

The most used technique is the risk matrix. In this technique, the degree of risk is

determined using probability and effect variables. Typically, the risk probability is displayed on the vertical axis and the risk effect on the horizontal axis. According to the risk score generated by these two events, the risk matrix is colored to indicate the significance of potential risk-related situations. Red indicates the highest danger, orange indicates medium risk, and green indicates the lowest risk.

Chapter 3

Artificial Intelligence and Machine Learning

Artificial intelligence (AI) is the ability of a computer or a computer-controlled robot to accomplish tasks that are normally performed by intelligent beings. The phrase is widely used to refer to a project aimed at creating systems with human-like cognitive abilities, such as the ability to reason, discern meaning, generalize, and learn from past experiences. Since the invention of the digital computer in the 1940s, it has been proven that computers can be programmed to perform extremely complicated jobs with ease, such as finding proofs for mathematical theorems or playing chess. Despite ongoing increases in computer processing speed and memory capacity, no programs have yet to equal human adaptability across broader areas or in activities requiring a great deal of common knowledge. However, certain programs have surpassed the performance levels of human specialists and professionals in executing specific tasks, and artificial intelligence in this limited sense can be found in applications as diverse as medical diagnosis, computer search engines, and voice or handwriting recognition. [5]

While there have been several definitions of artificial intelligence (AI) throughout the previous few decades, John McCarthy provides the following description in this 2004 [6] article. "The science and engineering of creating intelligent devices, particularly clever computer programs, is known as artificial intelligence. It's analogous to utilizing computers to investigate human intellect, but AI isn't limited to physiologically observable ways."

Artificial Intelligence: A Modern Approach [7], written by Stuart Russell and Peter Norvig, has now become one of the most widely used textbooks in AI research. They explore four possible aims or definitions of AI, which distinguishes computer systems based on their rationality and thinking vs. acting:

Human approach:

- Systems that think like humans
- Systems that act like humans

Ideal approach:

- Systems that think rationally
- Systems that act rationally

While there have been several definitions of artificial intelligence (AI) throughout the last few decades, John McCarthy provides the following description in this 2004 [6] article. "It is the science and engineering of creating intelligent machines, particularly computer programs. It's akin to the problem of utilizing computers to comprehend human intellect, but AI doesn't have to limit itself to biologically observable ways."

Artificial Intelligence: A Modern Approach [7], written by Stuart Russell and Peter Norvig, is one of the top textbooks in the field of AI. They look at four possible aims or definitions of AI, which distinguishes computer systems based on their rationality and thinking vs. acting:

3.1 Types of AI

Weak AI, also known as Narrow AI or Artificial Narrow Intelligence (ANI), is AI that has been trained and honed to do particular tasks. Most of the AI that surrounds us today is powered by weak AI. This form of artificial intelligence is anything but feeble; it allows sophisticated applications such as Apple's Siri, Amazon's Alexa, IBM Watson, and driverless cars, among others.

Artificial General Intelligence (AGI) and Artificial Super Intelligence (AIS) comprise strong AI (ASI). Artificial general intelligence (AGI), sometimes known as general artificial intelligence (AI), is a hypothetical kind of artificial intelligence in which a machine possesses human-level intellect, a self-aware awareness, and the ability to solve problems, learn, and plan for the future. Superintelligence, also known as Artificial Super Intelligence (ASI), would transcend the intelligence and capabilities of the human brain. Despite the fact that strong AI is yet totally theoretical and has no practical applications, this does not preclude AI researchers from studying its development. In the meanwhile, the finest instances of ASI may come from science fiction, such as HAL from 2001: A Space Odyssey, a superhuman, rogue computer aide.

Given that deep learning and machine learning are frequently used interchangeably, it is important to note the distinctions between the two. As stated previously, both deep learning and machine learning are subfields of artificial intelligence; nonetheless, deep learning is a subfield of machine learning.

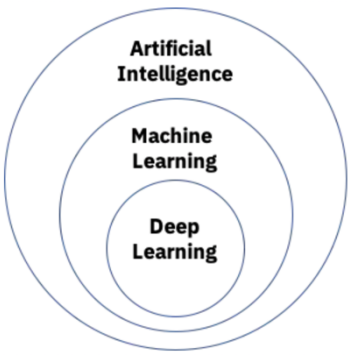

Figure 3.1

3.2 Machine Learning

Using machine learning techniques, AI was able to advance beyond merely fulfilling the tasks for which it was created. Prior to the general use of machine learning, AI algorithms were exclusively employed to automate low-level commercial and enterprise processes.

This covered duties such as intelligent automation and basic categorization based on rules. This meant that AI algorithms were only applicable to the domain for which they were designed. However, with the advent of machine learning, computers were able to transcend their programming and evolve with each repetition.

Machine learning is fundamentally distinct from artificial intelligence since it can evolve. Using a variety of programming approaches, machine learning algorithms are able to analyze and extract meaningful information from vast volumes of data. This enables them to improve upon their prior iterations by learning from the offered data. We cannot discuss machine learning without discussing big data, as it is one of the most crucial components of machine learning algorithms. As a subject that largely employs statistical approaches, the success of any sort of AI is typically contingent on the quality of its dataset.

A steady influx of diverse, well-organized data is essential for an effective machine learning solution. In today's online-centric society, businesses have access to vast amounts of client data, typically in the millions. This data, which is massive in terms of both the number of data points and the number of fields, is referred to as "big data" since it contains a vast quantity of information.

Big data is time-consuming and challenging to analyze by human standards, yet high-quality data is the most effective training material for a machine learning system. The greater the quantity of clean, useable, and machine-readable data in a large dataset, the more successful the training of the machine learning algorithm. As described, machine learning algorithms are capable of self-improvement through training. ML algorithms are being trained utilizing three common techniques. Three forms of machine learning exist: supervised learning, unsupervised learning, and reinforcement learning.

3.2.1 Types of machine learning

There are several approaches to train machine learning algorithms, each with their own pros and downsides. To comprehend the advantages and disadvantages of each

sort of machine learning, we must first examine the types of data they consume. There are two types of data in machine learning: labeled data and unlabeled data.

Labeled data include input and output parameters that are totally machine-readable, but labeling the data involves a significant amount of human effort. One or none of the parameters are in a machine-readable format for unlabeled data. This eliminates the requirement for human labor but necessitates more complicated solutions.

There are more sorts of machine learning algorithms that are employed in very specialized use cases, however the three most prevalent techniques are employed today.

Supervised Learning

Supervised learning is one of the most fundamental machine learning techniques. This method involves training the machine learning algorithm using labeled data. Despite the fact that the data must be appropriately labeled for this approach to operate, supervised learning is incredibly effective under the right conditions.

In supervised learning, a small training dataset is provided to the machine learning algorithm. This training dataset is a subset of the larger dataset and is used to provide the algorithm with a fundamental understanding of the issue, solution, and data points to be processed. The training dataset has many features with the final dataset and gives the algorithm with the labeled problem-specific parameters.

The program then identifies correlations between the specified parameters, thereby generating a cause-and-effect link between the dataset's variables. At the conclusion of training, the algorithm understands how the data operates and the input-output connection.

This solution is subsequently deployed for usage with the final dataset, from which it learns in the same manner as it did with the training dataset. This implies that supervised machine learning algorithms will continue to develop, uncovering new patterns and correlations as they train themselves on fresh data.

Unsupervised Learning

One of the most fundamental forms of machine learning is supervised learning. The machine learning algorithm is taught with labeled data in this type. Despite the fact that the data must be appropriately labeled for this method to be effective, supervised learning is a very effective technique when applied in the proper context.

In supervised learning, the machine learning algorithm is provided with a small training dataset to analyze. This training dataset is a subset of the larger dataset and serves to familiarize the algorithm with the issue, solution, and data points to be handled. In addition to sharing many similarities with the final dataset, the training dataset supplies the algorithm with the labeled parameters necessary to solve the problem.

In essence, the algorithm establishes a cause-and-effect link between the variables in the dataset. At the conclusion of training, the algorithm understands how the data functions and the link between input and output.

This solution is subsequently deployed for usage with the final dataset, which it learns from in the same way as the training dataset. This implies that supervised machine learning algorithms will continue to develop, identifying new patterns and associations as they train themselves on fresh data.

Reinforcement Learning

Reinforcement learning is directly influenced by how humans learn from data in their daily lives. It employs a trial-and-error mechanism for self-improvement and learning from new circumstances. Positive outcomes are encouraged or 'reinforced,' while negative outcomes are discouraged or 'punished.

Reinforcement learning operates based on the psychological idea of conditioning by placing the algorithm in a work environment with an interpreter and a reward system. In each iteration of the algorithm, the interpreter receives the output result and determines if the outcome is favorable.

In the event that the program finds the proper answer, the interpreter encourages it by rewarding the algorithm. If the results is unfavorable, the algorithm must iterate

until a better result is found. Typically, the incentive system is directly proportional to the efficacy of the outcome.

In most applications of reinforcement learning, such as determining the shortest path between two locations on a map, the result is not an absolute value. Instead, it is assigned an effectiveness score stated as a percentage. The greater this percentage value, the greater the algorithm's payout. Therefore, the software is taught to provide the optimal answer for the optimal reward.

