Introduction to Artificial Intelligence and Machine Learning with
Altair Knowledge Works™

1st Edition December 2020
# Table of Contents

Preface ........................................................................................................................................... 6
Acknowledgment .......................................................................................................................... 4
Disclaimer ...................................................................................................................................... 5

1 Introduction to artificial intelligence ......................................................................................... 9
   1.1 Historical Perspective of Artificial Intelligence .................................................................. 11
   1.2 The reason behind the rise of Artificial Intelligence ....................................................... 17

2 Application of artificial intelligence .......................................................................................... 19

3 Types of artificial intelligence and its analogies ......................................................................... 22
   3.1 Analogies of AI: Machine Learning, Deep Learning, and Data Science ......................... 24

4 AI Application In Mechanical Engineering ............................................................................... 27
   4.1 PRODUCT DESIGN: For design optimization and efficacy ............................................... 28
   4.2 MATERIAL SCIENCE: To engineer materials’ properties ................................................. 31
   4.3 Fault Diagnostics of mechanical system ......................................................................... 33
   4.4 Reverse Engineering ...................................................................................................... 34
   4.5 Artificial Intelligence in Design Simulation ...................................................................... 35
       4.5.1 The Triumph of simulation ...................................................................................... 35
       4.5.2 From credit cards to product design .......................................................................... 36
       4.5.3 How does ML work with simulation? ........................................................................ 37
       4.5.4 What about accuracy? ............................................................................................... 39
       4.5.5 The road ahead and challenges ................................................................................ 40

5 Programming languages in AI .................................................................................................... 42
   5.1 Python ................................................................................................................................ 44
   5.2 R-programming .................................................................................................................... 44
   5.3 Java ..................................................................................................................................... 45
   5.4 C++ ....................................................................................................................................... 45
   5.5 MATLAB ............................................................................................................................... 46
   5.6 LISP ...................................................................................................................................... 46
   5.7 PROLOG ................................................................................................................................. 46

6 The Machine learning Process ................................................................................................... 50
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Altair Knowledge Studio™</td>
<td>99</td>
</tr>
<tr>
<td>12</td>
<td>Machine Learning with Knowledge Studio</td>
<td>116</td>
</tr>
<tr>
<td>13</td>
<td>Conclusion</td>
<td>163</td>
</tr>
<tr>
<td>14</td>
<td>Reference</td>
<td>170</td>
</tr>
</tbody>
</table>
Acknowledgment

This book is a result of the joint effort of many colleagues who contributed in numerous different ways to get this edition published.

- **Pritish Shubham** the Author of this eBook, for the entire content
- **Rahul Ponginan** for detailed review, inputs and editing
- **Matthias Goelke** guidance and inputs throughout
- **Koushik Chandrashekhar** for feedback and formatting
- **Dr. Armin Veitl, Priyanka Nagaraj, Lekha Janardhan, Prasanna Kurhatkar and Sourav Das** for their constructive comments and warm encouragement.
- **Nelson Dias, Pavan Kumar CV, Vishwanath Rao, Mike Heskitt and Sean Putman**, for their support.
- The entire **Knowledge Works Documentation Team**, for putting together the many pages of documentation.

Please note that a commercially released software is a living “thing” and so at every release (major or point release) new methods, new functions are added along with improvement to existing methods. This document is written using Knowledge Works 2020, Any feedback helping to improve the quality of this book would be very much appreciated.

Thanking you,

**Pritish Shubham** (Assistant Professor at Amity University Noida, India)  
Altair Ambassador ([pshubham@amity.edu](mailto:pshubham@amity.edu))

**Dr. Matthias Goelke** (Senior Director Technical Sales Altair Channel Partner Program)
Disclaimer

Every effort has been made to keep the book free from technical as well as other mistakes. However, publishers and authors will not be responsible for loss, damage in any form and consequences arising directly or indirectly from the use of this book. © 2020 Altair Engineering, Inc. All rights reserved. No part of this publication may be reproduced, transmitted, transcribed, or translated to another language without the written permission of Altair Engineering, Inc. To obtain this permission, write to the attention of Altair Engineering legal department at:

1820 E. Big Beaver, Troy, Michigan, USA, or call +1-248-614-2400.
Preface

Humans have always desired to create machines that can perform the task intelligently without their inventions. Ancient Greek history has several such examples that show the concepts of machines that have intelligence; however, these concepts cannot be implemented, and thus, just remained as stories. In the early 20th century, with the development in neurology, the working of humans’ brains was described, and it shows that the brain consists of the electrical networks that fire signals in the form of pulses. Alan Turing, in the 1950s, showed that any form of computation could be described digitally. He also gave the theory of how a machine can be considered as an artificially intelligent machine – the Turing Test. This is the period when Artificial Intelligence is considered to come into existence.

Artificial Intelligence field, with an enriching history of more than 7 decades, that initially started with an academic perspective is now leading the 4th Industrial Revolution. Although the business application of Artificial Intelligence started way back in the 1980s, where it was used for predicting the stock prices but AI could not gain pace due to several factors such as lack of computational powers of the computers. Due to the emergence of new technologies and high-end computers, Artificial Intelligence has seen a surge in its applicability in recent years. Its application can be seen in numerous technologies such as self-driving cars, personal assistant robots, gaming, health care, banking, Chatbots, Social Media, etc.

The field of AI has not yet seen its peak and is still in the developing stage. Knowing its potential, AI will continue to grow in the coming years and numerous new technologies based on AI will emerge in near future. AI will play a pivotal role in improving the quality and will allow humans to do tasks more efficiently. The future of AI is enormous and due to the fact, many are now willing to gain expertise in this field. Those who wish to learn this field of Artificial Intelligence faces two main challenges – Where to begin from and How to implement Artificial Intelligence with
less or no knowledge of programming/coding. The book “An Introduction to Artificial Intelligence – A purview of Knowledge Works” is written to solve both the issues.

This book is written for those who are willing to explore the field of Artificial Intelligence and its allied field of Machine Learning and Deep Learning but are in the nascent stage and unable to determine the pathway of their learning curve. The book helps the readers to understand the theoretical concepts of AI as well as how to implement them in real problems in various ways. The book contains a total of 12 Chapters: 1 to 9 deals with conceptual learning and their field of applications and 10 to 12 chapters deal with the implementation of the concept through the Knowledge Works™ platform. A brief layout of each chapter are mentioned below:

Chapter 1 deals with the evolution of Artificial Intelligence; its historical perspective, the present applications and future trend. The chapter also identifies the reason why artificial intelligence has started gaining pace in the past decade and so.

Chapter 2 reviews the various applications of Artificial Intelligence in today’s world. It helps in understanding how the AI field has encompassed new horizons, without making us aware of it.

Chapter 3 discusses the various categories of Artificial Intelligence such as Narrow, General and Super Intelligence. This chapter also helps in understanding the various analogies of this field i.e. Artificial Intelligence, Machine Learning, Deep Learning and Data Science.

Chapter 4 specifically discusses the application of Artificial Intelligence in Mechanical Engineering. The chapter reviews various ways of implementing AI in different mechanical domains such as Product Design, Material Science, Reverse Engineering etc.

Chapter 5 helps in the understanding of the different programming languages that can be used for the application of AI. The chapter also compares these languages
and identifies which one could be the most preferable among all. The chapter also helps to know the various online platform which enables the application of Artificial Intelligence and Machine Learning with minimal knowledge of coding.

Chapter 6 presents the seven-step Machine Learning Process which is to be followed in general. In this chapter, for better understanding of the process, a hypothetical problem is considered and solved through an online programming platform – Jupyter Online Notebook – by using these Machine Learning Steps.

Chapter 7 discusses various categories of Machine Learning – Supervised, Unsupervised and Reinforced machine learning – and shows the mathematical logic working behind each of these. The chapter also discusses different types of algorithms and their usabilities in Machine Learning.

Chapter 8, addressed the issues and the limitations of machine learning. With this chapter, the reader will get to know why Machine Learning can’t be blindly applied to every problem and what are the things to keep in mind while creating effective Machine Learning Models.

Chapter 9 deals with the discussion on Deep Learning, its working and evolution. The chapter also reviews various types of neural network used in deep learning technique.

Chapter 10 gives the overview of the Altair Knowledge Works™ Software and its subsets – Knowledge Studio, Apache Spark, Knowledge Seeker and Monarch. The Knowledge Works software is used for AI and ML application that does not require high skills to create the predictive analysis. The chapter will make readers know how Machine Learning can be implemented without writing a single line of a program.

Chapter 11 reviews the Knowledge Studio platform and shows various tools and modes available in the software for creating Machine Learning Models. This chapter help reader to get acquainted with the Studio Platform.

Chapter 12 explains how to create the Machine Learning predictive analysis by using Knowledge Studio with an example. Through this chapter, the Decision Tree algorithm is explained by using a market campaign problem.
1 Introduction to artificial intelligence

“I believe there is no deep difference between what can be achieved by a biological brain and what can be achieved by a computer. It, therefore, follows that computers can, in theory, emulate human intelligence - and exceed it.”  Stephen Hawking

The evolution of intelligence took millions of years to be what it is at present; on the contrary, the evolution of machine and its intelligence just took a few centuries. It won’t be a metaphor to say, machines are inching closer to and may even surpass human intelligence. It all started in the 18th century when human muscular power was replaced by steam-powered machines and production of goods were mechanized. This brought the 1st Industrial Revolution, termed as Industry 1.0. Since then, the world has seen several revolutions, and at present, the trend is of Industry 4.0 where machines are in a position to replace human intelligence. With the desire to make the world smarter, humans have started building the machine’s intelligence – Artificially. This has led to the development of a whole new field of innovative technology termed as “Artificial Intelligence”.

Artificial intelligence, commonly termed as AI, is concerned with the design and development of intelligence in machines artificially. The primary goal of AI is to create systems that can work intelligently and independently. The oxford dictionary defines Artificial Intelligence as “the theory and development of computer systems able to perform tasks normally requiring human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages.”

Intelligence is something that characterizes humans, and it is expected from an intelligent machine, that it mimics humans in the best possible manner for
performing any task. However, the main question is how to define intelligence as it is a situational character that varies significantly based on an individual’s expertise and learnings. To concretize the definition of intelligence, 4 main characteristics of humans are considered for categorizing machines as intelligent; listed as – Act Humanly, Think Humanly, Act Rational, and Think Rational, Fig. 1.1.

![Fig. 1.1. Machine Characteristics to be recognized as Intelligent Machine](image)

**Act Humanly**
- The Machine that behaves like Human
  - The Turing Test

**Think Rational**
- The Machine that can think logically and correctly

**Think Humanly**
- The Machine that thinks alike Humans
  - Cognitive Approach

**Act Rational**
- The Machine that can behaves logically akin to Humans

*Act like Humans* – A machine can be called intelligent, which can behave similar to a human in a given specific condition where both can’t be differentiated. The Turing test was developed to determine its effectiveness in which a human interrogator asks questions to another human and a programmed machine without knowing who is who. If the logical replies of the machine can’t be differentiated from that of humans by the interrogator, then the machine is considered to be an intelligent machine. This characteristic to be demonstrated by a machine, requires a large amount of data, which is then correlated and accessed intelligently through a programming algorithm and used for performing the specified task.

*Think like Humans* – The characteristic of a machine that can enable it to think like a human through a cognitive approach also puts it into the intelligence
category. Although these types of machines are not yet fully developed and are mostly part of the research, deep learning with a neural network is one such algorithm that is extensively used for the machine’s cognitive intelligence.

*Act rationally* – The machine which can act logically and do things right is also considered an intelligent machine. These types of machines interact with their surrounding environment and behave similar to humans to achieve their goal. The generalized rational approach is most commonly used by maximizing the expected performance for making a machine act rationally.

*Think rationally* – When a machine shows the ability to think on its own logically and follow laws that humans follow, considered intelligent machine. This again requires a great amount of data, an efficient algorithm and a high degree of computational power.

When a machine possesses one or more of these 4 characteristics, then it is considered as artificially intelligent machine. An intelligent machine helps the most in those tasks which are of repetitive nature and can hence accomplish more in less time. This is the general expectation from an AI machine which is under its development phase. AI has great potential in future and is the most influential technology of the present time, however, it wasn’t the same a few decades ago. It is also imperative to understand the evolution of AI to appreciate what it is at present.

### 1.1 Historical Perspective of Artificial Intelligence

The concept of Artificial Intelligence goes back to classical ages. In Greek mythology, the machines and mechanical men are well thought of concepts. It’s best proven through the example of “Talos”. Talos was a gigantic animated bronze warrior that was programmed to guard the Crete Island which was created by Hephaestus. Although there is no proof of its exact implementation, Machine Learning and Artificial Intelligence were thought about long ago. A timeline for the evolution of Artificial Intelligence can be seen in Fig. 1.2.

The year 1950 was speculated to be the most significant year for the induction
of Artificial Intelligence; it was the period when the machines had just started to compute and calculate very simple arithmetic. At that time a well-known scientist Alan Turing challenged humanity with the questions - “Can machines think? Can machines have intelligence?” This intrigued many leading computer scientists of that time, and the likes of Marvin Minsky, Joe McCarthy, and many others got together to jump-start the futuristic innovative field that is today known as Artificial Intelligence.