3.3 Machine Learning techniques and algorithms

Internet of Things (IoT) data, cybersecurity data, mobile data, business data, social media data, health data, etc., are abundant in the digital world in the current period of the Fourth Industrial Revolution (4IR or Industry 4.0). Knowledge of artificial intelligence (AI), specifically machine learning (ML), is essential for intelligently analyzing these data and developing the associated smart and automated applications. There are several types of machine learning algorithms in the field, including supervised, unsupervised, semi-supervised, and reinforcement learning. Moreover, deep learning, which is part of a larger family of machine learning techniques, can effectively evaluate vast amounts of data. In this article, we provide a detailed overview of various machine learning methods that may be used to increase an application's intelligence and capabilities. The primary contribution of this work is thus the explanation of the concepts of several machine learning techniques and their use in numerous real-world application fields, including cybersecurity systems, smart cities, healthcare, e-commerce, agriculture, and many more. Based on our analysis, we also emphasize the limitations and prospective future research topics. Overall, this study intends to serve as a technical point of reference for academic and industry specialists as well as decision-makers in a variety of real-world circumstances and application domains.

In general, the efficacy and efficiency of a machine learning solution are determined by the type and qualities of the data as well as the performance of the learning algorithms. In the field of machine learning algorithms, classification analysis, regression, data clustering, feature engineering and dimensionality reduction, association rule learning, or reinforcement learning techniques exist to effectively build data-driven systems. [9]In addition, deep learning originated from the artificial neural network that can be used to intelligently analyze data, which is a subset of machine learning approaches [10]. Thus, finding a good learning algorithm for the intended application in a certain area is tricky. The reason for this is that the objectives of various learning algorithms vary, and even the outcomes of different learning algorithms within the same category might vary based on the qualities of the data [11]. Consequently, it is essential to comprehend the principles of various machine learning algorithms and their applicability for use in a variety of real-world application domains, such as IoT systems, cybersecurity services, business and recommendation systems, smart cities, healthcare and COVID-19, context-aware systems, and sustainable agriculture, among others, which are briefly described in Section "Applications of Machine Learning." [12]

Typically, machine learning algorithms ingest and analyze data to discover the associated patterns about persons, corporate processes, transactions, events, etc. Following are discussions of several sorts of real-world data and machine learning algorithm categories.

3.3.1 Types of real-world data

The availability of data is typically seen as the linchpin for building machine learning models or data-driven real-world systems. [13] There are a variety of data types, including structured, semi-structured, and unstructured [14]. Moreover, "metadata" is an additional data type that often represents data about data. Following is a quick discussion of several forms of data.

- *Structured:* It has a well-defined structure, corresponds to a data model with a consistent order, is well structured and simple to obtain, and is utilized by an

entity or computer program. Typically, structured data are kept in a tabular manner in well-defined systems such as relational databases. Examples of structured data include names, dates, addresses, credit card numbers, stock information, geolocation, etc.

- *Unstructured:* It has a well-defined structure, corresponds to a data model with a consistent order, is well structured and simple to obtain, and is utilized by an entity or a computer program. In well-defined systems, such as relational databases, structured data are often kept in a tabular fashion, i.e., in a table. Structured data includes, among other things, names, dates, addresses, credit card numbers, stock information, geolocation, etc.
- *Semi-structured:* Semi-structured data are not stored in a relational database like the structured data mentioned above, but it does have certain organizational properties that make it easier to analyze. HTML, XML, JSON documents, NoSQL databases, etc., are some examples of semi-structured data.
- *Metadata:* It is not typical data, but rather "data about data." The major distinction between "data" and "metadata" is that "data" simply refers to the information that may be used to categorize, measure, or describe something in relation to an organization's data attributes. Metadata, on the other hand, defines the pertinent data information, hence enhancing its importance for data users. Simple examples of document metadata include the document's author, file size, date of creation, keywords used to describe the content, etc.

3.3.2 Types of ML algorithms

It is not a standard type of data, but rather "data about data." The major distinction between "data" and "metadata" is that "data" simply refers to the stuff that may be used to categorize, measure, or describe something in relation to an organization's data attributes. Metadata, on the other hand, defines the pertinent data information, hence enhancing its value for data users. Metadata may include the document's author, file size, creation date, keywords, etc.

Classification

It is not a typical data format, but rather "data about data." The major distinction between "data" and "metadata" is that "data" refers to the information that may be used to categorize, measure, or even describe anything in relation to an organization's data attributes. Metadata, on the other hand, defines the pertinent data information, hence increasing its value for data users. The metadata of a document might include the document's author, file size, creation date, keywords, etc.

- *Binary classification:* It refers to classification jobs using two class labels, such as "true" and "false" or "yes" and "no." In such binary classification tasks, one class may represent the normal condition, whilst another class may represent the aberrant state. For example, "cancer not found" is the normal condition of a job involving a medical test, but "cancer identified" may be regarded the abnormal state. "spam" and "not spam" in the preceding example of email service providers are also considered binary classifications.
- Classification jobs with more than two class labels are referred to as multiclass classification. In contrast to binary classification problems, the multiclass classification lacks the idea of normal and abnormal results. Instead, instances are categorised as belonging to one class within a range of defined classes. Classifying various types of network attacks in the NSL-KDD [15] dataset, where the attack categories are categorized into four class labels, such as DoS (Denial of Service Attack), U2R (User to Root Attack), R2L (Root to Local Attack), and Probing Attack, is an example of a multiclass classification task.
- Multi-label categorization is an essential aspect of machine learning when one example is linked with many classes or labels. Consequently, it is an extension of multiclass classification in which the classes involved in the issue are hierarchically constructed and each example may simultaneously belong to more than one class at each hierarchical level, e.g., multi-level text classification. For instance, Google news may be categorized as "city name," "technology," "latest news," etc. Multi-label classification employs sophisticated machine learning techniques that enable the prediction of many

mutually non-exclusive classes or labels, in contrast to classical classification problems in which class labels are mutually exclusive [16].

- *Multiclass classification:* This term often relates to classification jobs with more than two class labels. Unlike binary classification problems, multiclass classification does not adhere to the idea of normal and abnormal results. In lieu of it, instances are categorized as belonging to one of a number of specified classes. Classifying various types of network attacks in the NSL-KDD [15] dataset, where the attack categories are classified into four class labels, such as DoS (Denial of Service Attack), U2R (User to Root Attack), R2L (Root to Local Attack), and Probing Attack, can be an example of a multiclass classification task.
- *Multi-label classification:* Traditionally, this relates to classification jobs with several class labels. In contrast to binary classification problems, the multiclass classification lacks the concept of normal and abnormal results. Instead, instances are categorized as belonging to one of a number of predetermined groups. Classifying various types of network attacks in the NSL-KDD [15] dataset, where the attack categories are categorized into four class labels, such as DoS (Denial of Service Attack), U2R (User to Root Attack), R2L (Root to Local Attack), and Probing Attack, can be an example of a multiclass classification task.

In machine learning and data science literature, several classification techniques have been suggested. Following is a summary of the most popular and widespread approaches used in a variety of application areas.

- *Naive Bayes (NB):* The naive Bayes method is based on Bayes' theorem with the premise that each pair of characteristics are independent [17]. In a variety of real-world scenarios, such as document or text categorization, spam filtering, etc., it performs effectively and may be used to both binary and multi-class categories. The NB classifier may be used to successfully categorize noisy

examples in the data and build a strong prediction model. The primary advantage is that, compared to more complicated algorithms, it requires a minimal quantity of training data to rapidly estimate the required parameters. However, its performance may be affected by its strong assumptions on the independence of characteristics. The common variations of the NB classifier include Gaussian, Multinomial, Complement, Bernoulli, and Categorical.

- *Linear Discriminant Analysis (LDA):* LDA is a linear decision boundary classifier built by fitting class conditional densities to data and applying Bayes'. This technique, also known as a generalization of Fisher's linear discriminant, projects a given dataset into a lower-dimensional space, i.e., a reduction of dimensionality that minimizes the complexity of the model or decreases the processing costs of the resultant model. The basic LDA model typically assigns a Gaussian density to each class, assuming that all classes have the same covariance matrix. LDA is closely connected to ANOVA (analysis of variance) and regression analysis, both of which strive to represent one dependent variable as a linear combination of other characteristics or measures.
- *Logistic regression (LR):* Linear Discriminant Analysis (LDA) is a linear decision boundary classifier built by fitting data with class conditional densities and using Bayes'. This approach is also known as an extension of Fisher's linear discriminant because it projects a given dataset into a lower-dimensional space, i.e., a reduction of dimensionality that minimizes the model's complexity or decreases the resultant model's processing costs. Assuming that all classes have the same covariance matrix, the typical LDA model normally assigns a Gaussian density to each class. LDA is closely connected to ANOVA (analysis of variance) and regression analysis, which strive to represent a dependent variable as a linear combination of other characteristics or measures.

$$g(z)=\frac{1}{1+e^{-z}} \qquad \text{(Eq 4.1)}$$

- *K-nearest neighbors (KNN):* K-Nearest Neighbors (KNN) [19] is a "non-

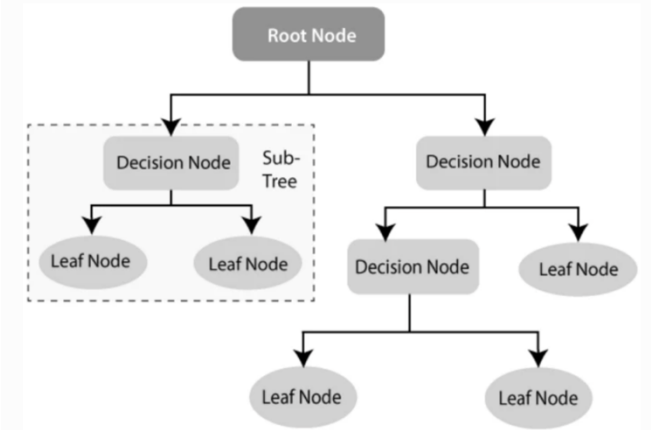

Figure 4.1

generalizing learning" or "instance-based learning" method, sometimes known as "lazy learning." It retains all instances matching to training data in n-dimensional space, as opposed to focusing on developing a generic internal model. KNN classifies new data points using on similarity metrics (such as the Euclidean distance function). Classification is determined by the simple majority vote of each point's k closest neighbors. It is fairly resilient to noisy training data, and its accuracy is dependent on the quality of the training data. The most difficult aspect of KNN is determining the appropriate number of neighbors to be evaluated. KNN can be utilized for both classification and regression.