Dr Alan Turing’s research speculated the possibility of creating machines that can think on its own and created the famous “Turing Test” which was used to determine whether a computer can have intelligent thinking like humans or not. To elucidate and categorize the term “Intelligent Thinking”, Dr Alan used the test which states that “if a machine can perform tasks which is indistinguishable from human work performance, then the machine can be considered to be an Intelligent Thinking Machine”.
Famous “Turing Test” was devised by Alan Turing to define the Intelligent Thinking of Machine or a computer system.

Artificial Intelligence was first coined during a conference in Dartmouth by John McCarthy.

The 1st robot was introduced in assembly line of General Motors that can perform human-like work which is repetitive in nature.

IBM developed Deep Blue machine based on AI that defeated the world champion Garry Kasparov in the chess game.

IBM developed a question-answering computer system named WATSON which is capable of answering questions posed in natural language.

Alan Turing

‘AI’ - Word

GM Robo

Deep Blue

Watson


1st AI Game

1st AI Lab

ELIZA

STANLEY

SOPHIA

Intelligent programs for playing Chess and checker game was developed by Dietrich & Christopher respectively.

1st Artificial Intelligence laboratory was established in MIT.

1st chatbot named “ELIZA” was developed in MIT Lab.

A robotic car named Stanley was developed by Stanford’s racing team that won the DARPA grand challenge.

A Social Humanoid developed by Hanson Robotics which can recognize human face and display more than 50 facial expressions.

Fig. 1.2 Evolution of Artificial Intelligence
Although, till date, no machine has fully cleared the Turing Test, it is considered as the first serious proposal of the field of Artificial Intelligence. The following year of 1951 saw further developments in AI and was also called the era of “Game AI”. Christopher Strachey, a computer scientist in the University of Manchester, wrote a program for checkers as well as chess by utilizing the Ferranti Mark1 machine. Although these programs had been further improved and redefined, this was considered to be the first attempt at creating machines that can play chess and compete with humans.

The year 1956 was considered as the birth year of “AI”. During a conference at Dartmouth College, New Hampshire, the term Artificial Intelligence was first used by John McCarthy. Later in the year 1959, the 1st Artificial Intelligence laboratory was established in MIT which marked the period of research and development of AI. During its early years, researchers used general methods for the wide categories of problems for imitating the complex thinking process. These general-purpose search mechanisms do not yield a good outcome, hence referred to as weak methods. Gradually, these weak methods were replaced by new promising learning algorithms that offered great potential, however, due to limited computing capabilities, their real applications were delayed by decades.

The year 1960 saw the introduction of the intelligent Unimate robot in the assembly line of General Motors which can perform tasks like humans. These robots used a magnetic drum for storing step-by-step instructions required for performing the task. The following year of 1961 saw the development of the first natural language processing chatbot named “ELIZA” in MIT Lab; Fig. 1.3(a). The chatbot was designed for communication between machine and human, superficially. It used a pattern matching and substitution methodology which gave an illusion of machine communication, although it does not have any inbuilt framework for contextualizing the event. Later in the year 1997, IBM developed Deep Blue machine based on AI which is most famous for defeating the world champion Garry Kasparov in a chess game; Fig. 1.3(b). It used brute computing power as its playing strength. This was considered as the 1st greatest accomplishment of AI machines that can think on its own.
In the year of 2005, AI achieved another feat when a robotic car named Stanley won the DARPA grand challenge; Fig. 1.3(c). Stanley was designed by Stanford University’s racing team in collaboration with Volkswagen Research Laboratory. Later in the same year, an artificially intelligent humanoid robot named ASIMO (Advanced Step in Innovative Mobility) was developed by Honda; Fig. 1.3(d). ASIMO was capable to walk like humans with speed and balance itself when faced with hurdles. It can respond to basic questions and can recognize human gestures.
Xbox-360 - an award-winning device that can track human body motion by machine learning algorithm was developed by Microsoft. It was the first gaming device that enables the user to play wirelessly. The next big accomplishment of AI was in the year 2011 when IBM’S WATSON, Fig 1.3(e); a question-answering computer system capable of answering questions posed in natural language, won the famous quiz show “Jeopardy” and a prize of $1 million.

The most recent advancement in AI field is “SOPHIA” – a Social Humanoid, which was developed by Hanson Robotics, by inventor David Hanson, in the year 2016; Fig 1.3(f). Sophia is the most advanced robot which works on AI and is capable of recognizing the human face and display more than 50 facial expressions. It is the 1st AI-driven humanoid robot to have received the citizenship of any country – Saudi Arabia; during an innovation conference. The architecture of Sophia consists of a chat system, scripting software and AI algorithm. ELIZA and Sophia both are conceptually similar and were designed for simulating a human conversation.

The above-mentioned examples are just a few among many of the most important developments in the evolution of Artificial Intelligence since its emergence in the 1950s. It started with a hypothetical situation during its early years but in today’s world, AI has become the most significant technology. If a careful observation is made, then one can find things all around us presently using AI – Deep Learning or Machine learning.
1.2 The reason behind the rise of Artificial Intelligence

Artificial Intelligence covers the domains of Machine Learning, Deep Learning, Neural Networks and Natural Language Processing (NLP) as shown in Fig 1.4. AI has also made its way into Computer Vision and Image processing. Artificial intelligence is probably one of the biggest scientific breakthroughs in the 21st century and it will give the power to probe the universe and humanity with a different approach.

![Fig. 1.4. Domains of Artificial Intelligence](image)

AI has immense potential to redefine future technology and has seen exponential growth in recent years compared to its inception year and has already been touching our lives in several ways. The main question is - if AI was present for over half a century, why has it gained so much importance in recent years? Why AI is the most talked about topic and what is the main reason it has made its insurGENCY in recent times? The most notable reason that could justify its insurGENCY is the rise of computation power. Artificial Intelligence requires a lot of computing power and the recent advances have made complex calculations much easier through advanced computers. Due to this the complex deep learning models can easily be deployed and processed by high-power GPUs.

The 2nd most vital reason is that at the present time, data is being generated at an immeasurable pace through social media, IoT and other similar devices. These data play a very crucial role if it can be analyzed and processed systematically to predict the trend for growing the business. Artificial intelligence can be one of the efficient tools that can be utilized for data analysis and prediction for making smart decisions.
AI algorithms are trained on large data sets which makes them most efficient. The convergence of the maturing of statistical machine learning tools, the convergence of big data brought about by the internet, advancement in sensors, and the convergence of advancements in computing and hardware that Moore’s law predicts, has created the best conditions for AI. Furthermore, at present, we have better algorithms than the past and can use efficient prediction of the trend. One of such effective algorithms is “Neural Network” which is the concept behind deep learning. Since we have better algorithms which perform better and quicker computations with more accuracy, the demand for AI has increased significantly.

Another reason is that universities, governments, startups, and tech giants are all investing heavily in AI because of the increase in efficiency it brings and its high capabilities. Companies like Google, Amazon, Facebook, Microsoft, etc. have heavily invested in artificial intelligence because they believe that AI is the future. These are some of the important reasons behind the rapid growth of Artificial Intelligence both as a field of study & research as well as an industry too.

All these factors came together and lifted AI from a nascent stage into what it is at present where AI is making a real impact on the world, and note that this is just the beginning. The future of AI is highly bright and bound to redefine technology and reshape the future.
2 Application of artificial intelligence

The term Artificial Intelligence was first devised by John McCarthy at the Dartmouth Conference in the year 1956 which is considered as the Birth year of AI. McCarthy defined Artificial Intelligence as the science and engineering of developing intelligent machines. Theoretically, AI is the stream which involves the development of machines or systems that are capable of performing tasks that mimics human intelligence such as Visual perception, speech recognition, decision making etc. In other words, AI is a technique of creating machines to work and behave similar to humans. In the recent past, artificial intelligence has been widely used in developing machines and robots that are efficiently utilized in various fields such as healthcare marketing, business analytics, autonomous robots, etc.

One of the most well-known applications of AI is the search engine of Google which gives suggestions and recommendations based on users’ past actions and activities. These recommendations are based on analyzing the data through AI which were collected through users’ browsing history, age, location, and other relevant details. This predictive search by Google involves a lot of deep learning, machine learning and natural language processing methods.

Another very famous application of AI in the finance sector is JP Morgan Chase’s contract intelligence platform which uses a combination of machine learning, deep learning and image recognition software to analyze the legal documents. In general, the manual reviewing of about 10,000 agreements will surely take more than 35,000 working hours, which might involve huge manpower. However, if these tasks were to be replaced by AI Machines, the whole task can be completed in a matter of a few minutes to review and compile the same amount of data. Even though AI cannot have logical thinking like humans, their computational power is very strong compared to any highly efficient human due to powerful algorithmic
models. AI has reached a stage wherein it can compute the most complex problems in a matter of seconds.

An important Artificial Intelligence application is also seen in the field of healthcare in which IBM is among the pioneers for developing the AI software for medicines. More than 250 healthcare institutes and organizations utilize IBM AI-technology which is basically “IBM’S WATSON”. The Watson technology was able to cross-reference more than 20 million oncology records and correctly diagnose the rarest of the rare leukaemia patient’s conditions. Since machines are now used in medical fields as well, it is clear how important AI has become. It has reached every domain of our lives.

The next significant advancement of AI in healthcare is Google’s “AI Eye Doctor” in which Google is working with an Indian eye care chain named “Aravind Eye Care” to develop an artificial intelligence system which can examine retinal scans and identify a diabetic retinopathy condition which can cause blindness.

Social media platforms such as Facebook are also using Artificial Intelligence to recognize faces where the concepts of deep learning, neural networks and machine learning are used to detect facial features for auto-tagging friends. Although we are aware of such features, we are unaware that these are in fact Artificial Intelligence technologies which are being used extensively in everyone’s life.

Another example for the application of AI in social media is Twitter which uses AI for identifying hate speech and terroristic languages in tweets. Again, the concept of deep learning, machine learning and natural language processing was utilized for filtering out any offensive and objectionable content. Very recently, the company discovered around 300,000 accounts with terroristic links and 95% of these were detected through AI machines.

Considering the field of virtual assistants, there are virtual assistants like Siri and Alexa that can not only respond to calls and book appointments for you but also add a human touch. Their added human filters and similar features make it sound very realistic. Many a time, it’s hard to distinguish between a human and AI machine speaking over the phone. Another newly released - Google Duplex, Google's virtual assistant, has astonished the world through its performance.
Another famous application of AI is fully automated self-driven cars which implement computer vision, image detection, deep learning and machine learning techniques to manoeuvre the vehicle and detect any object or obstacles without human intervention. Elon Musk’s Tesla is the leading automobile company in the field of self-driven cars. Tesla also has plans for a Robo-taxi version that can ferry passengers without anyone behind the wheel.

Netflix uses similar concepts of machine learning and deep learning for providing a personalized movie recommendation for individual users based on their previous search results. For its prediction, Netflix studies each user’s personal details and tries to recognize the user’s interests and movie watching patterns. This helps in creating more engagement of the user with the Netflix platform and there’s a known fact that over 75% of what you watch is recommended by Netflix. So, it can be easily concluded that their recommendation engine is working most efficiently.

Apart from Netflix, Gmail also uses AI on an everyday basis. While opening the mail inbox, very distinguishable sections can be found such as primary section, social section, etc. Gmail has a separate section called the spam mails also. The concept of machine learning algorithm is used to classify emails and put them into specific categories. The terms such as money, lottery, earn, etc. will have greater chances to be categorized as spam mails. These words are correlated and understood by machine learning and natural language processing and hence categorized under a specific tag.

These are just a few of the examples of AI application which is just the tip of the iceberg, many more relevant applications do exist at present. However, with these examples, the wide application of Artificial Intelligence and its utility in day to day life of every individual can be clearly understood.
3 Types of artificial intelligence and its analogies

The previous 2 chapters have dealt with the definition, evolution and important applications of Artificial intelligence it is now important to understand the types of Artificial Intelligence which are being used.

Humans are considered to be the most intelligent species on this planet, therefore, to categorize a machine as intelligent, it must be compared with human intelligence. Based on this, Artificial Intelligence is categorized into three different evolutionary stages which are:

1. Artificial Narrow-Intelligence
2. Artificial General-Intelligence
3. Artificial Super-Intelligence

The Artificial Narrow-Intelligence (ANI) is also known as a weak AI which involves the application of AI only to a single task. The ANI machine lacks self-awareness and can perform a specific task exceptionally well and repeatedly for a substantial period of time. Many existing systems which claim to be using Artificial Intelligence are primarily working on ANI, which commonly focuses on a narrowly targeted objective. Alexa is one such example which uses Artificial Narrow Intelligence within an unlimited pre-defined range of functions.

Google’s search engine and Translate, Siri, Alexa, Self-driving cars, Sophia robot, and the famous AlphaGo are some of the significant technologies that use Artificial Narrow Intelligence. Looking at these examples, many would assume that these are very advanced application tools having a high order of intelligence and shouldn’t be categorized as “Weak AI”. But with careful insight, one can realise that these tools do not possess any self-awareness, genuine intelligence and consciousness that can match human intelligence. Hence, termed as Weak or Narrow Intelligence in comparison to human intelligence. One can imagine if such advanced latest
application tools work on Narrow-Intelligence, what could be the actual potential of Artificial General Intelligence or the Artificial Super Intelligence.

Artificial General-Intelligence (AGI) is also known as “Strong AI”. A machine can be categorized under General-Intelligence if it can perform any task as intelligently as a human being can carry out. Undoubtedly, machines don’t possess human-like abilities. Although machines have a high-power processing unit that can perform high-level of computations within a fraction of second, still it is not yet capable of performing simple and most reasonable tasks that any human being can. For instance, a machine can process a million documents in a matter of a few seconds, but it is unable to switch on the Air Conditioning system by walking to the adjacent room or picking up a remote and select a TV channel, on its intelligence.

Although few domains of AI, such as Deep Learning and Neural Network, have pushed the machine’s intelligence close to humans, the machine’s inability to think on its own is still prevalent. Due to this fact, Narrow-Intelligence is vastly used in the majority of AI applications instead of General-Intelligence.

The Artificial Super-Intelligence (ASI) term refers to those machines which have the ability to surpass human intelligence. This implies that if a machine has greater creativity, logical thinking, general wisdom and problem-solving ability than a human, it will be considered an Artificial Super-Intelligent machine. In reality, there is no such machine, and it will certainly take a while to achieve the Super-intelligence stage. At present, Super-intelligence can only be depicted in movies such as Terminator, I-Robot or science fiction books where machines have taken over humans to rule the world.
3.1 Analogies of AI: Machine Learning, Deep Learning, and Data Science

The field of Artificial Intelligence (AI) uses several terms such as machine learning (ML), deep learning (DL), and data science (DS). Although, they seem to be indistinguishable many a time due to their similarity in application and non-availability of any concrete definition which can clearly distinguish these terms are mutually inclusive, and it is imperative to understand how they differ from each other.

Artificial intelligence, machine learning, deep learning and data science are the most common terms widely used by scientists, researchers and technologists who work in the very fast-evolving field of AI. The difference between all these terms can be understood by the help of a Venn Diagram as shown in Fig. 3.1.

Artificial Intelligence is the technology that is concerned with the automation of intelligent behaviour and attempts to build intelligent entities. Artificial intelligence can be considered as a broad umbrella under which machine learning and deep learning techniques work to achieve the objectives of AI.

Machine learning is a subset of artificial intelligence where we teach a machine how to make decisions with the help of input data by using statistical tools. In other words, we can say that the method of training computers to learn patterns from a set of data, and is commonly used for making decisions or predictions is known as Machine Learning. It is frequently used in medical diagnostics such as brain-machine interfaces, chemical informatics, self-driving cars, stock market analysis, and many such areas. It can also be used for recommending the right product to the customer, based on their requirements.
Deep learning is a further subset of machine learning where intelligent algorithms try to mimic the human brain. It implies that the machine learns by imitating the human way of gaining knowledge. It is interesting to know how a machine can mimic the human brain. More importantly, it is also necessary to understand how the brain is composed and works, in order to make the machine behave like the human brain.

The brain is primarily composed of millions of neurons stacked and connected in a layered form that sends and receives electrochemical signals. The signal received by a neuron is processed in its cell body and an output signal is generated, which is then transferred to other neurons through the axon and processed further. These processed signals are then cumulatively used by the human brain to take decisive actions. If an algorithm needs to mimic the human brain, then artificial neurons have to be created that work the same way as biological neurons. To have a greater understanding of Deep Learning concepts, refer to Chapter 9.

Data science is a multi-disciplinary field where scientific methods and algorithms are used to get insight from any unstructured data. Data science is related to Big data, which requires powerful hardware and computational systems.
along with efficient algorithms for finding solutions with the bulk of data. Data science can be considered mutually inclusive with Artificial Intelligence and Machine learning, but the aspects of deep learning remain unhindered
4 AI Application In Mechanical Engineering

Artificial Intelligence, by definition, is a process of gleaning insight by using computer algorithms to go through bulk input data whose end result is prediction. It enables the systems to learn from data training experiences, and the algorithms are used to recognize patterns and structures with supplied data. These learnings are then applied to new and unknown cases. Artificial Intelligence can be applied in any domain where a large amount of data is generated, thus resulting in its extensive usage in the field of engineering. Machine learning is predominantly used in Computer science and Information technology; vital applications - Facebook face recognition tags, YouTube & Netflix recommendations based on the usage pattern of the user, Email Spam and Malware Filtering, Virtual Personal Assistants, etc., are just a few among many.

With further advancement and exploration, Artificial Intelligence is now deemed fit for much wider application areas such as the dominion of Mechanical Engineering too. The influence of AI and its implementation in automation with respect to the modern mechanical engineer is conceivably a hot topic in the industry. Some of the world’s top thinkers and entrepreneur minds, like Elon Musk, have emphasized the advancement of AI and what it might do to the world. Although significant developments in this domain are yet to come, the application of “AI in ME” is gaining popularity at a considerable pace. Fig. 4.1 shows the number of research publications annually since 1990; which highlights the gaining popularity of Machine learning in the domain of Mechanical Engineering - the number of research publications in last 5 years (2012-18) has almost equaled the publications of previous 20 years (1990 – 2012). The report highlights the usage of Machine learning is not new in Mechanical Engineering, and it is gaining significant pace in recent years. The application of ML in Mechanical engineering can drastically enhance the efficiency, flexibility and quality of the mechanical systems through the effective utilization of the available data.
Some of the important facets of Mechanical Engineering where Artificial Intelligence presently plays a crucial role are:

1. Product Design
2. Material Science engineering
3. Fault Diagnostics of mechanical system
4. Reverse Engineering

### 4.1 PRODUCT DESIGN: For design optimization and efficacy

Product design is an analytical process of creating solutions with an interface to improve the quality of life of the user. It is an iterative process – that consumes much of the company’s time – is subdivided into various phases that include sketching and prototyping. Artificial Intelligence application in product design permits the virtual testing of input data through a set of combined efficient learning algorithms and prediction of the performance of prototypes. Although it is challenging to integrate the impact of different parameters into straight “Yes” or “No” decisions, how well AI predicts depends on the capacity to compile and process the information; but still, AI can significantly influence the outcome of the designed product.
Mode of AI application: The design success of the product is dependent on many aspects, like functionality, performance, ergonomics, aesthetics, life cycle, etc. – all systematically documented by expert design engineers to build significant knowledge through several case studies and effective practices. With each iteration of design, its success/failure possibilities are analyzed, which gradually becomes a large chunk of data that has the ability to predict the trend.

To expand on it further, consider an example of the design of a piston for the IC Engine. The primary function of a piston is to transmit the gas force to the connecting rod and bear a significant amount of heat and pressure force. The designing of the piston involves ideation of the concept, creation of the model, analysis and evaluation of the design through virtual and experimental techniques – all of these need to be iterated till a successful design is obtained. With every design iteration, a specific set of data is generated, which concludes the performance of the designed piston. Collectively these data play a vital role in deciding the course of action for designing the future products of similar category.

The senior mechanical design engineers use these data to create case studies that are referred for future design explorations; the efficient usage of such case studies will result in better designing of the product. The role of Artificial Intelligence can be significant for analyzing these data through well-developed machine learning algorithms and predicting the outcome without the need for design iterations – saving a considerable amount of time and cost. However, how well the prediction is will entirely depend on compilation and process information. Artificial Intelligence can also be used for improving the way parts are designed by performing concurrent analysis of physics, solid mechanics, fluid mechanics, manufacturing estimations, etc. Such complex analysis is required to be supervised by skilled engineers to get the most desired outcome consistently. Fig. 4.2 shows an optimized design of bracket.
One of the most advanced software solutions for implementing Artificial Intelligence and Machine Learning in Design Simulation to improve design process is Altair’s “Knowledge Works”. “Knowledge Works” is a set of data intelligence solutions that allow individuals and organizations to incorporate more data, unite more minds with agility, and engender more trust in analytics and data science. The platform enables users to quickly and accurately capture and prepare data for any project, use data to accurately predict outcomes and produce insight and foresight, and visualize trends and insights which are then communicated across the business. Knowledge Works empowers the organization to face challenges with trust, literacy, diversity and complexity of data and removes ambiguity from analytics and data science. To have a better overview of the Knowledge Works software, click on the hyperlinked image, Fig. 4.3, or see Chapter 10.

Fig. 4.2. Design optimization in product design
4.2 MATERIAL SCIENCE: To engineer materials’ properties

Physical properties of materials are dependent on their chemical composition and arrangement of the atoms, any minor change in the two can alter its characteristics significantly. Ever-rising demands of more efficient and better-performing materials have made the material science an area of great interest among engineers and scientists – who try to enhance the properties of well-established materials.

Chemical doping is the traditional way of altering the properties of materials; it involves the addition of impurities to modify properties. On the contrary, a controlled
straining mechanism is relatively new and in trend for altering the materials’ characteristics. It is a well-established fact that the atomic arrangement and its structure can be altered to change the materials’ properties significantly. Although both these methods have a well-established theoretical correlation, it is still crucial that this is validated by experimentation, which is a time-consuming and costly affair.

The advancement in artificial intelligence in the field of Mechanical engineering has empowered scientists to predict and control these changes, thus resulting in the opening of new research prospects on advanced materials for future technologically advanced devices. It provides a systematic way of determining the amount and direction of strain for achieving the given set of properties for specific applications. This helps in creating explicit materials with specific properties needed for electronic and photonic devices that are used for communications, information processing, and energy applications.

The implementation of machine learning principles, in materials science plays a vital role in the efficient prediction of the outcome. Initially, the description of different learning approaches for determining stable materials and predicting the structural arrangement of crystals is done. Thereafter through various quantitative approaches, the relationship between the structure and properties is predicted. Fig. 4.4 depicts the sequence for predicting material properties through Artificial Intelligence. With further data training, optimization and machine learning experience, the prediction of materials’ characteristics become more accurate and efficient; however, the considerations of different facets and its interpretability plays a key role in such predictions.
4.3 Fault Diagnostics of mechanical system

Failure in a mechanical system can occur due to sudden unexpected changes in loading conditions, design and operational errors, or uncontrolled natural reasons. Failure may also occur due to gradual accumulation of material damage, which disperses as crack propagation and causes sudden failure. The failures are also possible due to continuous wear over a long period in mechanical systems such as gear mechanism, piston-cylinder, and valve mechanism of IC engine, etc. It is very imperative to identify such gradual faults, which can result in sudden failure.

Fault diagnostics is concerned with the monitoring of a mechanical system, identifying the occurrence of the fault, determining the location and type of fault, etc. It involves two main approaches, namely - Direct Pattern recognition and Discrepancy Analysis. Direct Pattern recognition deals with the determination of the pattern by reading the data obtained through various sensors, whereas Discrepancy analysis deals with the measurement of the variation between the expected value and the sensor data. Although these approaches ensure the safety of the mechanical system, it requires constant effort to keep diagnosing the possibilities of occurrence of faults and taking preventive steps – which is a costly affair. Moreover, fault diagnosis depends on the extraction of features from signals obtained from the sensors. The nature of feature characteristics directly influences the fault recognition approaches.

The application of artificial intelligence in both the aforementioned approaches can play a pivotal role in fault detection. One of the efficient algorithms in Artificial
Intelligence for fault diagnosis is - Bayesian networks (BN) algorithm. BN model can be effectively used for supervised as well as unsupervised learning which works on cause-effect-relationship approach rather than pattern analysis approach. The data recorded through sensors are bifurcated into training and testing data in which BN algorithmic model is applied to train the dataset for predicting the fault probability.

4.4 Reverse Engineering

Reverse engineering is the process of dissecting the existing object to reveal its designs, architecture, or to extract knowledge from the object. Reverse engineering (RE) of the mechanical part is a method of back-engineering in which a virtual model is created from a real product. The RE-process involves scanning of the part through scanners and generation of cloud data points. Thereafter, these data points are used for creating triangular faced meshes for extracting the constructional features, which are further interpolated to form a CAD Model. Any inevitable error during the RE-Process will significantly affect the creation of the virtual models. Even the most advanced setups, which are much costlier, have 1 to 3% error possibilities.

The error due to the presence of complex contour shapes will affect the quality of the image generated during the Non-Contact type of scanning process that results in issues in data point extraction. The errors in these cloud point data interpretation will have a multifold effect on the outcome of the RE-Process. Another prominent issue is the image stitching algorithm which results in misplacement of connections and this causing error during the standard cell extraction. Although some errors can be compensated through various techniques such as plausibility test, data refinement, etc., still error can’t be eliminated completely due to the complexity in each step. The presence of errors in the RE-Process may result in distorted or partially created virtual models. To correct and re-design the actual CAD model through these RE generated models will take a considerable amount of time and cost.

Machine learning implementation to resolve such issues can play a vital role; it not only reduces the cost but also minimizes the time to correct the virtual model with ease. The missing facets and distorted meshes can be recognized through similar structural features of Machine learning by using the K-Nearest-Neighbor (KNN)
algorithm. The structural features are extracted by using different graph algorithms, resulting in creating a data set for each category. These data sets are then used for training the machine learning algorithm which can predict the actual shape obtained through the distorted CAD model.

4.5 Artificial Intelligence in Design Simulation

Artificial Intelligence is rapidly encompassing several domains of mechanical engineering, and one of the most potential field of application is “Design Simulation”. Altair’s Chief Data Scientist and Senior Vice President, Dr Mamdouh Refaat, has brilliantly highlighted the importance of Artificial Intelligence and Machine Learning in Simulation in his latest article. For a better understanding of the concept, relevant materials from his article are presented below.