- *Support vector machine (SVM):* K-Nearest Neighbors (KNN) [19] is a method for "instance-based learning" or non-generalizing learning, commonly referred to as "lazy learning." Instead than emphasizing the construction of a generic internal model, it maintains all instances matching to training data in n-dimensional space. KNN utilizes data and classifies new data points based on similarity measurements (e.g., Euclidean distance function). Classification is determined by a simple majority vote of the k closest neighbors of each location. It is highly tolerant of noisy training data, and its accuracy depends on the data quality. Choosing the appropriate number of neighbors to consider

is the most difficult aspect of KNN. KNN may be used for classification as well as regression.

- *Decision tree (DT):* The decision tree (DT) is a well-known non-parametric approach for supervised learning. For both classification and regression problems, DT learning approaches are employed. ID3, C4.5, and CART are well-known DT algorithms [21]. In addition, the recently suggested BehavDT [22] and IntrudTree by Sarker et al. are effective in their respective application domains, namely user behavior analytics and cybersecurity analytics. DT classifies instances by sorting the tree from its root to its leaf nodes, as shown in Figure 4.1. Instances are categorised by examining the attribute specified by each node in the tree, beginning with the root node and proceeding along the branch corresponding to the attribute value. The most common criterion for splitting are "Gini" for the Gini impurity and "entropy" for the mathematically expressible information gain.

  Entropy: $H(x) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$ (Eq. 4.2)

  $\text{Gini}(E) = 1 - \sum_{1}^{c} p_i^2$ (Eq. 4.3)

- *Random forest (RF):* A random forest classifier is a well-known ensemble classification approach utilized in several application fields of machine learning and data science. This approach employs "parallel ensembling," which fits several decision tree classifiers in parallel on distinct subsamples of a data set and use majority voting or averages to determine the conclusion or final result. Thus, it reduces the problem of overfitting and improves forecast accuracy and control. Consequently, the RF learning model with several decision trees is often more precise than a model with a single decision tree. It combines bootstrap aggregation (bagging) with random feature selection to produce a succession of decision trees with controlled variance. It fits well for both categorical and continuous data and is flexible to classification and regression issues.

- *Adaptive Boosting (AdaBoost):* Adaptive Boosting (AdaBoost) is an ensemble learning procedure that uses an iterative strategy to improve ineffective classifiers by learning from their mistakes. This concept was created by Yoav Freund and others and is known as "meta-learning." In contrast to random forest, which use parallel ensembling, Adaboost employs sequential ensembling. It produces a strong classifier by integrating a large number of underperforming classifiers to obtain a high-accuracy classifier. In this respect, AdaBoost is referred to be an adaptive classifier since it considerably improves the efficiency of the classifier, although in other cases it might result in overfitting. However, AdaBoost is susceptible to noisy data and outliers.
- *Extreme gradient boosting (XGBoost):* Similar to Random Forests, Gradient Boosting is an ensemble learning technique that creates a final model from a collection of individual models, often decision trees. Similar to how neural networks improve weights via gradient descent, the gradient is utilized to minimize the loss function. Extreme Gradient Boosting (XGBoost) is a kind of gradient boosting that takes into consideration more precise approximations while choosing the optimal model. It computes second-order gradients of the loss function to minimize loss and enhanced regularization (L1 and L2) [16], so reducing overfitting and enhancing model generalization and performance. XGBoost is quick to comprehend and can effectively manage big datasets.
- *Stochastic gradient descent (SGD):* Stochastic gradient descent (SGD) is an iterative method for maximizing a smooth objective function, where stochastic refers to random chance. This minimizes the computational load, especially in high-dimensional optimization problems, allowing for quicker iterations at the expense of a slower convergence rate. A gradient is the slope of a function that estimates the degree of change of one variable in response to changes in another variable. Gradient Descent is a convex function whose output is the partial derivative of its input parameters. If denotes the learning rate and J_j represents the cost of the i th training example, then Eq. (4) represents the stochastic

gradient descent weight update procedure at the j th iteration. SGD has been effectively applied to difficulties frequently encountered in text categorization and natural language processing in large-scale and sparse machine learning. However, SGD is sensitive to feature scaling and requires a variety of hyperparameters, such as the number of iterations and the regularization parameter.

$$w_j \coloneqq w_j - \alpha \frac{\partial J_j}{\partial w_j} \qquad \text{(Eq. 4.4)}$$

- *Rule-based classification*: Stochastic gradient descent (SGD) is an iterative method for maximizing an objective function with the necessary smoothness criteria, where stochastic refers to random chance. This minimizes the computational load, especially in high-dimensional optimization problems, allowing for quicker iterations at the expense of a reduced convergence rate. A gradient is the slope of a function that determines a variable's degree of change in response to the changes of another variable. Mathematically, the Gradient Descent is a convex function whose output is a partial derivative of the set of input parameters. If is the rate of learning and Ji is the cost of the i th training example, then Eq. (4) is the stochastic gradient descent weight update procedure at the j th iteration. SGD has been effectively used to text classification and natural language processing challenges in large-scale and sparse machine learning. Nevertheless, SGD is sensitive to feature scaling and requires a variety of hyperparameters, including the regularization parameter and the number of iterations.

Regression

Regression analysis comprises a number of machine learning techniques for predicting the value of a continuous outcome variable (y) based on the value of one or more predictor variables (x). Classification predicts separate class labels, but regression predicts a continuous quantity. There are many similarities between the

two types of machine learning algorithms. Regression models are currently widely utilized in several sectors, such as financial forecasting or prediction, cost estimate, trend analysis, marketing, time series estimation, and medication response modeling, to name a few. Following is a quick explanation of linear, polynomial, lasso, and ridge regression, among other well-known forms of regression algorithms.

- *Simple and multiple linear regression:* This is one of the most often used ML modeling strategies and a well-known regression method. The dependent variable is continuous, the independent variables might be continuous or discrete, and the regression line is linear. Using the best-fit straight line, linear regression establishes a link between the dependent variable (Y) and one or more independent variables (X) (also known as the regression line). It is defined by the equations shown below:

  $y=a+bx+e$ (Eq. 4.5)

  $y = a + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$+e (Eq. 4.6)

  This is a well-known regression approach and one of the most used ML modeling strategies. In this method, the dependent variable is continuous, the independent variables may be continuous or discrete, and the regression line has a linear shape. Linear regression establishes the link between the dependent variable (Y) and one or more independent variables (X) using the best-fit straight line (also known as the regression line). The following equations describe its definition.
- *Polynomial regression:* This is a common ML modeling tool and a well-known regression method. In this method, the dependent variable is continuous, the independent variable(s) may be continuous or discrete, and the regression line has a linear shape. Linear regression establishes a link between the dependent variable (Y) and one or more independent variables (X) using the best-fit

straight line (also known as the regression line). The following equations define it:

$$y = a + b_1 x + b_2 x^2 + b_3 x^3 + \cdots + b_n x^n + e \quad \text{(Eq. 4.7)}$$

Here, y represents the predicted/target output, b_0,b_1,..., b_n represent the regression coefficients, while x is an independent/input variable. If data are not distributed linearly but are instead polynomial to the nth degree, then polynomial regression is used to obtain the required result.

- *LASSO and ridge regression:* Due to their capacity to prevent over-fitting and reduce the model's complexity, LASSO and Ridge regression are well-known as potent approaches that are frequently employed when developing learning models in the presence of a high number of features. The LASSO (least absolute shrinkage and selection operator) regression model employs the shrinkage-based L1 regularization approach, which penalizes "absolute value of coefficient magnitude" (L1 penalty). LASSO appears to convert coefficients to absolute zero as a consequence. Thus, the objective of LASSO regression is to identify the subset of predictors that minimizes the error in prediction for a quantitative response variable. Alternatively, ridge regression employs L2 regularization, which is the "squared magnitude of coefficients" (L2 penalty). Thus, ridge regression requires the weights to be tiny, but never sets the coefficient value to zero, and produces a solution that is not sparse. Overall, LASSO regression is effective for obtaining a subset of predictors by removing less significant characteristics, but ridge regression is useful when a data set contains "multicollinearity," which refers to predictors that are connected with other predictors. [16]

Clustering

Due to their capacity to prevent over-fitting and reduce the model's complexity, LASSO and Ridge regression are well-known as effective strategies for developing learning models when a high number of features are available. The LASSO (least

absolute shrinkage and selection operator) regression model employs the L1 regularization approach that penalizes "absolute value of magnitude of coefficients" through the use of shrinkage (L1 penalty). Consequently, LASSO seems to convert coefficients to absolute zero. Consequently, the objective of LASSO regression is to identify the subset of predictors that minimizes the prediction error for a quantitative response variable. In contrast, ridge regression makes advantage of L2 regularization, which is the "squared magnitude of coefficients" (L2 penalty). Consequently, ridge regression requires the weights to be tiny, but never sets the coefficient value to zero, and generates a non-sparse solution. In general, LASSO regression is good for obtaining a subset of predictors by removing less significant characteristics, but ridge regression is effective when a data set contains "multicollinearity," which refers to predictors that are connected with other predictors. [16]

Dimensionality reduction

Due to their capacity to prevent over-fitting and reduce the model's complexity, LASSO and Ridge regression are widely recognized as effective strategies for developing learning models in the presence of numerous features. The LASSO (least absolute shrinkage and selection operator) regression model employs the shrinkage-based L1 regularization approach, which penalizes the "absolute value of coefficient magnitude" (L1 penalty). Thus, LASSO seems to convert coefficients to absolute zero. Consequently, the objective of LASSO regression is to determine the subset of predictors that minimizes the prediction error for a quantitative response variable. Alternatively, ridge regression utilizes L2 regularization, which is the "squared magnitude of coefficients" (L2 penalty). Thus, ridge regression causes the weights to be tiny, but never sets the coefficient value to zero, and produces a solution that is not sparse. Overall, LASSO regression is effective for obtaining a subset of predictors by removing less significant characteristics, whereas ridge regression is useful when a data set contains "multicollinearity," which refers to predictors that are connected with other predictors. [16]

Reinforcement Learning

Reinforcement learning (RL) is a machine learning approach that enables an agent to learn via trial and error in an interactive environment by utilizing feedback from its actions and experiences. In contrast to supervised learning, which is based on provided sample data or examples, the RL technique relies on interaction with the environment. The issue to be addressed in reinforcement learning (RL) is characterized as a Markov Decision Process (MDP) [25], i.e., it involves making judgments consecutively. Typically, an RL problem consists of four components: Agent, Environment, Rewards, and Policy.