How AI and ML Crashed the Simulation Party

4.5.1 The Triumph of simulation

The last three decades have experienced rapid growth in the use of simulation tools in product design. Integrated with computer-aided design (CAD) tools, simulation software is now a fundamental ingredient in the practice of computer-aided engineering (CAE). During this journey, the finite element method (FEM) emerged as the main tool for simulation in many areas of product design such as structural integrity, computational fluid dynamics (CFD), electromagnetic analysis, and structural topology optimization. Because of the success of FEM addressing many design challenges, the complexity of the problems presented in FE simulation has grown rapidly.

FEM has become the top physics-based simulation technique and the number of elements involved in a FE simulation have increased by a factor of ten every decade. The typical design problems that were being simulated in the 1990s used a few thousand elements and these problems are now solved using several million elements. This allows a more authentic representation of real-life problems. As a
result of the increased problem size, the computing resources needed for FE simulation has grown dramatically and represented a non-trivial cost element in the design process.

While the FE simulation world was integrating and solving a wide range of problems, jobs started to become larger and larger. At the same time, artificial intelligence (AI) and machine learning (ML) have been advancing and inventing new methods that address the complexity of the same design problems. Recent advances in deep learning and the implementation of these methods using specially designed platforms running on GPU-based clusters are allowing ML models to shortcut the simulation process by summarizing the results of simulations. In doing so, the ML model serves as a repository of the wisdom gained from multiple simulation runs. The clear benefit of using ML is the reduction of a number of simulation runs during the design of a new, but similar, product.

4.5.2 From credit cards to product design

ML models have been applied on a large scale in other industries and most notably, the credit industry. In fact, ML models have been in production in the financial credit industry for more than three decades. The only difference is that they were not labelled as ML models, they were called “credit scoring” models. The most common format of these models is the “standard credit scorecard”. Scorecards are ML models that are fitted (or trained) on historical data of customers who were awarded some form of credit, such as a credit card or loan. The training data, which contains the default behavior of the customers, is typically extracted from the application and transaction history of these customers. The scorecard models are fitted to predict the default behavior of the customers in terms of some variables representing the characteristics of customers and their transaction patterns. These models are then used to score new applicants to determine their creditworthiness.

The same process is currently being applied to the simulation-based product design process. In this case, the physics-based simulation generates the data that describes the behavior of the different product designs under operating conditions (loading of different forms including mechanical, electromagnetic, fluid flow, and
more). This data is then used to train ML models to capture the behavior of the different product designs under loading conditions. For example, one could simulate the crash behavior of a car door with different designs and loading configurations and generate data that represents the crashworthiness of different door designs with materials. Then an ML model is fit on this data, to capture this crash behavior. When testing a new design the design is scored using the ML model rather than starting with FE simulation. The ML model scores may reveal that the proposed design does not meet the design criteria. Before conducting a time consuming and expensive simulation, ML models are used to validate the proposed design.

The above methodology is being used by several clients of Altair and is showing a dramatic reduction in the needed number of physics-based simulation runs reaching 90 percent in some cases.

4.5.3 How does ML work with simulation?

The Fig. 4.5 shown below summarizes how ML works with simulation data. We start with a set of proposed designs for the product in mind. We describe these designs in terms of some parameters or variables that represent the geometry, material, and the loadings of each design. The number of these designs should be sufficiently large to cover the range of possible designs we expect to have. We then perform physics-based simulation on these designs and calculate the targeted performance criteria. In the case shown in Fig. 4.5, we use some failure criterion to determine whether the product would sustain the applied operational loads.
The data from different simulation runs are then organized in a dataset as in above Fig. 4.5. This data is used to train (or simply fit) a ML model. In this case, because the output of the ML model is a binary outcome (failure: Y/N), the ML model will be fitted to predict the probability of failure. The inputs to this model are the various variables that can be used to characterize the product design in each FE simulation run. These variables characterize the design geometry, material properties, loading conditions, and any special feature that differentiates between the different designs. We will denote these variables as $x_1 – x_n$ in Fig. 4.5. When there is a new proposed design of the product, we will characterize that design with these variables. The data set that contains the parameters $x_1 – x_n$, as well as the output of the simulation in each case is known as the training data. This is the dataset that will be used to train the ML model.

There is a large variety of ML algorithms that could be used to fit such training data for different purposes. These algorithms vary according to the nature of the task at hand. The task in the example of Fig. 4.5 is a classification task. This means that we aim at finding a model (an equation, or rule) that will classify new designs using some failure criteria by predicting the likelihood (probability) of failure. One of the most powerful family of such algorithms for training ML models is deep learning. This has been successfully applied to the problem of part design from results of physics-based simulation. There are several powerful open source libraries as well as commercial
software that offer a variety of ML algorithms, including deep learning, that can be used in ML models for product design as described above.

After training the model, it can be used to calculate the likelihood of failure of products of similar designs within the same range of characteristics as those used in the simulation runs. This second phase of using ML is depicted in Fig. 4.6.

When encountering one or more newly proposed designs of the product to evaluate, it is not done by running FE simulations, but by scoring these designs using the trained machine learning model. With this, the values of the characteristics $x_1 - x_n$ are extracted for the new designs and are used as inputs for the trained ML model. This model will predict the probability of failure for each of these designs. This scoring process is depicted in Fig 4.6. The process of scoring ML models is very fast and computationally inexpensive compared to FE simulations. Typically scoring an ML model of real complex products is expected to take a few seconds or less.

### 4.5.4 What about accuracy?

One of the important factors in the success of FEM as the prime physics-based simulation tool is high accuracy. Any candidate approach that would be introduced to the world of simulation will have to produce comparable accuracy to that offered by the FEM.

It has been demonstrated that ML-based models can produce accurate results as those obtained by physics-based simulation. For example, the accuracy of using ML modeling to reproduce the FE results representing the mechanical behavior of the
human heart showed how ML can act as an accurate surrogate to FEM to calculate stress distribution.

### 4.5.5 The road ahead and challenges

Using ML models to shorten the simulation cycle has been successful in the design cycle of many products. Although we expect to see rapid growth in the application of this methodology in the next few years, there are some challenges that need to be addressed before ML becomes a standard tool for the design engineer.

The first challenge is the current engineering design practice. Most engineers create a set of designs and use FE simulation to test whether these designs meet the acceptance criteria. When a design fails and proves to be sub-optimal, the data from simulation run is deleted as a failed attempt. In order to train ML models, the simulation data of these failed designs is needed. Engineers will need to update their data management practices and create a repository of simulation runs, both good and failed designs, in order to use this data in ML models.

The second challenge is to effectively use a data repository for the simulation data. Engineering design groups will need to invest in data management platform in order to organize, clean, and make data available from simulation runs. This platform is different from the current product life-time cycle software. The focus of the new platform is the organization of the data for the purpose of developing ML models. The challenge is not a technical one, but the fact that engineering design departments will have to adopt a data-driven approach to design, rather than a simulation-driven approach. The FE simulation changes from being a tool in the design cycle to a tool of data generation. Transforming from a platform of managing data to a platform in which the product design lives and functions.

The third challenge is the need to train design engineers in the art of data science to enable them to implement best practices in ML. Although it is expected that many steps in the ML model developing process will be automated over the next few years (auto-ML), the design engineer still needs to have some fundamental knowledge about ML models and how to properly implement them. It is expected that
most of the FE simulation vendors and engineering departments in universities will launch their own ML training courses for design engineers to fill this skill gap.

The last challenge is the influence of the Internet of Things (IoT) and digital twin on product design and the overlap with ML. Current practices in product design are already taking advances in IoT and digital twin to adapt designs to the new world of connected products. Sensors generating data about the product performance during operation will also need to be integrated into the new data management platform to be used in training ML models for the design of new products, and the performance optimization of existing products in operations. In fact, the advances in IoT and digital twin are removing the boundaries between the data available at design time and what is available during operating the product. The boundaries are becoming fuzzy, and that in fact adds another opportunity for ML to use operational data to further tune the ML models.

In numerical analysis courses, it is taught that numerical problems could be solved quickly using a limited number of iterations, but in most cases, it would lead to lower accuracy. In order to achieve higher accuracy, more iterations or higher-order terms would need to be used to result in more complex computations that need more resources.

Today with the recent advances in ML, an exception to the above rule has been found. FE simulation allows the modelling of the most complex structures, while ML can help optimize the use of simulation resources to make product designs more efficient without sacrificing accuracy.
5 Programming languages in AI

A programming language is a way of representing an idea and communicating with the machines through algorithms. The selection of a language for programming a machine is mainly dependent on the affinity to the language a programmer is acquainted with. Another crucial factor in language selection is its capability of being scalable, fast and fault-tolerant.

There has been some misconception among those who are relatively new to the AI field that artificial intelligence can only be done through Python, which is completely incorrect. But one thing that can be inferred from this misconception is that Python is the most commonly used language for AI. Apart from Python, many other programming languages can be used for AI. It is imperative to know the other options that are available for programming in AI for the creation of mathematical models and which language is the best among all.

Artificial Intelligence is considered to be a predictive tool in which statistical and mathematical models are created through various computer programming languages – that’s a general perception which is not “completely true”. It is vital to understand that Artificial Intelligence is not only about writing programs or codes, there are several other aspects that play a significant role in AI. The other aspects of AI are – recognizing and understanding the problem, determining modes of the solution and converting them into an algorithm which is then finally used for writing the programs/codes. That’s where the need for a programming language comes into the picture.

For writing programs of the mathematical model, a user requires the help of a programming language and various tools available in its library; based on the variety of these tools, its flexibility and features – the programming languages can be ranked by the users and hence one can choose the preferred language for a specific type of work. According to the Kaggle worldwide survey in October 2018, the most widely
used programming language by the coders in the field of Data Science, Artificial Intelligence and Machine learning is – Python. Fig. 5.1 show the Pie-chart of the summarized report of the Kaggle survey and it can be seen that Python, SQL and R – Programming are the top 3 choices of the users worldwide.

Python is the most preferred and effective language for application in the field of artificial intelligence. The main reason behind this is, that the syntaxes in python are very simple and easy to learn by the majority. Apart from Python, there are a variety of programming languages which are being used by users in the field of AI. Some of the other significant names in AI programming languages are Java, MATLAB, R-programming, Lisp, Prolog, etc. To get a holistic idea, some of these languages are discussed individually in brief; this will also help in language selection based on a specific requirement.
5.1 Python

Python is a general-purpose high-level programming language that is most commonly used to develop different types of software, 3D graphics, websites, games apps etc. Python programming language was created by “Guido Van Rossum” in the early 1990s and the name “Python” inducted from one of the most popular shows “Monty Python's Flying Circus”. As stated earlier, due to the easy syntax system, Python is the most widely used programming language among all. Python programs are easily portable among different software platforms, user friendly and easy to maintain the language. Another important factor that makes Python the most widely appreciated programming language is that it also supports object-oriented, functional oriented and procedure-oriented programming styles and requires considerably lesser development time compared to others such as Java, C++, etc.

There are a few shortcomings in Python language – it is considered as a weak programming language for mobile computing and consumes high memory due to the data flexibility. Although Python is considered to be slow in the process when compared with C++, it’s still considered to have many advantages over the latter. Apart from all these factors, one feature which keeps Python in the top of the league for the field of Artificial Intelligence is the presence of various inbuilt libraries which have predefined functions for implementing Machine learning and Deep learning algorithms. These libraries reduce the overall burden of programmers who intend to work in the field of rapidly growing AI technologies. And that’s why Python is considered to be the best among all for AI.

5.2 R-programming

R-programming language was crafted by Ross Ihaka and Robert Gentleman in the year 1993. It is used extensively in the area of statistic, visualization, and machine learning. It can easily produce well-designed publication-quality plots, including mathematical symbol and formula, wherever needed. R-programming language is some-what complex when compared to python but does have several advantages. A simplistic comparison between R-program and Python can be understood through the following analogy.
<table>
<thead>
<tr>
<th>Python</th>
<th>R-Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mostly for general purpose</td>
<td>Widely used in the field of statistics</td>
</tr>
<tr>
<td>User friendly and flexible</td>
<td>Has lesser flexibility</td>
</tr>
<tr>
<td>Required greater setup</td>
<td>Easy to use in well-defined problem</td>
</tr>
<tr>
<td>Higher processing speed</td>
<td>Lower processing speed</td>
</tr>
</tbody>
</table>

Similar to Python, it also has various predefined library functions that supports statistics, data science, AI, machine learning and natural language processing. Owing to these features, R-programming is also a widely used language in the field of Artificial Intelligence.

### 5.3 Java

Java is a general-purpose programming language which is object-oriented, platform-independent, reliable, fast and secure. It was released in 1995 by Sun Microsystems which enabled a programmer to write computer instructions through English language-based commands instead of numeric codes. Java is most popular for web scripting, however, machine learning libraries and framework do support JavaScript. The major issues with Java are its longer processing time, high memory consumption, non-supportive to low-level programming and no control of garbage collection. On the other side, Python easily overcomes these issues as well as imparting flexibility in the programming.

### 5.4 C++

One of the oldest general-purpose and most popular programming is C++, which was released in 1985. It is an intermediate level language which has generic programming features, works on object-oriented functions and easily runs on numerous platforms such as Windows, Mac, Linux, etc. C++ has a collection of predefined classes where data types can be instantiated multiple times. It is also capable of accommodating member functions for implementing a specific function.

The prominent issues with C++ are that it has no security and high-level programming is much too complex. Another main issue, when used with web applications, is the debugging problem. These prime concerns make C++ less
preferred for machine learning.

5.5 MATLAB

MATLAB is a high-performance language that was developed by MathWorks and released in 1984. MATLAB stands for Matrix Laboratory and is capable of performing numerous mathematical operations, data & function plotting, algorithm implementation and developing user interface as well as interfacing with programs written in other languages.

5.6 LISP

LISP is the 2nd oldest language used worldwide after Fortran and was released in the year 1958. Over the period of time, Lisp has evolved into a strong dynamic programming language and many consider it has good utility in Artificial Intelligence due to the liberty developers get through it. The main benefit which Lisp offers in AI is its flexibility in experimentation and prototyping. It has a microsystem which supports the implementation of various levels of Artificial Intelligence. It is more efficient in problem-solving as it adapts to the requirement of the solution which programmer is coding for; making it suitable for logical projects and Machine Learning. The prime concern with Lisp in its usability for Artificial Intelligence is that it requires new software and hardware configuration for accommodating its applicability.

5.7 PROLOG

Like Lisp; Prolog is also among the oldest logical programming languages which is also suitable for programming in Artificial Intelligence. It has a flexible framework mechanism for developers which is a rule-based and declarative language. It does support pattern matching, backtracking automation, tree-based data structuring which are key factors in AI programming. Apart from its usage in AI, Prolog can also be used in medical technology. It frequently releases modules and allows simultaneous database creation which makes it efficient for fast prototyping in Artificial Intelligence programs. Despite many advantages with Prolog, its only main issue is the standardization of some of its features which creates trouble for the developers to implement it effectively.
So to summarize - Python is easy, Java is verbose, Clojure is hard to read, Scala is functional, Prolog is logical, R is not in the grasp of everyone. All the software mentioned above requires skills to write the programs for algorithmics that can train the data and perform predictive analysis. However, there are many other software modules available in the form of open-source or licensed that are dedicated to the Artificial Intelligence and Machine Learning application. Some of the most important software modules are mentioned in Table 5. The list is not exhaustive and only comprises widely used software modules in the domain of AI and ML.

Table 5.1. Software Modules details for dedicated AI and ML application

<table>
<thead>
<tr>
<th>Sr No.</th>
<th>Software Module</th>
<th>Detail</th>
<th>Provider</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.</td>
<td>Amazon Machine Learning</td>
<td>Amazon ML provides tools and wizards with visuals that guides through the creation of machine learning models without requiring to learn the complex algorithms.</td>
<td>Amazon</td>
<td><a href="https://aws.amazon.com/machine-learning/">https://aws.amazon.com/machine-learning/</a></td>
</tr>
<tr>
<td>3.</td>
<td>Keras</td>
<td>Keras is a neural-network Python library which is an open-source and capable of running on top of TensorFlow.</td>
<td>Python library</td>
<td><a href="https://keras.io/">https://keras.io/</a></td>
</tr>
<tr>
<td>5.</td>
<td>Mindsphere</td>
<td>MindSphere ML provides learning techniques for creating models to optimize product.</td>
<td>Siemens</td>
<td><a href="https://sitemens.mindsphere.io/en">https://sitemens.mindsphere.io/en</a></td>
</tr>
<tr>
<td></td>
<td>Software</td>
<td>Description</td>
<td>Company</td>
<td>Website</td>
</tr>
<tr>
<td>---</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>6.</td>
<td>Movidius</td>
<td>Movidius is a powerful deep learning framework developed to solve some of the most difficult challenges of modern problems.</td>
<td>Intel</td>
<td><a href="https://www.movidius.com/">https://www.movidius.com/</a></td>
</tr>
<tr>
<td>7.</td>
<td>Veles</td>
<td>It is a distributed platform software solution which is mostly used for application development in deep learning.</td>
<td>Samsung</td>
<td><a href="https://velesnet.ml/">https://velesnet.ml/</a></td>
</tr>
<tr>
<td>10.</td>
<td>Caffe</td>
<td>Caffe is a deep learning framework developed by Berkeley AI Research (BAIR). Caffe is considered perfect for research experiments which has lightning fast speed of processing over 60M images per day - almost 1ms/image speed.</td>
<td>Berkeley AI Research</td>
<td><a href="https://caffe.berkeleyvision.org/">https://caffe.berkeleyvision.org/</a></td>
</tr>
</tbody>
</table>

In conclusion, it is safe to say that Artificial Intelligence can be implemented for a variety of applications by using system software or software modules. For
complete control of data training and predicting algorithms, software such as Python, R-programming, LISP, Java, PROLOG etc are widely used, however, they require skilled programmers, experts and computer scientists for proper application. On the other hand, there are several dedicated software modules such as Altair’s Knowledge Works, Azure, Amazon AWS, Keras, Mindsphere, etc., used for AI and ML application that does not require high skills to create predictive algorithms. These modules have predefined set of instructions which can be easily managed by a novice. Based on the need and the skills, software selection can be done.
6 The Machine learning Process

Before we begin to understand the concept of Machine learning, it is very crucial to understand why there is a need for Machine Learning technology. One of the most well-known facts is that there are specific tasks where computers are much better than humans. For example, if any person is asked to perform division operation between two big numbers, which are in hundreds of thousands, it is certain that this process will take a lot of time, whereas computers can do this in less than a millisecond. Similarly, if any human needs to search through millions of records, then that search operation is challenging and might not even be accomplished. On the contrary, computer machines can do the same process in a matter of nanoseconds.

On the other side, there are some tasks where humans outperform machines. The best example of such a case is driving a car where a robot has no match with human skills. Although big names such as Google and Tesla are working to develop such a driverless car, they are still relatively new and prone to many issues. Another example is understanding the language and conversation for performing any task. Machines are still behind humans in these aspects.

Therefore, the big question is, “what if a machine can start working like humans”? It will be a considerable achievement and may reshape the future. What can only be possible is that machines start learning similar to humans. This is what Machine Learning process is trying to achieve. The efforts are being made to create a process where machines can make decisions on their own and think like humans to make machines better at things where traditionally humans were outperforming them.

The next big question is how to make a machine learn things like humans do. A machine can think like humans when they can be trained like humans. This training can be of mechanical or biological type. The mechanical training refers to the
instruction based learning process where directions are provided to perform specific tasks. Then based on these instructional learning the next set of task is performed without providing the directions and the similar outcome is expected. This way of learning refers to the Machine Learning process of Artificial Intelligence.

On the other hand, biological learnings are referred to the system where the concept of human intelligence is utilized for the learning process. Earlier, we had briefly discussed how the human brain works, which consists of several billions of tiny neurons. Whenever the brain thinks or makes a decision, a signal, which is a chemical signal, is generated and these tiny neurons light up. These input signals are then processed to get the output signal based on previous learnings and also what signals other neurons have given. This type of learning process is categorized as Deep Learning; to understand it further, refer Chapter 9.

6.1 Machine Learning

The earliest known conventional definition of a machine learning states that “A computer program is said to learn from experience E with respect to some class of task T and performance measure P if its performance at a task in T, as measured by P, improves with experience E.” This definition was quoted by the well-known researcher ‘Arthur Samuel” in 1959, but it does not give any clear picture of what machine learning is. So the simplified understanding of it is – Machine learning is a method to predict or classify the outcome of the future data through a computer model by recognizing a pattern from the past data set which gradually improves with experience. Machine learning involves creation of predictive mathematical models by using statistical tools and a programming language which can help a machine train on its data and predict specific outcomes based on this training. This machine learning process can be easily understood with the help of a flow diagram as shown in Fig. 6.1.
Machine learning is based on training data by using mathematical models. The higher be the discrete nature of data, the better will be the machine learning process. One very important word of caution here is that “Machine Learning process is majorly presumed as data training process” but this is not the only important factor in ML, there are several other aspects also which play a significant role for creating efficient Machine Learning process. It implies that for solving a real problem and making a good prediction, the data set is not the only factor which has a most influential role; understanding the type of data set, checks for missing gaps in data, training and validating the data set and selecting the best suitable algorithm are also of much importance as that of data training. It’s very imperative to understand the complete process of Machine learning for maximizing its utility.

6.2 Steps in Machine Learning

The machine learning process is aimed at creating a model that can recognize a pattern from the data set and predicting future trends. To get the desired outcome Machine Learning process comprises of 7 major steps – Data Acquisition, Curation, Exploration, Model Creation, Model training, Evaluation, and Prediction, as shown in Fig. 6.2.
Data Acquisition is a technique of gathering information from a variety of sources in a pre-defined form. The data can be in numeric values (age, annual turnover, temperature, loan, income, etc.) as well as in textual form (color category, gender, level of study, emotional feeling, etc.) These data are the records of past events which is captured for analysis purpose to determine the recurring trend. The data collection method is based on the requirement and the type of its source. For example - a machine functioning data can be collected through sensors, user details can be gathered through survey and feedback form, company performance data can be obtained through its balance sheet, etc. These data set are collected and stored systematically and called as Training-Testing Data in machine learning.

Data Curation: the efficacy of the machine learning process is primarily dependent on data quality which is obtained through various data acquisition techniques. Many a time these collected data are just in bulk form with scattered details along with several issues such as voids or redundant data. These issues arise due to the volume of data growing exponentially that are majorly of heterogeneous in nature and require thorough check before their immediate usage. The process of creating the correct set of data with necessary details in the required condition is known as data curation. It comprises 3 steps – Data Analysis, Data Cleaning, and Data
Transformation. In a way, Data Curation can be seen as “data management” which focuses on improving the quality of data for fulfilling its two main aims – quality compliance and data retrieval for future research. Data curation creates trustworthy data that can be further optimally used for a variety of applications.

Data exploration is another very important task in machine learning. The exploration deals with the analysis of the data and computing the mean standard deviation, median and mode for the curated data. Further, distribution plots such as histograms, box plots, bar plots are created for correlating among features or correlation between features and the variable that are required to predict the nature of the existing data. Thereafter, it becomes very important to determine which features are most relevant for the specific problem in a real application. The data set with a large number of features are further scrutinized by using different techniques, such as Principal Component Analysis (PCA), for feature selection and dimensionality reduction. The data exploration helps in understanding the data set which can be helpful in selecting or creating the machine learning models.

Model selection or creation is considered as one of the most important steps in machine learning which deals with selection or creating the algorithms which can process the curated data and formulate the mathematical equation which is the best fit for the data set. It is a method of selecting that statistical tool that can define the equation that satisfies almost all the data with the least error.

To better understand the modelling in machine learning, consider a random set of data that can be plotted as shown in Fig 6.3(a). This randomized data can be of company annual product sales or product demand. If a curve is to be made on this data set in such a way that it can be the best fit with all the data points, which satisfies all the data set, a linear curve will not do the justification, Fig 6.3(b), therefore, a high degree of non-linear curve is needed for fitting the data points, i.e. Fig 6.3 (c) & (d). Among all, Fig 6.3(d) shows the best fit curve to the data set.

If a mathematical equation can be defined for such a highly non-linear curve with the given set of variables such that all the data set can be satisfied perfectly with minimum error, then that equation can be called as the model of the system. This model can be used for predicting the future trend and its validity can be checked by
the data set itself. This is what “Modeling” in machine learning refers to. It is a method of formulating mathematical equations that can best define the dataset and it can be used for predicting future outcomes. Mathematically, modelling is defined as determining the function (f) for the variable \((x_i, y_i)\) that fits the data well; i.e. \(y_i = f(x_i)\).

Data exploration holds the key in the determination of which statistical tool or algorithm is to be selected that is best suitable for the given dataset, ease the process and improves the efficiency of modelling. In a way, modelling tasks in machine learning can be seen as finding the right balance between approximation and estimation of errors.

Model Selection refers to the selection of pre-defined standard algorithms such as Linear regression, Random forest, K-means, Naive Bayes, etc. that can define the dataset perfectly and can predict the desired outcome accurately. The selection of the algorithm depends on the type of machine learning problem such as supervised, unsupervised, or reinforced learning; which will be discussed later. A detailed discussion of algorithms is presented in chapter 7. On the other hand, Model creation is the method of formulating mathematical equations for a specific type of task and

![Modeling in Machine learning](image-url)
predicting its outcome.

**Model Training** is a process of making machine learning algorithmic models learn from the “training dataset”. It is to be noted that the Machine Learning process does not consider all data for training the model; in general, the complete dataset is bifurcated into two parts – one for training and other for testing. The bifurcation can be done manually or through the software, but the aim should be of selecting the randomized set of data for training such that errors in prediction due to specific pre-patterning of data can be avoided. The division of the dataset is generally in the ratio of 80:20, where 80% is considered as training data and 20% considered for testing purpose to validate the authenticity of the model. Other ratios such as 70:30 and 90:10 are also used by many and it completely depends on the domain, dataset particulars and amount of data available for the machine learning.

The bulk data part, i.e. training data, must contain the target attribute, which is the expected outcome. The model is then trained on these data to determine a pattern or correlation between input variables and target attributes for predicting the required outcome. Thereafter, these predictions are verified for the non-training data to check the correctness of the model. The type of algorithm should be selected judiciously such that it can capture the data pattern accurately.

**Model Evaluation** deals with checking the performance of the trained model outcome. The untrained testing dataset (10%- 30% of the total data) with a known outcome is used in the trained model to predict the outcome. The predicted outcome is then compared with the known outcome and error % is determined to evaluate model performance. If the error is more than acceptable, then further finetuning is done in model hyperparameters such as steps for training the model, its rate of learning, distribution, initialization value, etc. These hyperparameters are also controlled to check for the overfitting and underfitting issues.