RL may be loosely divided into Model-based and Model-free methods. Model-based RL is the process of deducing optimum behavior from an environment model by conducting actions and watching the effects, which include the next state and the immediate reward. AlphaZero and AlphaGo [26] are two model-based methods. A model-free method, on the other hand, does not utilize the related transition probability distribution and reward function. Model-free algorithms include Q-learning, Deep Q Network, Monte Carlo Control, SARSA (State–Action–Reward–State–Action), etc. The policy network, which is necessary for model-based RL but not model-free, is the primary distinction between model-free learning and model-based learning. Following is a discussion of the prevalent RL algorithms.

- *Monte Carlo methods:* Monte Carlo procedures, also known as Monte Carlo experiments, are a broad class of computing algorithms that rely on repeated random sampling to get numerical results. The fundamental idea is to employ randomness to solve problems that are, in principle, deterministic. Optimization, numerical integration, and drawing from probability distributions are the three problem groups in which Monte Carlo methods are most frequently employed.
- *Q-learning:* Monte Carlo procedures, also known as Monte Carlo experiments, are a large class of computing algorithms that rely on repeated random

sampling to get numerical results. Using randomness to solve issues that are, in principle, deterministic is the basic notion. Optimization, numerical integration, and drawing from probability distributions are the three problem groups for which Monte Carlo methods are most frequently employed.

- *Deep Q-learning:* Deep Q-Learning entails feeding the starting state to a neural network, which then returns the Q-value of all potential actions as an output. Nonetheless, Q-learning is effective when faced with a somewhat basic environment. However, when the number of states and actions increases, deep learning may be utilized to approximate functions.

Along with supervised and unsupervised learning, reinforcement learning is one of the fundamental machine learning paradigms. RL can be used to solve numerous real-world problems in a variety of disciplines, including game theory, control theory, operations analysis, information theory, simulation-based optimization, manufacturing, supply chain logistics, multi-agent systems, swarm intelligence, aircraft control, and robot motion control, among others.

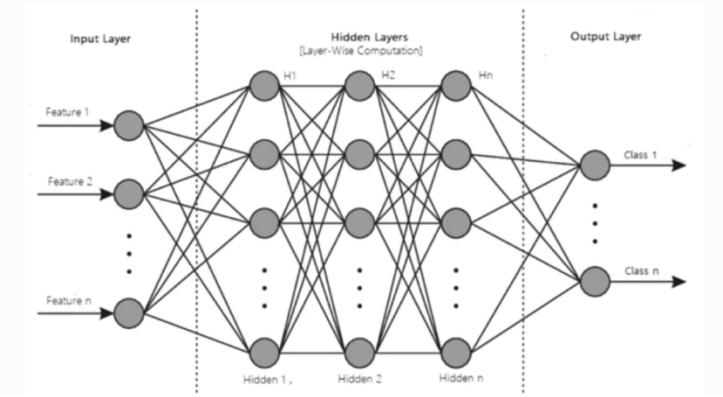

Figure 4.2

4.4.2.6 Artificial Neural Networks and Deep Learning

Reinforcement learning is one of the fundamental machine learning paradigms, alongside supervised and unsupervised learning. RL can be used to solve numerous real-world problems in numerous disciplines, including game theory, control theory, operations analysis, information theory, simulation-based optimization,

manufacturing, supply chain logistics, multi-agent systems, swarm intelligence, aircraft control, and robot motion control, among others.

Multi-layer Perceptron (MLP), Convolutional Neural Network (CNN, or ConvNet), and Long Short-Term Memory Recurrent Neural Network (LSTM-RNN) are the most used deep learning algorithms [27]. Following are many sorts of deep learning approaches that may be utilized to create successful data-driven models for a variety of applications.

- *MLP:* The base architecture of deep learning, which is also known as the feed-forward artificial neural network, is called a multilayer perceptron (MLP) A typical MLP is a fully connected network consisting of an input layer, one or more hidden layers, and an output layer, as shown in *Fig. 4.2*. Each node in one layer connects to each node in the following layer at a certain weight. MLP utilizes the "Backpropagation" technique [6], the most "fundamental building block" in a neural network, to adjust the weight values internally while building the model. MLP is sensitive to scaling features and allows a variety of hyperparameters to be tuned, such as the number of hidden layers, neurons, and iterations, which can result in a computationally costly model.
- *CNN or ConvNet:* The most prevalent deep learning algorithms include Multi-layer Perceptron (MLP), Convolutional Neural Network (CNN, or ConvNet), and Long Short-Term Memory Recurrent Neural Network (LSTM-RNN) [27]. Following is a discussion of the numerous sorts of deep learning approaches that may be used to construct successful data-driven models for diverse applications.
- *LSTM-RNN:* Multi-layer Perceptron (MLP), Convolutional Neural Network (CNN, or ConvNet), and Long Short-Term Memory Recurrent Neural Network (LSTM-RNN) are the three most used deep learning algorithms [27]. Following is a discussion of the numerous sorts of deep learning approaches that may be used to construct efficient data-driven models for a variety of applications.

Chapter 4

Language and Technologies used

Multi-layer Perceptron (MLP), Convolutional Neural Network (CNN, or ConvNet), and Long Short-Term Memory Recurrent Neural Network (LSTM-RNN) are the most used deep learning algorithms [27]. Following are many sorts of deep learning approaches that may be utilized to create successful data-driven models for a variety of applications.

4.1 Python

Applications for machine learning and scientific computing frequently employ linear algebra operations on multidimensional arrays, which are computational data structures for encoding vectors, matrices, and tensors of a higher order. Since these processes are frequently parallelizable over several processor cores, libraries such as NumPy [28] and SciPy use C/C++, Fortran, and third-party BLAS implementations to circumvent threading and other Python restrictions. NumPy is a multidimensional array library providing fundamental linear algebra routines, and the SciPy library adorns NumPy arrays with several useful primitives, including numerical optimizers, signal processing, statistics, and sparse linear algebra. SciPy is utilized in about half of all machine learning projects on GitHub as of 2019 [29].

Pandas enhances NumPy by offering a data frame-like object that supports heterogeneous column types and row and column information. Both NumPy and Pandas give abstractions over a collection of data points with operations that operate on the dataset as a whole. In recent years, the Pandas module has become the standard way to represent tabular data in Python for extract, transform, and load (ETL) and

data analysis. Twelve years and twenty-five versions after its initial release in 2008, the first version 1.0 of Pandas was published in 2020. In the open-source community, where the majority of projects adhere to 4 of 48 semantic versioning standards, a 1.0 release signifies that a library has achieved a significant level of maturity and a stable API. Similar to Pandas, even though the original version of NumPy was released more than 25 years ago (under the name "Numeric"), it is still being developed and maintained. The NumPy development team won a $645,000 grant from the Moore Foundation in 2017 to aid in the library's continued development and maintenance [30]. Pandas, NumPy, and SciPy remain the most user-friendly and recommended options for many data science and computing tasks as of this writing.

4.2 Sckit-learn

Scikit-learn [19] has become the industry standard Python package for feature engineering and classical ML modeling on small to medium-sized datasets6 mostly due to its simple, consistent, and user-friendly API. In addition, with assistance from the open source community, the Scikit-learn programming team prioritizes code quality and thorough documentation. Having pioneered the basic "fit()/predict()" API paradigm, their approach has acted as a model and inspiration for several libraries, as it provides a recognizable face and reduces code modifications when users explore alternative modeling possibilities. Scikit-learn contains a first-class API for unifying the construction and execution of machine learning pipelines, the pipeline API, in addition to its various classes for data processing and modeling, known as estimators (*Figure 4.1*). It supports the end-to-end execution of a collection of estimators, including data processing, feature engineering, and modeling estimators. In addition, Scikit learn offers an API for testing trained models using standard approaches such as cross validation.

The Scikit-learn development team only considers well-established algorithms for inclusion in the library in order to strike the optimal balance between delivering useful features and maintaining high-quality code. However, the surge in machine learning and artificial intelligence research over the past decade has produced a large number of algorithms that are best left as extensions and not integrated into the core. Newer and frequently lesser-known algorithms are donated as Scikit-learn compatible libraries or so-called "Scikit-contrib" packages, the latter of which are maintained by the Scikit-learn community under a common GitHub organization named "Scikit-learn-contrib." When these independent packages adhere to the Scikit-learn API, they may take use of the Scikit-learn ecosystem, allowing users to inherit some of Scikit-advanced learn's capabilities, such as pipelining and cross-validation, for free. [7]

a

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.svm import SVC
from sklearn.pipeline import make_pipeline
from sklearn import datasets
from sklearn.model_selection import train_test_split

iris = datasets.load_iris()
X, y = iris.data, iris.target
X_train, X_test, y_train, y_test =\
    train_test_split(X, y, test_size=0.3,
                     random_state=42, stratify=y)

pipe = make_pipeline(StandardScaler(),
                     PCA(n_components=2),
                     SVC(kernel='linear'))

pipe.fit(X_train, y_train)
y_pred = pipe.predict(X_test)
print('Test Accuracy: %.3f' % pipe.score(X_test, y_test))
```

b

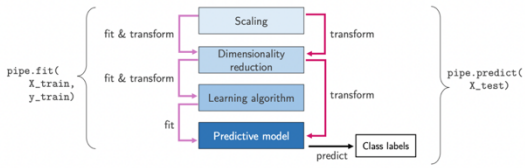

Figure 4.1

4.3 Pytorch

Facebook created and maintains the open-source Python library for deep learning known as PyTorch. The project was initiated in 2016 and grew rapidly in popularity among developers and scholars.

Torch (Torch7) is a C-based open-source software for deep learning that is often accessed through the Lua interface. It was a project that predated PyTorch and is no longer being actively maintained. PyTorch incorporates "Torch" in its name, recognizing the previous torch library, with the prefix "Py" reflecting the Python-centric nature of the new project.

Academics and researchers appreciate the PyTorch API because to its simplicity and adaptability in the building of new deep learning models and applications. Extensive use has led to the development of several extensions for specialized applications (such as text, computer vision, and audio data), as well as pre-trained models that may be employed immediately. As such, it may be the most popular academic library.

PyTorch's versatility comes at the expense of its usability, especially for newcomers, when contrasted to interfaces like as Keras. The decision to utilize PyTorch instead of Keras sacrifices some usability, a somewhat steeper learning curve, and additional code in exchange for greater flexibility and maybe a more active academic community.

A model in PyTorch has a life-cycle, and this very basic concept is essential for both modeling a dataset and comprehending the PyTorch API. The five life-cycle stages are as follows:

1. Prepare the Data.
2. Define the Model.
3. Train the Model.
4. Evaluate the Model.
5. Make Predictions.

4.4 Pandas

Facebook created and maintains the open-source Python package PyTorch for deep learning. The project began in 2016 and became a popular framework among developers and scholars almost immediately.

Torch (Torch7) is an open-source project for deep learning written in C and commonly accessed via the Lua interface. It served as a forerunner to PyTorch and is no longer being developed. The inclusion of "Torch" in the name of PyTorch acknowledges the preceding torch library, but the "Py" prefix indicates the Python-centric nature of the new project.

PyTorch is used by academics and researchers for the creation of novel deep learning models and applications due to its easy and adaptable application programming interface. Extensive use has resulted in several expansions for specialized applications

(such as text, computer vision, and audio data), and pre-trained models that may be utilized immediately may be available. As a result, it is maybe the most popular academic library.

Compared to simpler interfaces like Keras, PyTorch's functionality comes at the expense of its usability, particularly for novices. The decision to utilize PyTorch as opposed to Keras sacrifices some usability, a slightly steeper learning curve, and additional code in exchange for greater flexibility and maybe a more active academic community.

A model in PyTorch has a life-cycle, and this fundamental fact is the foundation for modeling a dataset and comprehending the PyTorch API. The life-cycle comprises the following five stages:

4.5 Matplotlib

Python contains a multitude of libraries. Matplotlib is one of the most popular and widely used libraries for data visualization in Python, providing a variety of capabilities.

It is one of Python's most potent charting packages. It is a cross-platform package that provides several tools for generating 2D plots from lists or arrays of data in Python.

John D. Hunter designed and programmed it in the Python programming language in 2003. The latest stable version of matplotlib was published on May 8, 2021, and is version 3.4.2. NumPy, a package that offers the numerical mathematics extension for Python, is utilized.

It also provides an object-oriented API that enables the integration of static plots into programs utilizing the many Python GUI toolkits that are available (Tkinter, PyQt, etc).

Chapter 5

ML Models for business continuity Prediction

This thesis research is the follow up to the paper "A Machine Learning-based DSS for Mid and Long-Term Company Crisis Prediction" by Guido Perboli and Ehsan Arabnezhad published in July 2020.

Decision Support System(DSS)

ARISK's Decision Support System (DSS) was utilized for financial risk prediction. ARISK is a 2017-founded, Milan and Turin-based, fin-tech spin-off of Politecnico di Torino that provides business interruption prediction services to small and medium-sized enterprises. A training and tuning module and a prediction server comprise the two elements of the DSS system.

Financial risk prediction was conducted using ARISK's Decision Support System (DSS). ARISK is an innovative start-up firm located in Milan and Turin that was created in 2017 and is a spin-off of Politecnico di Torino's fin-tech division. It provides SMBs with business interruption prediction services. The DSS system consists of two distinct components: a training and tuning module and a prediction server.

The training and tuning module data were acquired from the Italian Camera di Commercio, which provides the public database, in public financial data, whilst the financial indexes and ratios were calculated using financial data given by AIDA Bureau van Dick. ARISK's proprietary interface was utilized concurrently to obtain additional data. Following the collection of data, the data cleansing and fusion processes are implemented. Thus, the data is grouped into core and non-core categories. Non-core data cannot be utilized directly in the machine learning module,

but core data can. Non-essential data might be seen as qualitative data. During the machine learning process, fundamental data is analyzed. First, the Feature Selection technique is executed to achieve feature reduction, followed by the use of the suitable machine learning algorithm. This DSS supports a variety of machine learning methods, including Random Forest, XGBoost, Logistic regression, and Neural networks. According to company-collected data, 150 distinct financial factors were identified during the initial creation of the data collection. However, if a financial feature does not affect the classification by more than 1 percent in each step, it is eliminated in order to use these financial features effectively and obtain an efficient output. Using the iterative feature removal process, the 15 most important financial features were identified. As stated in a number of prior publications, these financial elements are not explicitly provided under the Non-Disclosure Agreement.

The outputs of this system are then passed to the prediction module. On the other hand, non- core data is considered secondary data and is used as perturbations of the Machine Learning features, since it cannot be combined directly with Machine Learning predictor.

Following the creation of the risk matrix based on the company's data, a request is submitted to the prediction server using REST APIs. After the prediction server verifies the data, the core data are passed to the machine learning model and the non-core data are processed by the risk impact matrix. Along with the accompanying oscillations of the core data, the predictive impact created by non-core data prediction is conveyed to machine learning. For each core and non-core data collection, five distinct 12-, 24-, 36-, 48-, and 60-month forecasts are created. In addition, international and national rules are investigated, and performance indices are compiled and presented in the report. [8]

5.1 Approach

In current research, different types of both simple machine learning and deep learning models are built, using experimental data set provided by ARISK company. Data set consists of 20000 entries and 84 feature columns.

5.1.1 Machine learning approach

This approach consists of several consecutive steps, namely, preprocessing, model training , testing and assessment of the model.

Dataset

Dataset provided by ARISK was already preprocessed, so preprocessing is skipped, and it made the process faster.

Splitting

It's not a good idea to feed the model with the whole dataset. Because, before deploying model into production, one should be sure about the performance of his/her model. As a result, when one trains his/her model with the whole dataset, he/she has nothing reliable left to evaluate.

Splitting dataset into parts is a good idea in that case. Good practice requires trainset, validation, and test sets. By doing this, there is now a reliable dataset that the model has never seen before during the raining phase.

```python
def split(df,train_ratio = 0.85):
    """
    function takes dataset and train set ratio,
    returns train,validation and test features and labels
    """
    X = df.drop(['PIVA','LivRischioNum','percCard','percRisk'],axis=1)
    y = df['LivRischioNum']
    X_non_test,X_test,y_non_test,y_test = train_test_split(X,y,train_size=train_ratio)
    X_train,X_valid,y_train,y_valid = train_test_split(X_non_test,y_non_test,train_size=train_ratio)


    return X_train,X_valid,X_test,y_train,y_valid,y_test
```

Python

Figure 5.1

Figure 5.1 illustrates function, which split dataset, first into t*est* and *non-test* sets. Then non-test set also split into two, namely *train* and *validation* sets.

Trainset will be used to train the model; validation set will be used to validate the model. Test set will be used to test our model prior to deployment.

Model selection and hyper parameter tuning

In machine learning tasks, there are 2 types of parameters. The one, called hyper parameter, which is chosen prior to training process by researcher. Second one, called model's parameters, will formed, and optimized during training phase.

In short, structure of the model, as well as its performance will be dependent on chosen hyper parameters.

There are a lot of combination of available hyper parameters to choose. If it's done manually, it takes enormous amount of time. But there are several ways of automation of this process. One of them is GridSearchCV by Scikit-Learn library. This is used in this research.

There are several possible candidates for the current regression task. But, for the sake of time and efficiency, the most efficient models are chosen. the *Random-Forest model*, *Gradient boosting* tree models, and *Support Vector Machine* models.

The Random Forest model is an ensemble of several Decision-Tree models. It uses bootstrapping (which means, subsamples are constructed out of the whole dataset with replacement. Each decision tree in a random forest ensemble is fed with different sub-samples. This can help to avoid overfitting problems) and aggregation (each decision tree in a random forest ensemble, generates prediction. In regression tasks, the sum of prediction of all trees is calculated, then an average of it becomes the final answer).

Gradient boosting models are also based on decision trees, but there is a difference between random forest and these types of models. They are sequential, rather than parallel. This means a successive tree will be built based on the loss of the previous tree, by optimizing it. They are fast and efficient models.

The most complex model among candidates is support vector machines, SVM in short. They are very sophisticated models; it means they are tented to be slow and resource demanding.

The next task is hyper-parameter tuning. For these tasks, one can use GridSearchCV offered by Scikit-Learn library. It takes chosen hyperparameters of the model as key-value pairs. The easier way of providing these values is by constructing a dictionary with parameter names as keys and possible numerical values as values.(*Figure 5.2*)
For tree models, both random forest and gradient boosting, the number of trees in the forest (in the sequence), and the max depth of each tree are hyper parameters. the Default value for the number of estimators (trees) is 100. For this task 100, 200, and 300 are chosen as possible values. For the max depth of trees, 1, 3, 5, and 10 are chosen.