“Overfitting” issue arises due to the usage of a higher number of parameters for defining the data which fails to justify itself. It is an outcome of the data analysis that is too close with the data set that may fail to fit additional data. On the other hand, “Underfitting” issue is related to the model where data parameters are missing due to which the model can’t capture the essence of data. Underfitting could be well
explained by an example where we try to fit a linear model for non-linear data.

**Prediction** relates to the process of forecasting the likelihood of the outcome through the trained algorithm on the historical dataset. It is a step of getting answers by utilizing the dataset and model. Prediction is the final stage of the machine learning which refers to the deployment of the trained model for getting the desired outcome.

### 6.3 Example

To substantiate the understanding of machine learning steps, let us consider a simple hypothetical example and use the machine learning concept through Python Programming to predict the outcome. A company designs and fabricates a “Hook” which is commonly used for hanging heavy loads under the static condition for several years in the industrial setup. Although the hook is designed for working under a normal room temperature of 20°, it is being used under different temperatures at different geographical locations. Due to this, the failure of hooks increased which affected the company’s reputation. To save its reputation, the company wanted to know, how long the hook can survive the load under different temperatures. Thus, the company defined an objective of “Predicting the Hook’s life based on its operating temperatures and loading conditions. To have a better understanding of the situation, the company contacted the industries that use their product and asked their mode of usage of the hook. The machine learning steps for the Hook’s life prediction are discussed hereafter.
Step 1: Data Acquisition

The data which was obtained from the industry based on the survey of the usage of the Hook in their actual application was combined and a table was formulated; as shown in Table 3.

<table>
<thead>
<tr>
<th>Sr No.</th>
<th>Working Load</th>
<th>Working Temperature</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5300</td>
<td>35</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>4900</td>
<td>24</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>2350</td>
<td>40</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4900</td>
<td>21</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>2770</td>
<td>28</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>2560</td>
<td>34</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>3500</td>
<td>45</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>2800</td>
<td>42</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>1900</td>
<td>21</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>4450</td>
<td>26</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>3800</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>3180</td>
<td>24</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>5900</td>
<td>28</td>
<td>3</td>
</tr>
<tr>
<td>14</td>
<td>5200</td>
<td>32</td>
<td>4</td>
</tr>
<tr>
<td>15</td>
<td>2350</td>
<td>24</td>
<td>6</td>
</tr>
<tr>
<td>16</td>
<td>3400</td>
<td>36</td>
<td>4</td>
</tr>
<tr>
<td>17</td>
<td>2300</td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>18</td>
<td>4210</td>
<td>27</td>
<td>3</td>
</tr>
<tr>
<td>19</td>
<td>4320</td>
<td>38</td>
<td>3</td>
</tr>
<tr>
<td>20</td>
<td>1850</td>
<td>38</td>
<td>6</td>
</tr>
<tr>
<td>21</td>
<td>2950</td>
<td>26</td>
<td>6</td>
</tr>
</tbody>
</table>

Step 2: Data Curation

The obtained dataset was checked carefully to see whether any data is missing or any odd data which is completely different from the list and may cause trouble in the outcome of the prediction is present. Here, the 11th series in the data seems to be odd as it works under much lower temperature compared to others. This may possess some issue in capturing the true nature of the data, therefore for the
betterment of the outcome, this data will be eliminated. This is data curation, where we try to remove issues within data.

**Step 3: Data Exploration**

In this stage, to know the nature of data, the plotting is done between the variables and the outcome. For the simplification of understanding and usage, we will be using Python programming language through a free online python notebook “Jupyter” henceforth and the steps involved in it are mentioned below.

### JUPYTER ONLINE NOTEBOOK

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>To enable the Python program to read the data set in excel, import “Pandas”.</td>
</tr>
</tbody>
</table>
|      | ```python
import pandas as pd
df = pd.read_csv("Hook-Life_determination.csv")
df.head()
``` |
| 2.   | To create plots for data exploration, import Matplotlib |
|      | ```python
import matplotlib.pyplot as plt
%matplotlib inline
``` |
| 3.   | Creating plots between input variables (load & temperature) and output signal (Age) |
|      | ```python
plt.scatter(df['Working Temperature'],df['Age'])
plt.scatter(df['Working Load'],df['Age'])
``` |
| 4.   | For input variables to x-axis and output to y-axis |
|      | ```python
X = df[['Working Load','Working Temperature']]
y = df['Age']
``` |
| 5.   | Bifurcate the dataset into training data & testing data in 70:30 ratio. |
|      | ```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3)
``` |
| 6.   | To check the data bifercation |
|      | ```python
X_train; y_train; X_test; y_test
``` |
| 7.   | To import Linear Regression Algorithm for training dataset |
8. To check the predicted outcome & compare with existing outcome

```python
clf.predict(X_test); X_test
```

9. To check the performance of predicted and actual outcome

```python
clf.score(X_test, y_test)
```

**Outcomes of the Machine Learning Program**

**Plotting for Data Exploration (stage 3)**

**Step 4: Model Selection**

The data exploration stage shows that there is no specific trend present in the available data. Although, the data seems to be in a random pattern (as shown in the outcome of stage 3), for the sake of simplicity of understanding, we are considering the algorithm “Linear Regression” to predict the outcome. This selection of algorithm is incorrect because the trend does not seem to be linear, but this wrong selection will help to understand the machine learning concepts better.

**Step 5: Model Training**

For training the model, we will define the training data and testing data from the available dataset. This bifurcation of data can be done manually or with the help of the system, but the consideration should be given to the selection of randomized data. Therefore, to get a randomized data set for training and testing, the help of the
program was taken. The input variables are defined for x-axis whereas y-axis is for output variables.

The outcome of random bifurcation through programming (stage 6)

<table>
<thead>
<tr>
<th>Working Load</th>
<th>Working Temperature</th>
<th>X_train</th>
<th>y_train</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>5900</td>
<td>28</td>
<td>11</td>
</tr>
<tr>
<td>13</td>
<td>2350</td>
<td>24</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>2560</td>
<td>34</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>1900</td>
<td>21</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>3180</td>
<td>24</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>5200</td>
<td>32</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>2800</td>
<td>42</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>4900</td>
<td>21</td>
<td>3</td>
</tr>
<tr>
<td>16</td>
<td>4210</td>
<td>27</td>
<td>16</td>
</tr>
<tr>
<td>0</td>
<td>5300</td>
<td>35</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>4900</td>
<td>24</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2770</td>
<td>28</td>
<td>4</td>
</tr>
<tr>
<td>19</td>
<td>2950</td>
<td>26</td>
<td>19</td>
</tr>
<tr>
<td>14</td>
<td>3400</td>
<td>36</td>
<td>14</td>
</tr>
</tbody>
</table>
Step 6: Model Evaluation: Once the training of the selected dataset was done through the linear regression method, it is then required to predict the outcome for the input data and compare to the known outcome and evaluate the performance of the machine learning process. Stage 8 & 9 of the machine learning process through the Jupyter Notebook shows this prediction and evaluation.

Prediction Outcome

```
array([4.48636161, 5.33793764, 3.3166905, 3.56967612, 4.97702107, 4.73023623])
```

<table>
<thead>
<tr>
<th>Testing Sample No</th>
<th>Predicted Outcome</th>
<th>Actual Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4.48636161</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>5.33793764</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>3.3166905</td>
<td>3</td>
</tr>
<tr>
<td>17</td>
<td>3.56967612</td>
<td>3</td>
</tr>
<tr>
<td>18</td>
<td>4.97702107</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>4.73023623</td>
<td>4</td>
</tr>
</tbody>
</table>
The evaluation of the model shows that there is variation in actual outcome and predicted outcome. This is because Linear Regression is suitable for the linear natured data, but in the data exploration plot, we saw the data was of random non-linear type.

\[ 0.4589965879961132 \]

Therefore there is a difference in predicted and actual outcomes. The model efficiency can be determined by stage 9 and its outcome is mentioned below.

This 0.45899658 value represents that the machine learning model which was programmed above has a 45.89% correctness possibility for the predicted outcome. To increase the prediction accuracy, appropriate algorithm should be selected that can define the pattern of data in better way.
7 Types of Machine learning & algorithms

Machine learning deals with the method of forecasting the outcome based on its learning from the available dataset. The learning refers to the training of the algorithmic model by using previous data for determining a pattern in it. In general, there are three types of learning in ML

1. Supervised Learning
2. Unsupervised Learning
3. Reinforced Learning

7.1 Supervised Learning

Supervised Learning is a method of training the machine learning model with the help of an input vector and its supervisory signals. The input vector refers to the dataset that acts as the input to the system whereas the supervisory signal is related to the output data. This learning deals with training a model with a dataset that is labelled, i.e. whose outcome is already known. Supervised learning can be considered as training a child with images of specific categories of fruits that are labelled accordingly. The child sees the image and learns its detail from labels and further learns to distinguish it from other images; shown in Fig. 7.1. This input-output relation-based training is known as Supervised Learning.

In supervised learning, the model analyzes the training data and determines a function that can map the relation between input and output data which can be used for predicting the new unseen data. The
creation of the mapping function is an iterative process where the machine learning model analyzes the data, makes a prediction, compares with actual output to determine the error %, and repeats the process till the error is minimized to an acceptable level.

Some of the commonly used algorithms in supervised learning are Linear regression, Logistic regression, Linear discrimination analysis, Decision trees, Bayesian logic, Support vector machines (SVM), Random forest, etc. The main applications where supervised learning is used are – spam detection, handwriting recognition, speech recognition, computer vision, biometric attendance, weather predictions, etc.

**Algorithm behind Supervised Learning**

Consider a set of N training examples in the form \( \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \) such that \( x_i \) is the feature vector of the \( i \)th example and \( y_i \) is its label (i.e. a specific outcome). The supervised learning algorithm seeks to form a function “\( g: X \rightarrow Y \)”, where \( X \) refers to the input space, and \( Y \) is for the output space. This function \( g \) is an element of some space which is possibly a function \( G \), and commonly known as hypothesis space. The representation of \( g \) could be done by using the scoring function \( f: X \times Y \rightarrow R \) such that \( g \) is defined for returning \( y \) value that gives the highest score: \( g(x) = \arg \max f(x, y) \). Assume \( F \) is the scoring function that denotes
the space. Although the function $G$ and $F$ can be of any space, in general, learning algorithms are based on probabilistic models where $g$ forms a conditional probability model $g(x) = P(y \mid x)$, or $f$ forms a joint probability model $f(x, y) = P(x, y)$.

For selecting $f$ and $g$, two approaches can be considered, they are – structural risk minimization which uses a penalty function to control the variance tradeoff and empirical risk minimization which uses the best fit function of training data. In both the approaches, the training data set consists of independently distributed pair $(x_i, y_i)$, and to determine which function best fits with the training data, a loss function is defined as $L: X \rightarrow Y$. The risk $R(g)$ of the function can be calculated by

$$R(g) = \frac{1}{N} \sum_{i} L(y_i, g(x_i))$$

Those supervised learning models which consider the empirical risk minimization method, consider the minimization of $R(g)$ for constructing the optimal algorithm for determining “g”. On the other side, models that use structural risk minimization method consider incorporating regularization penalty $C(g)$ in the optimization.

$$J(g) = R_{emp}(g) + \lambda C(g)$$

Where $\lambda$ is the controlling parameter for the bias-variance tradeoff. $\lambda=0$ signifies high variance with low bias and when $\lambda$ is a large value, then bias is high.

### 7.2 Unsupervised Learning

*Unsupervised Learning* is a method of training the model where input vectors do not have any supervisory signals. This implies that for the given set of input data, there is no definite output value/label. All the dataset is fed into the model and the algorithm determines the properties or patterns in these data which is further used to learn the group or cluster formation; Fig. 7.2. The newly-organized clusters are the set of data with similar properties.

Unsupervised learning is a data-driven learning process where the outcome is dependent on the characteristics of data and considers the probability densities of
the given input. The common applications of unsupervised learning are: E-commerce websites for clustering the users based on buying habits and prompting the buying suggestions, YouTube & Netflix video recommendation systems, Astronomical applications for identification of heavenly bodies, and differentiating among stars, planets, asteroids, etc.

In Unsupervised learning, the selection of an algorithm depends on the type of job associated with it. Some of the widely used algorithms are: K-means, C-means, Hierarchical, Mixture Models, Gaussian Mixture, Hidden Markov Model, Principal Component Analysis (PCA), Linear Discrimination Analysis (LDA), etc.

![Algorithm behind Unsupervised Learning](image)

**Algorithm behind Unsupervised Learning**

To train the model for identifying the hidden parameters $\varphi$, which is known as characteristics of the input data $(x_i)$, the unsupervised learning algorithm takes in $x_i$ and outputs a trained set of parameters $\varphi$. The parameters are used for defining the feature functions $\alpha (x, \varphi)$ which then labels the dataset based on these feature functions. The machine learning algorithms in unsupervised learning predicts “y” from feature $\varphi$. The two widely used unsupervised learning methods for relating $\varphi$ with “y” are – Clustering Analysis and Principal Component.