```
forest = RandomForestRegressor()
xg_booster = XGBRegressor()
X_train,X_valid,X_test,y_train,y_valid,y_test = split(df)
svr = SVR()

models = {
            'Random Forest model':
                {
                    'model': forest,
                    'params': {
                        'n_estimators':[100,200,300],
                        'max_depth':[1,3,5,10]

                    }

                },

            'XG boost':
            {
                'model': xg_booster,
                'params': {
                    'n_estimators':[100,200,300],
                    'max_depth':[1,3,5,10]

                }

            },
            'Support Vector':
            {
                'model':svr,
                'params': {
                    'kernel':['rbf','poly','sigmoid'],
                    'C':[1,5,10],
                }

            },
}
```

Figure 5.2

For Support Vector Machine models its kernel type and constant C, regularization parameter are hyperparameters. For kernel, radial, polynomial, and sigmoid types are chosen as possible values. 1,5 and 10 are for regularization parameter.

Hyper parameter tuning process revealed that, gradient boosting tree model preform slightly better than random forest model. As one can see from the result, best score of the XGBoost model is -0.1048 which is slightly better than Random Forest model score of -0.1156. similar situation can be seen on test sample, -0.1023 to -0.1147.

	name	model	best_score	best_parameters	accuracy_on_test
1	XG boost	XGBRegressor(base_score=None, booster=None, co...	-0.105473	{'max_depth': 3, 'n_estimators': 300}	-0.104247
0	Random Forest model	RandomForestRegressor()	-0.115868	{'max_depth': 10, 'n_estimators': 200}	-0.114401

Figure 5.3

```
scores = []

for model_name,param in models.items():
        grid_clf = GridSearchCV(param['model'],param['params'],scoring='neg_mean_absolute_error')
        grid_clf.fit(X_train,y_train)
        acc = grid_clf.score(X_valid,y_valid)



        print(f'{model_name} model has completed fitting process')
        scores.append(
            {
                'name':model_name,
                'model': param['model'],
                'best_score':grid_clf.best_score_,
                'best_parameters':grid_clf.best_params_,
                'accuracy_on_test':acc,

            }
        )
sorted_df = pd.DataFrame(scores).sort_values(by=['best_score'],ascending=False)
```

Figure 5.4

Result

Hyperparameter tuning process revealed that, for current task, XGboost has best combination of hyperparameter when:

Max depth = 3 and number of estimators(trees) = 300

Knowing that we can construct tree model and evaluate its performance separately, at the end we can plot its loss values. Evaluation criterion in that case is mean absolute error, we evaluate its performance both in train set and validation set.

```
X_train,X_valid,X_test,y_train,y_valid,y_test = split(df)
clf = XGBRegressor(n_estimators=300,max_depth=3)
eval_set = [(X_train,y_train),(X_valid,y_valid)]
clf.fit(X_train, y_train, eval_metric='mae', eval_set=eval_set)
```

Figure 5.5

Figure 5.6

Figure 1 illustrates loss function values for train and validation sets. From chart it can be seen that in range of 25 to 50, loss values and validation have not that much difference in value. that in range of 25 to 50, loss values and validation have not that much difference in value. After that model starts overfitting. Early stopping can be used to avoid this issue.

Testing

Figure 5.7 illustrates, model performance on test set.

To visualize properly, value of the loss categorized into 3:

- Green – loss value is less than 0.05
- Yellow – loss value is between 0.05 and 0.1
- Red – loss value is greater than 0.1

trend of the value is categorized into to:

- Arrow pointing up – predicted higher than ground truth
- Arrow pointing down – predicted lower than ground truth

Figure 5.7

5.1.2 Deep learning approach

As deep learning is taking over simple machine learning approaches, it is good idea to test it in practice.

In recent days, PyTorch library offers huge set of tools in that field, so it's been chosen to do it.

Process

In PyTorch environment, training the model and evaluating consists of several steps, namely, building custom dataset and data loaders, choosing specific criterion (loss function) and optimizer, model structure and training loop.

Pytorch Custom dataset

Pytorch has built in dataset class which help manage the dataset easily, when used correctly saves a lot of time.

In custom dataset, we need specify features (x in current case) one will use to feed model with, and target which is one wants to predict. When initiating those values, it's needed to transform them into PyTorch float value. It's also required to initiate

```
class CustomDataset(Dataset):
    def __init__(self,x,y):
        self.x = torch.tensor(x.values,dtype=torch.float)
        self.y = torch.tensor(y.values,dtype=torch.float)

    def __getitem__(self,ind):
        return self.x[ind],self.y[ind]

    def __len__(self):
        return len(self.x)
```

Figure 5.7

__getitem__() function, it takes index as an argument, which in the process of feeding , it used to fetch desirable item of dataset. *__len__()* method is used to get total size of the dataset. *(Figure 5.3)*

Data Loader

In deep learning tasks, usually, large datasets is used. It creates memory limitation issues. When model is fed with all items of dataset, it becomes too large for RAM to handle all data. To avoid this problem, it's suggested to feed model with dataset as batches of samples. PyTorch offers custom data loader feature.

```
trainset = CustomDataset(X_tr,y_tr)
validset = CustomDataset(X_val,y_val)
testset  = CustomDataset(X_ts,y_ts)
```

Figure 5.8

```
trainloader = DataLoader(trainset,batch_size=100,shuffle=True)
validloader = DataLoader(validset, batch_size=100,shuffle=False)
testloader  = DataLoader(testset, batch_size=100,shuffle=False)
```

Figure 5.9

Data Loader takes custom dataset (custom dataset object), desirable batch size(integer) and binary value for shuffle (creating random samples as batches from dataset). For performance level of ordinary personal computers, laptops batch size of 100 is optimal choice. For training dataset, shuffle is set to True, as result model is fed randomly

created samples. For validation, it's not mandatory to generate them randomly and feed the model. *(Figure 5.9)*

Model construction

Deep learning model, in our case, multilayer perceptron (a.k.a. MLP), consists of input

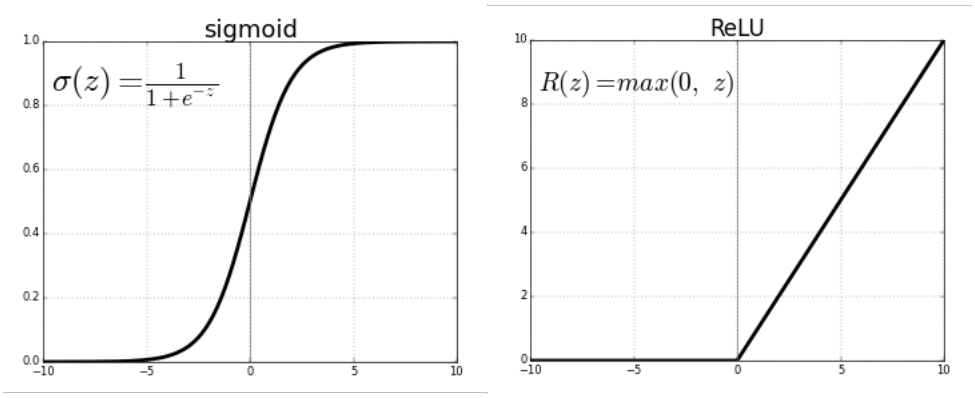

Figure 5.10

layer, hidden and output layer. Number of neurons in input layer should be equal to number of features of training dataset. Depending on whether classification or regression, neurons in output layers varies from 1 to number of categories in target. In current case, problem is regression task, so output layer neuron number is one. Activation function also plays huge role in this process. For current task, ReLU() is used as activation function.(usage of sigmoid function as activation function is also considered). ReLU function returns positive values as it is, returns 0 when value is negative. Sigmoid on the other hand, returns values between 0 and 1, which sometimes brings vanishing gradient problem. Dropping random neurons during training is also good practice. It may help to make model robust.

Unlike ordinary machine learning models, in deep learning, there is systematic and certain way of hyper parameter tuning, something like Grid Search CV.

The best solutions come with experience; it means different tasks requires different approaches of hyper parameter modification. For this current tasks, 3 different models with different depth are chosen.

First one simpler model, with depth of 3. *(Figure 5.11)*

```
model  = nn.Sequential(OrderedDict([
    ('lin1',nn.Linear(80,64)),
    ('act1',nn.ReLU()),
    ('lin2',nn.Linear(64,16)),
    ('act2',nn.ReLU()),
    ('lin4',nn.Linear(16,1)),
]))
```

Figure 5.11

Second MLP model, becomes little bit complex with the help of drop out of random neurons, and it comes with depth of 5.*(Figure 5.12)* The last one has dropouts and depth of 7.*(Figure 5.13)* By choosing different depth, we can find out whether performance of the model depends on it or not. Additionally, different learning rates are used, but it showed that it only slightly smooths out the convergence path but doesn't change minimum point value of the loss function.

```
model = nn.Sequential(OrderedDict([
    ('lin1',nn.Linear(80,64)),
    ('act1',nn.ReLU()),
    ('drop',nn.Dropout(0.4)),
    ('lin2',nn.Linear(64,64)),
    ('act2',nn.ReLU()),
    ('drop2',nn.Dropout(0.2)),
    ('lin3',nn.Linear(64,32)),
    ('act3',nn.ReLU()),
    ('drop3',nn.Dropout(0.2)),
    ('lin4',nn.Linear(32,16)),
    ('act4',nn.ReLU()),
    ('lin5',nn.Linear(16,1))
]))
```

Figure 5.12

```
model = nn.Sequential(OrderedDict([
    ('lin1',nn.Linear(80,64)),
    ('act1',nn.ReLU()),
    ('drop',nn.Dropout(0.4)),
    ('lin2',nn.Linear(64,32)),
    ('act2',nn.ReLU()),
    ('drop2',nn.Dropout(0.3)),
    ('lin3',nn.Linear(32,16)),
    ('act3',nn.ReLU()),
    ('drop3',nn.Dropout(0.2)),
    ('lin4',nn.Linear(16,8)),
    ('act4',nn.ReLU()),
    ('drop4',nn.Dropout(0.1)),
    ('lin5',nn.Linear(8,8)),
    ('act5',nn.ReLU()),
    ('lin6',nn.Linear(8,4)),
    ('act6',nn.ReLU()),
    ('out',nn.Linear(4,1)),
]))
```

Figure 5.13

Criterion and Optimizer

Loss Function, now called criterion, is essential part of machine learning. It evaluates the loss during learning, according to the values of it performance and model fit can be assessed.

```
criterion = nn.L1Loss()
optimizer = torch.optim.Adam(model.parameters(),lr=lr)
```

Figure 5.14

It's talked about above that dataset is batchified because of memory limitation problem. Batches also have effect on gradient decent process. When model is fed with all data at once path in gradient decent is smoother, but when it's fed with small parts of dataset, path during gradient decent is coarse and has zig zag like nature. To smooth out this path, optimizer can be used.

Training Loop

In Pytorch, to train deep learning model, training loop is used. Number of cycles are called epochs. It's exactly equal to how many times model is fed with dataset repeatedly. In each epoch we can train, and validate at the same time, comparing performance of the model on training and validation sets.

```
def train(epochs,trainloader,validloader,model,criterion,lr):
    optimizer = torch.optim.Adam(model.parameters(),lr=lr)
    """
    train function takes number epochs [int], dataloaders, model
    loss function ,optimizer and learning rate as input,
    returns trained model

    """
    losses_tr = []
    losses_val = []
    for epoch in range(epochs):
        model.train()
        print(f'running [{epoch+1}/{epochs}] epoch')
        epoch_loss_tr = []
        epoch_loss_val = []
        for i,(x_tr,y_tr) in enumerate(iter(trainloader)):
            optimizer.zero_grad()
            pred_tr = model.forward(x_tr)
            loss_tr = criterion(pred_tr,y_tr.unsqueeze(-1))
            #back prop
            loss_tr.backward()
```

Figure 5.15

In each cycle, optimizer. zero_grad() is used to avoid accumulated gradient problem. Model, then is fed with input, using forward method prediction can be gotten. After

prediction step, criterion function is used to calculate numerical value of loss. When loss value becomes available, backpropagation process will be executed. In PyTorch Computation Graphs using autograd, automate backpropagation process.

After backpropagation is executed, optimizer's step method is used to update weights of each neuron. This steps continues to be executed until epochs is reached.

```
            loss_tr.backward()
            #optimizing weights
            optimizer.step()
            #saving loss values
            epoch_loss_tr.append(loss_tr.item())
        losses_tr.append(np.mean(epoch_loss_tr))

        model.eval()
        with torch.no_grad():
            for j,(x_val,y_val) in enumerate(iter(validloader)):
                pred_val = model.forward(x_val)
                loss_val = criterion(pred_val,y_val.unsqueeze(-1))
                epoch_loss_val.append(loss_val.item())
            losses_val.append(np.mean(epoch_loss_val))

    return losses_tr,losses_val,model
```

Figure 5.16

For current task, epochs number is chosen as 100. After each epoch, training and validation losses is printed.