Clustering Analysis deals with grouping of a set of data based on similarities among each other compared to those of other groups, Fig. 7.3. The simplest clustering algorithm is Gaussian mixture which considers that the data with K components are generated with a mixture model which is determined by
\[ P(x) = \sum \pi_i P_i(x; \phi_i) \]

where the group distribution \( P_i \) is parameterized by \( \phi \) with prior probabilities \( \pi_i \) sampled from each component.

7.3 Reinforcement Learning

Reinforcement Learning is a unique model training process where learning is done by doing mistakes. Compared to Supervised learning (labelled outcome) and Unsupervised Learning (non-labelled outcomes), reinforced learning deals with taking actions based on its environment by maximizing the total reward and minimizing the penalty. It is something similar to trial and error to get the right decision.

Reinforced learning deals with training the model to take a sequence of decisions by the agent considered to be in a complex environment and a balance is tried to achieve between the unknowns and already known details. In this method, the learner is known as an agent and everything outside of the agent is considered as the environment, Fig. 7.4. The agent interacts with the environment continuously by taking certain action steps and the environment responses to these actions by giving a new situation to the agent. The environment also provides a reward to the agent that is a step close to correct action and the agent tries to maximize this reward over
To have a better understanding of agent and environment relation in reinforcement learning consider a hypothetical situation where a man (agent) is dropped off in an isolated island (environment). Initially, agent will panic and will question its own survival possibilities. But gradually when he explores the island and understands the climate and conditions, he could identify the means of his survival. He will gradually learn from his actions and mistakes while living on this island to have better survival chances. He will take decisions to overcome the danger that possesses a threat to his survival. And finally, he will learn to live peacefully with his environment. This is what reinforcement learning follows, learn by observing and performing actions.

Reinforcement learning can be most efficiently used in controlling the traffic light system, robot’s decision making, web system configurations, and chemical industries for reaction optimizations. The common types of algorithm that are used in this learning method are Monte Carlo, Q Learning, Deep Q Network (DQN), Proximal Policy Optimization (PPO), SARSA, etc.

From these discussions, it can be inferred that the machine learning’s 3 categories, i.e. Supervised Learning, Unsupervised Learning, and Reinforcement learning are the methods of training the models on the data set. The type of learning
process will be dependent on the kind of available data set and the desired field of application. Fig. 7.5 shows some of the common applications of the aforementioned Machine Learning types.

All these 3 types of learning methodology use various algorithms to train the model and the selection of these algorithms plays a vital role in the efficacy of Machine Learning process. It is imperative to understand the types of popular algorithms available within the genus of Machine Learning.

### 7.4 Algoritms of Machine Learning

An algorithm is a finite sequential set of well-defined instructions to solve a class of problems or to perform a computation. In machine learning, algorithms are used for training the data for predictive analysis. There are various types of predefined algorithms that can be used to get the desired outcome, and its selection depends on the nature of the available data. Fig. 7.6 shows some of the commonly used algorithms in three different categories of machine learning.
In supervised learning, the Classification Algorithm is a method of segregating the data into a set of categories and identifying which category a new observation is associated with. Classification algorithm can be used when the desired outcome is labelled separately. The sorting of e-mails into different categories such as spam, junk or update is an example of a Classification Algorithm. It is very important to select the right kind of algorithm for training the available data for the efficient machine learning process. In this chapter, some of the most commonly used algorithms are discussed to get an overview and their efficacies.

Regression is a statistical modelling method to determine the relationship between dependent variables and one or more independent variables by iteratively refining the error in measurement. The dependent variables are commonly known as the outcome or target variables whereas independent variables are known as predictors or covariates. Few of the important regression methods are Linear, Logistic,
Polynomial, Stepwise, Ridge etc. Among all, Linear and Logistic regression are frequently used in machine learning; are discussed below.

### 7.4.1 Linear Regression

Linear regression is a supervised machine learning algorithm where the statistical relationship is modelled between two or more “continuous variables”. The independent variable is called as a Predictor and the dependent variable is known as a target variable. If the single predictor is used for determining the target variable – it is called Simple Linear Regression; if more than one predictor is used, it is called Multiple Linear Regression. In both, the linear predictor functions are used for estimating the outcome.

Linear Regression uses a regression line, shown in Fig. 7.7, for predicting the trend of the dataset; it is created by using the Least Square method. Mathematically the regression line can be represented by equation

\[ Y = mX + C \]

where \( Y \) is the Target and \( X \) is the Predictor; “\( m \)” is the coefficient concerning the slope of the line and “\( C \)” is the vertical intercept – both being constant. The regression line is drawn by the determined value of \( m \) and \( C \) from the dataset in such a manner that the error is minimized – i.e. the difference between the estimated values and the actual values are minimum.

![Fig. 7.7. Linear Regression Algorithm](image)

The relationship between the predictor and target variable can be considered deterministic if the predictor variable can accurately express the target variable. For instance, the sale of the number of umbrellas can be accurately expressed through
the amount of rainfall in the city. There is a direct relationship which can be correlated precisely through the Linear Regression algorithm.

### 7.4.2 Logistic Regression

The logistic regression is a statistical algorithm which is used for creating the predictive model when the target variable is of Binary Type. This implies that if the outcome is in the form of accept/reject, pass/fail, Yes/No, 0/1, High/Low, win/lose, or Go/No-Go – Logistic Regression should be considered as a suitable algorithm. Logistic Regression is a transparent algorithm that helps in evaluating individual predictors relation with the target variable. It works on the Sigmoid Function, shown in Fig. 7.8, that takes input as any real number and gives the output as 0 or 1, and the equation for the function can be represented as:

\[
Y = \frac{1}{1 + e^{-(mx + c)}}
\]

![Logistic Regression Algorithm Function](image)

**Fig. 7.8. Logistic Regression Algorithm Function**

Although the Logistic Regression is a simple algorithm that can be used for quick data training and good predictive outcome, it does not perform satisfactorily when the data distribution is uneven or messy. Its accuracy increases when there are minimum outliers and relationships between variables are not very complex. Logistic Regression is also not advisable where the data is of continuous, Fat-Data (i.e. having a large number of data feature columns and fewer rows) and Skinny data type (which have a large number of rows but fewer feature columns). In general, Logistic
Regression is considered most suitable for the initial quick benchmarking model which can give fairly good results.

### 7.4.3 Decision Tree

Decision Tree is a supervised machine learning classification algorithm which is used for the data which has the target variable in the binary form, similar to Logistic Regression algorithm. It is a method in which a graphical representation of decision making is presented as a flow-chart in the form of Tree-structure. The classification or sorting of decision is done similar to the tree branching; the root will have the details of the entire population and the branches will split the information categorically. A typical Decision Tree can be seen in Fig. 7.9, where the root of the tree contains the information of Capital Gain by the individuals.

![Decision Tree structure](image-url)
The root node then further splits into various branches or sub-trees nodes according to the different categories such as educational qualification, occupation, etc. Each sub-node will act as a test of certain attributes and based on these splits most significant variable is identified.

The main advantage of the Decision Tree Algorithm is its ease of interpretation due to its intuitive graphical representation of the outcome. The output is very easy to understand even if the persons has no analytical background. Another advantage is its ease of Data Exploration where the most significant variable can be easily identified and the relationship between the predictor and target variable can well be defined. It requires minimal data preparation and data cleaning for performing and is capable of handling categorical as well as a numerical variable.

Although the Decision Tree algorithm can manage the continuous variable data by discretizing the data before the model building, it is not considered suitable for continuous data. The small variation in data can result in the generation of a completely new tree-structure, therefore well-discretized data gives good predictive results. One of the common drawbacks with the Decision Tree is that it creates an over-fitting model which results in the formation of a very complex tree-structure. This can be solved by properly constraining the model parameters. In general, a Decision Tree model has low accuracy in predicting the outcome when compared with many other algorithms.

**7.4.4 Support Vector Machine (SVM)**

Support Vector Machines, in short SVM, is a supervised machine learning algorithm. Although, SVM is primarily used for the classification problems it can be applied for regression problems. For a classification problem, SVM defines an optimal hyperplane which can maximize the support vector margin from the closest element of the different classes.

For instance, consider a data set which has two separate categories of cars and prediction is to identify the type of the car based on the different features – i.e. predictor variables. Through SVM algorithm the classification can be done by initially plotting each data point, shown in Fig. 7.10, by considering each feature as the value of a specific coordinate. The plotting is done on the “n-dimensional” space where n
Is the total number of features. Thereafter, the decision boundaries are created – these are the separation lines which is closest to a given category that differentiates it from other categories. Further, an optimal Hyperplane is determined by maximizing the distance between the decision boundaries through the help of support vectors – this is used for differentiating the classes. Support Vector Machine (SVM) algorithm is very useful in the case of unstructured or semi-structured data such as image classification and text or hypertext categorization. The character recognition of hand-written text, cancer diagnosis, intrusion detection and satellite data classification are some of the areas of application of the SVM algorithm. It has less probability of over-fitting of data, which is an issue with the decision tree algorithm. The prediction accuracy of SVM is generally high if a clear separation margin exists between the different categories. It is also considered as an efficient memory algorithm.

The major drawback with the SVM algorithm is it does not perform satisfactorily if the target data set is overlapping – i.e. high noise. It is also not suitable when the data set is very large as the data training time increases considerably. Through the SVM algorithm, it is also very difficult to interpret the final outcome for relating the weights of predictor variables and its impact on the target variable. Although the SVM algorithm can be used for non-linear classification algorithm through kernel function, it performs best for linear separation.
7.4.5 Naive Bayes

Naive Bayes is a probabilistic type of classification algorithm in machine learning which assumes that the predictors are independent and contributes equally. This implies that the Naive Bayes algorithm considers that a particular feature of predictor class is not present in any of the other class. For instance, if the classification of fruits is the problem to tackle, then an apple will be defined based on the red colour, round shape and diameter of about 3 to 5 inch. Although these features are inter-dependent, they independently contribute to describing the fruit being an apple – hence called Naive.

This algorithm uses Bayes theorem which is a mathematical expression for computing the conditional probability. The Bayes theorem determines the occurrence of the probability of an event based on the assumption that the other event has already occurred. Mathematically, Bayes theorem can be expressed as

$$P(A|X) = \frac{P(X|A) \cdot P(A)}{P(X)}$$

In this expression, A is the Target variable and X is the Predictor variable; P(A|X) is the probability of A occurring when X already occurred; P(X|A) is the probability of B occurring when A already occurred; P(A) is the probability of occurrence of A; P(X) is considered as the probability of occurrence of X.

Naive Bayes is a super-fast classifier algorithm that performs better than the other algorithms. It is easy to implement and with a small amount of data training is can predict the outcome accurately in real-time. Text classification, sentimental analysis (i.e. positive or negative emotions), spam filtration, detection of Alzheimer’s disease and face recognition are some of the application areas of Naïve Bayes algorithm. The major drawback with this algorithm is that it considers all the predictors to be independent, but in reality, it is not always true. Hence the predicted outcome needs to be validated before its implementation.

7.4.6 K-Nearest Neighbor Algorithm (K-NN)

K-Nearest Algorithm (K-NN) is a supervised learning algorithm which is primarily used for classification problem. It is a non-parametric algorithm and considers entire data as training data without making any assumption. Initially, K-NN algorithm stores
the training data in a structured form and does not try to learn from the data. When any prediction is to be made for any unseen data, it tries to find a similar set of data from the training data and put it into the classified category. In a way, it performs the predictive analysis by finding the similarity with the previously available data. The “K” in K-NN represents the number of similar data (considered as a neighbor) found in the training data set for the newly fed data and its value lies from 1 to n. The algorithm is said to be as good as the number of neighbours, i.e. if K-value is high, the accuracy of prediction is better.

K-NN algorithm is used for those applications where the similarity between the items are to be identified. It is a powerful classifier algorithm which gives good predictive outcome even for the noisy data. K-NN is very simple to implement and very effective when the training data set is very huge – as no training step is involved in it.

The K-NN algorithm uses the computation of Euclidean distance for determination of the value of K. This leads to high computation cost because of calculation of the Euclidean distance needed to be performed for all the training data points. Hence, if there is a large set of data, the algorithm prediction shows down. Any error in the computation of the value of K will result in poor performance of the predictive model. Another disadvantage of the K-NN algorithm is that it is very sensitive to the outlier data. It also has no capability to deal with the missing data, thus data cleaning is very important for applying this algorithm.
8 Limitations of machine learning

Machine learning, a subset of Artificial Intelligence, enables the machine to mimic like humans and learn from its own experience (i.e. past data) for making informed decisions. In recent years, Machine Learning application has gained a significant pace and considered to be a revolutionary technology of the future. The machine learning application can be seen in almost all realm of human’s life; be it an e-mail spam filter or an online search engine or detecting critical disease in health care – Machine Learning is ubiquitous. The machine learning technology makes a machine smart and intelligent and gradually society is becoming more reliant on these intelligent machines that ease human’s effort. Despite having so many success stories and its revolutionizing capabilities, machine learning still has several limitations and required substantial efforts to overcome these. Some of these limitations are discussed below.