```
losses_tr,losses_val,model= train(epochs=100,trainloader=trainloader,\
                                  validloader=validloader,model=model,criterion=criterion,lr=0.0005)
```

Figure 5.17

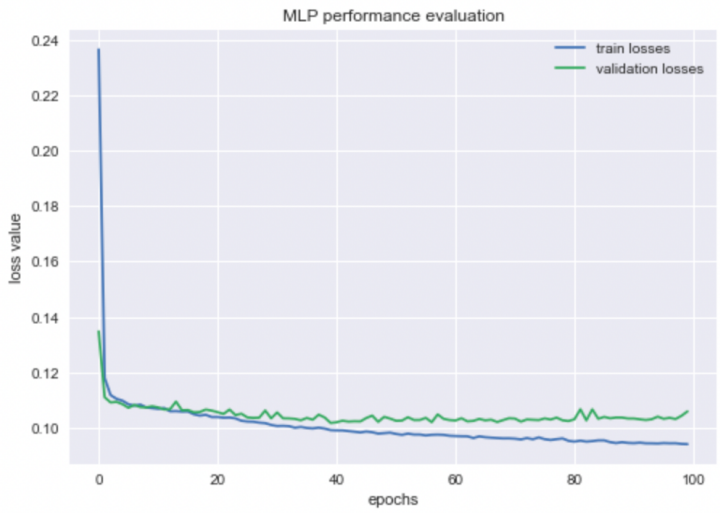

Figure 5.18

Figure 5.18-5.20 illustrate relationship between epochs and loss value at that epoch. Model with depth of 3, had some overfitting issues, the other 2 had smooth convergence through out training.
Comparing 5 and 7 depth, one can conclude that depth has not that much effect on

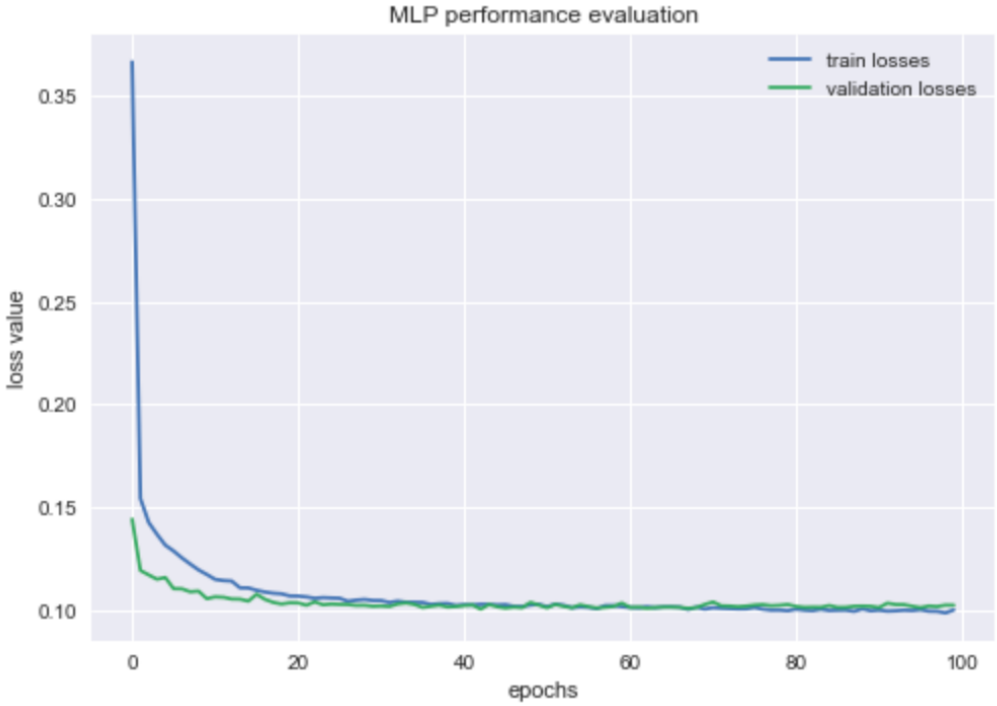

Figure 5.19

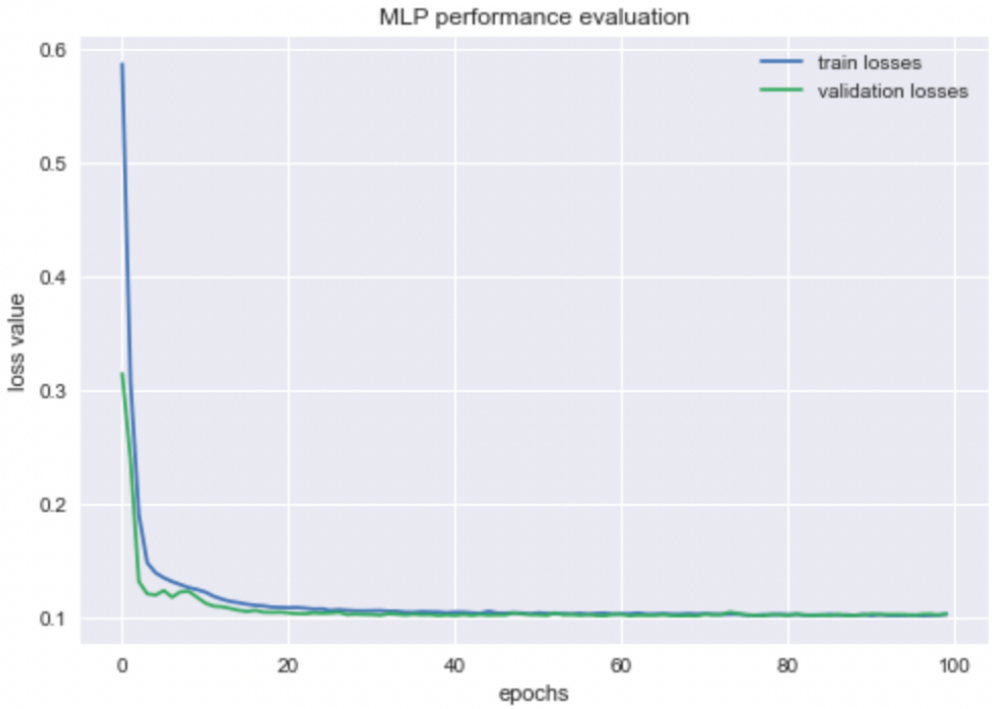

Figure 5.20

local minima as well.

Testing

After training and validation assessment of the model, testing phase comes. MLP model gets tested on test set. Figure 5.21 shows that, loss values have balanced distribution.

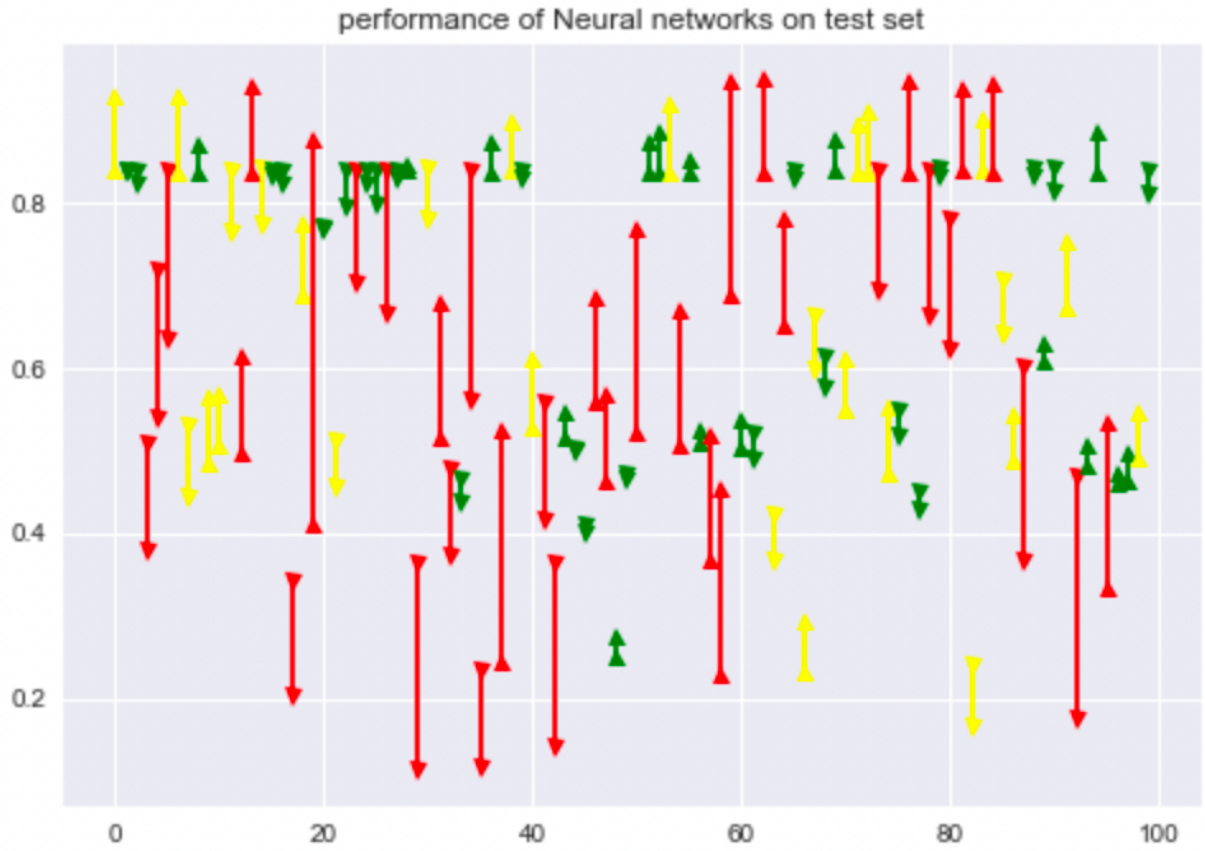

Figure 5.21

5.2 Conclusion

In this research several models, namely, XGBoost, Random Forest and Multi-layer Perceptron (Neural Networks) are trained, then assessment is proceeded using validation and test sets.

When it comes to simple machine learning approach for this research, as XGBoost shows slightly better result, it's chosen for further assessment. For Deep Learning approach, models with several depth is trained and tested. MLP with depth of 5 shows better result in assessment process, so this model is chosen for further comparison.

Figure 5.6 shows loss function values of XGBoost model. One can see from the graph that this model tends to overfitting. On the other hand, it's easy and fast to train and overfitting can be avoided with the help of early stopping during training process. *Figure 5.20* illustrates the loss values for training and validation phases of the MLP model. Graph shows that, MLP model converges its local minimum very fast, just after 10s epochs. It's robust when it comes to the overfitting issues. One can say that model performance can be also better if there is bigger datasets.

Bibliography

[1] E. Verma. [Online]. Available: https://www.simplilearn.com/financial-risk-and-types-rar131-article.

[2] "project risk manager," [Online]. Available: https://www.project-risk-manager.com/blog/qualitative-risk-techniques/.

[3] Washington State Department of Transportation, Project Risk Management Guide, 2018.

[4] D. Hillson, "Effective Opportunity Management for Projects," 2003. [Online]. Available: https://www.google.it/books/edition/_/bZDUDwAAQBAJ?hl=en&sa=X&ved=2ahUKEwjQnNa4iK74AhXjwQIHHfawCVYQ7_IDegQIGBAC.

[5] Brittanica, "Brittanica- Artificial Intelligenc," [Online]. Available: https://www.britannica.com/technology/artificial-intelligence.

[6] P. J. K. M. Han J, Data mining: concepts and techniques., Amsterdam: Elsevier, 2011.

[7] J. P. a. C. N. Sebastian Raschka, "Machine Learning in Python: Main developments and technology trends in data science, machine learning, and artificial intelligence".

[8] G. A. E. Perboli, "A Machine Learning-based DSS for mid and long-termcompany crisis prediction," *Expert Systems with Applications,* 2021.

[9] IBM, "IBM - what is aritificial intelligence," [Online]. Available: https://www.ibm.com/cloud/learn/what-is-artificial-intelligence.

[10] S. IH., Deep cybersecurity: a comprehensive overview from neural network and deep learning perspective, SN Comput Sci, 2021.

[11] W. P. K. A. Sarker IH, "Effectiveness analysis of machine learning classification models for predicting personalized context-aware smartphone usage," [Online]. Available: https://link.springer.com/article/10.1186/s40537-019-0219-y.

[12] H. M. M. U. T. A. Sarker IH, "Mobile data science and intelligent apps," in *concepts, ai-based modeling and research directions.*.

[13] M. A., Information extraction: distilling structured data from unstructured text..

[14] B. E. L. W. G. A. Tavallaee M, " A detailed analysis of the kdd cup 99 data set. In. IEEE symposium on computational intelligence for security and defense applications," [Online]. Available: https://scholar.google.com/scholar_lookup?title=IEEE%20symposium%20on%20computational%20intelligence%20for%20security%20and%20defense%20applications&journal=IEEE&volume=2009&pages=1-6&publication_year=2009.

[15] V. G. G. A. M. V. T. B. G. O. B. M. P. P. W. R. D. V. Pedregosa F, Scikit-learn: machine learning in python.

[16] L. P. John GH, Estimating continuous distributions in bayesian classifiers. In: Proceedings of the Eleventh conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc, 1995.

[17] V. H. J. LeCessie S, Ridge estimators in logistic regressio.

[18] K. D. A. M. Aha DW, Instance-based learning algorithms.

[19] S. S. B. C. R. K. M. Keerthi SS, Improvements to platt's smo algorithm for svm classifier design.

[20] F. J. S. C. O. R. Breiman L, Classification and regression trees, CRC Press, 1984.

[21] A. C. J. H. K. A. A. Y. K. S. B. Sarker IH, a behavioral decision tree learning to build user-centric context-aware predictive model.

[22] H. RC, "Very simple classification rules perform well on most commonly used datasets," [Online]. Available: https://link.springer.com/content/pdf/10.1023/A:1022631118932.pdf.

[23] F. E. Witten IH, Data Mining: Practical machine learning tools and techniques, Morgan Kaufmann, 2005.

[24] F. E. T. L. H. M. H. G. C. S. Witten IH, Weka: practical machine learning tools and techniques with java implementations.

[25] P. ML, Markov decision processes: discrete stochastic dynamic programming, John Wiley & Sons, 2014.

[26] H. A. M. C. G. A. S. L. V. D. D. G. S. J. A. I. P. V. L. M. Silver D, Mastering the game of go with deep neural networks and tree search.

[27] S. IH, "Deep cybersecurity: a comprehensive overview from neural network and deep learning perspective," *SN Comput Sci,* 2021.

[28] J. McCarthy, "WHAT IS ARTIFICIAL INTELLIGENCE?," 2004.

[29] J. McCarthy, "WHAT IS ARTIFICIAL INTELLIGENCE?," 2004.

[30] S. R. a. P. Norvig, Artificial Intelligence: A Modern Approach.

[31] I. H. Sarker, Machine Learning: Algorithms, Real-World Applications and Research Directions.

[32] T. Oliphant, "Python for scientific computing.," in *Computing in Science and Engineering*, 2007, pp. 9,10-20.

[33] P. Virtanen, R. Gommers, T. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright and others., ". SciPy 1.0: fundamental algorithms for scientific computing in Python".*Nature Methods 2020.*

[34] "Authors, N. NumPy receives first ever funding, thanks to Moore Foundation," [Online]. Available: n. https://numfocus.org/blog/ numpy-receives-first-ever-funding-thanks-to-moore-foundation.