8.1 Limitation 1 — Data Requirement for Training ML Model

In machine learning, the models are trained with the dataset instead of explicitly programming it. The dataset is fed into the machine learning model and trained by using computer algorithms for predicting the outcome; as shown in Fig. 8.1. The efficacy of the model solely depends on how well it is trained, and training depends on the dataset. It is a well-known fact that larger be the amount of dataset, greater will be the information provided to

![Fig. 8.1. Machine Learning Training](image-url)
the model and better will be the training. With machine learning, there has always been an issue with obtaining the quality data. In many cases, the available dataset is not sufficient to train the model appropriately due to which relationship between the target and dependent variables are not established properly, thus resulting in an erroneous outcome which is never close to the actual result. If Machine Learning is expected to perform like a human, an enormous amount of data is needed to train the model. Since data acquisition is not easy, the applications of machine learning are limited.

### 8.2 Limitation 2 — Data Quality Dependency

In machine learning, the algorithmic model extracts statistical information from the training dataset and correlate variables for determining the pattern or trend to predict the outcome. Therefore, the type of information available in the dataset also plays a significant role. A large amount of dataset with greater noise will result in a poor correlation of the variables which leads to a reduction in prediction accuracy. In a simple way, it can also be inferred as – a large amount of poor-quality data will also result in an erroneous and inaccurate outcome. Therefore, it is very important to check the statistical characteristics of the dataset, Fig. 8.2, to determine the amount of noise present in it before using for machine learning model training. The availability of the quality data to train the model is another big limitation with Machine Learning.

![Fig. 8.2. Data quality check](image)

### 8.3 Limitation 3 — Data Bias

The machine learning dataset will never have parity among all variables. Some predictors will have dominating features compared to others which will have a greater influence on the target variable; Fig. 8.3. Thus, the predicted outcome will always be biased.
For instance, consider a machine learning model is required to classify a flower species named - Primrose, and the dataset for training the model consists of variables which define the type of flower through characteristics such as petals length, stamens, carpels, etc. Among all, if petal length is the most influential predictor, and the testing data of flower does not have the required petal length of Primrose, then even if it is a Primrose flower, the machine learning model will not predict the outcome as Primrose, despite satisfying other parameters. This bias may also exist for the noise present in data which may show the model performing extremely well but in actuality, it is influenced by the noise.

### 8.4 Limitation 4 — Expensive Computation

The accuracy of prediction through machine learning mainly depends on the training of the algorithmic model by using past data. The greater is the dataset for training better will be the prediction – additionally expensive will be the computation. For instance, consider weather forecasting similar to MM5 model need to be performed through machine learning. To do so, a huge amount of climate data is used to train the model to forecast the next day weather. These model training is done throughout the day and requires huge computational power to process the data and correlate the predictors. This expensive computation is also considered to be a limitation in machine learning.
8.5 Limitation 5 — Data Labeling limitation

The supervised learning requires the dataset to be labeled in a systematic manner, based on which model training uses historic data for defining target attributes. However, there is a high possibility that the raw data obtained through various techniques do not have a proper label and machine learning model cannot ingest that directly. Therefore, the data cleanup and reorganization are needed before feeding into the model training. Although, it is not rocket science, but to perform such operation for a huge amount of data is tedious and time-consuming; without this machine, learning prediction will be inaccurate. Thus, the data labeling is another limitation which affects the machine learning applications.
9 Deep Learning

The Artificial intelligence systems are “trained” systems rather than a “programmed system” that requires huge amounts of data to perform multiple complex tasks. Deep learning is a sub-category of Machine Learning in the domain of Artificial Intelligence which has the capability of learning unstructured or unlabeled data through unsupervised learning. In this learning technique, human brain data processing ability is imitated for analyzing and taking predictive decisions. Deep Learning also known as Deep Structured Learning or Deep Neural Network uses networks of artificial neurons for extracting high-level insightful features from raw input data. Due to the usage of artificial neurons for processing of information and distributed communication nodes, deep learning is also widely known as Artificial Neural Network (ANN). Although ANN is designed to mimic the human brain which is dynamic and analog, the erstwhile tends to be static and symbolic.

9.1 Evolution of Artificial Neural Network

The Artificial Neural Network is a mathematical model used for computation that simplifies and simulates the working of the human brain. This model uses a large network of neurons positioned as layers and inter-connected with some weighted function. The first neural network was defined in 1943 by Warren Sturgis McCulloch and Walter Pitts. During the initial phase this neuron network acted as a verifier to confirm that these networks can be used as computational models; but can’t be used for training and learning. Since then it followed a steady development phase; in 1958 Perceptron model was developed by Frank Rosenblatt for computing pattern recognition. This Perceptron model was later enhanced to Multi-layered Perceptron using backpropagation in 1986 by Paul Werbos and finally evolved as Deep Neural Network in 2006. Between the years 2009 and 2012 Jurgen Schmidhuber’s research group developed the most comprehensive models of recurrent neural network and deep feedforward neural networks which are widely used nowadays.
9.2 Working of Deep Neural Network

Neural Network computation is based on the type of architecture it uses for learning input-output relation. The neural network algorithm is inspired by the nerve cells (neurons) of the human brain for the creation of computational model. A Neuron is the structural and functional unit of the nervous system that consists of four main parts – Axons, Axon terminals, Cell body, and Dendrites, Fig. 9.1. The cell body is made up of nucleus mitochondria and other organelle cells which governs the growth of cell body and carries the life process of neurons whereas other three communicate and integrate with other neurons. The dendrite receives information from the previous neuron cells and transfers the signal to the cell body which further conducts the information towards next dendrite neurons. Each neuron consists of a single Axon, which transmits signals out from the nerve cell and is a long and thin wire-like structure. Axon gains information from the central nervous system and brain chord and is capable of carrying information reliably and quickly across the neuron and transfers to the dendrites of the next neuron. Unlike axon, dendrites can be several in number in a single neuron and has the tree branches like appearance which has a primary function of combining and integrating the received information.

![Fig. 9.1. Multipolar Neuron Cell](image)
The neuron cell takes the input signal, processes the information and transfers to the other neuron. Based on this fundamental theory artificial neurons were designed and developed for computational models of Deep Neural Network. In 1958, Frank Rosenblatt created the “Perceptron” model for pattern recognition. The perceptron model uses a single neuron which takes input and adds appropriate weight and their weighted sum is then passed to an activation function, shown in Fig. 9.2. These activation functions, such as sigmoid, linear or hyperbolic tangent function, is used by the neurons in the hidden layers of the model for achieving the targeted output.

![Perceptron Model](image)

**Fig. 9.2. Perceptron Model**

In neural network, computation in neurons is based on bias, weight and activation function. The activation function in the network is a mathematical gate between the current input neuron and outgoing next neurons. The type of activation function decides whether the present neuron should be activated or not, and is dependent on the weighted sum involved with the neurons. The three commonly used activation functions can be represented mathematically for the targeted output ($y$) as:

- **Sigmoid Function**, \[ y = \frac{1}{1 + e^{-net}} \]

- **Linear Function**, \[ y = net \]

- **Hyperbola Tangent Function**, \[ y = \frac{1 - e^{-net}}{1 + e^{-net}} \]

Where,
\[ net = W_0 + \sum_{i=1}^{N} W_i x_i \]

\( i = \) Number of nodes from input layer \( = 1 \) to \( N \)

\( W_0 = \) Bias of the node,

\( W_i = \) weight for the connection from \( i \)-th node of input layer

\( X_i = \) input from the \( i \)-th node of input layer

Due to several limitations of perceptron model, such as its output value can be either 0 or 1 due to the limitation of the transfer function and its applicability for classifying a linearly separable set of variables, the model had undergone several modifications with the passage of time to enhance its capability. These modifications were greatly towards the stacking of the artificial neurons and the flow of information between neurons to get the desired outcome. From single neuron to multiple neurons stacked in series or parallel or both were created for increasing the computational capability of the model, as shown in Fig. 9.3 (a - d).

![Fig. 9.3. Stacking arrangement of neurons](image-url)
The stacking of multiple neurons is typically done in a layered form which creates a complex structure that is widely called as Artificial Neural Networks. Generally, a neural network consists of three types of layers – Input layer, Output Layer and Hidden Layer and in each layer, multiple neurons are stacked. There can be more than one input and output parameters and can have multiple sub-layers as shown in Fig. 9.4. These sub-layers are known as hidden layers which perform computation on the weighted input for determining the targeted output.

Fig. 9.4. (a) Artificial Neural Network; (b) Deep Neural Network

The different nodes can have different activation functions that can add up linear combinations of their previous layer. Stacking a greater number of hidden layers will create a highly non-linear flow of information where each layer is performing sequential operations. The network in which stacking of neurons with two or more hidden layers are present is known as Deep Neural Network, shown in Fig. 9.4 (b). With the evolution and better performance of deep neural network algorithm, it has now become imperative to know how to select the number of hidden layers and number of neurons in each hidden layer which plays a significant role.

### 9.3 Selections in Hidden Layer

The stacking of neuron in the hidden layer plays a vital role in formulating an efficient computational neural network model. There are two types of parameters required to be defined for hidden layer – (a) number of hidden layers and (b) number of neurons in each hidden layer. There are variety of rules that can be applied for determining these two parameters effectively - either theoretical understanding of
the problem situation or through specially designed algorithm. The general rules for selection are mentioned below:

(a) Selection of number of hidden layers

1 hidden layer is used for the problem statement that can be represented through linearly separable function. This means that the output values can be separated through the linear line as shown in Fig. 9.5

2 Hidden layers are opted for the condition where two finite spaces are required to be mapped continuously. Single and double hidden layers are used for a simple data set and in general capable of defining most of the real situation effectively.

More than 2 Hidden layers are selected when the data set is highly complex and involve time-series for the representation of features. For computer Vision type of problem more than 2 hidden layers are advisable.

(b) Selection of number of neurons in single hidden layer

Defining the number of neurons in each hidden layer is equally vital in neural network architecture. If a smaller number of neurons are considered, then it may give rise to the under-fitting problem where available neurons are not able to detect the signals from the considered data for recognizing a pattern. On the other side, if a greater set of neurons are incorporated in the hidden layer it will create an over-fitting problem, where lack of data set will lead to inadequate training of neurons in the layer. Both the issues will lead to the poor performance of the predictive model; thus it is advisable to do a trade-off between the upper and lower limit of the number of neurons.

The number of neurons in a hidden layer should be between the size of the input and output layer. Ideally, the number of neurons is determined by considering two-thirds of the mean of the total number of neurons present in input and output layer. For example, if there are 3 input parameters represented
by 3 neurons and 2 output dependent variables, then the number of neurons in each hidden layer should be approximately equal to 3. (i.e. \( \frac{2}{3} \ast (3+2) = 3.3 \))

In deep learning, selection of neurons in the hidden layer, the architecture of neurons in input, output, and hidden layer, as well as flow of information among the interconnected neurons, plays a significant role. Based on the type of information flow in the neurons, the neural network can be categorized into three types:

1. **Feed-Forward Neural Network**
2. **Recurrent Neural Network**
3. **Recursive Neural Network**

1. **Feed-Forward Neural Network** is a Multi-Layer Perceptron (MLP) model where information between neurons flows in unidirectional form. In this network, information generally originates at input node (x) and is passed through hidden layers which terminate at the output node (y), as shown in Fig. 9.6, i.e. information is fed forward. Every neuron (often called node) of a layer is connected with all nodes of the previous layer in the feed-forward network.

![Feed-Forward Neural Network](image)

Fig. 9.6. Feed-Forward Neural Network

The connections between nodes are weighted differentially which governs the knowledge of the network. The operation of the network involves two phases – learning phase and classification phase. During the learning phase, the weights involved with each node modifies in such a way that it represents the pattern correctly. On the other side, during the classification phase, the weights of the network remain fixed. The classification is done by associating the nodal weights that give the largest output value. The feed-forward type of neural network is used when
data are labelled and structured, such as supervised learning. In this mode of learning, data learning is neither time-dependent nor sequential.

2. **Recurrent Neural Network** is a network of neurons in which connections between nodes for sharing information are the feed-forward type with a feedback loop, as shown in Fig. 9.7. This implies that information is fed to one layer for computation and its output is added to the next layer as well as fed back to the same layer itself as a cyclic loop. The next layer processes the information and follows the temporal sequence and remembers the computation of previous nodes too. This allows the network to exhibit time-dependent dynamic behavior which is helpful for situations such as prediction of next word in a sentence that is dependent on the words that came before.

3. **Recursive Neural Network** parses the inputs in a sequential fashion. A recursive neural network is similar to the extent that the transitions are repeatedly applied to inputs, but not necessarily in a sequential fashion. Recursive Neural Networks are a more general form of Recurrent Neural Networks. It can operate on any hierarchical tree structure. Parsing through input nodes, combining child nodes into parent nodes and combining them with other child/parent nodes to create a tree like structure Fig. 9.8. Recurrent Neural Networks do the same, but the
structure there is strictly linear. i.e. weights are applied on the first input node, then the second, third and so on.

The appropriate stacking structure of neurons in the network helps in improving the model efficiency and is largely dependent on the type of problem it is applied for. However, it is equally important to assign the appropriate weight factor to each neuron for improving the effectiveness of network in deep learning. The determining and optimizing of the correct combination of weights substantially improves the computational capability of the model. There are different ways to optimize the weight factor of neurons, some of the commonly used algorithms are:

a. The backpropagation algorithm
b. Conjugate Gradient algorithm
c. BFGS algorithm

a. The backpropagation algorithm in a neural network is the learning algorithm where the optimization of the weights of the neurons is done by minimizing the error function using the gradient descent method. In this method, the weights of neurons try to adjust the weights based on the previous weights. At each step, it iteratively updates the weights by moving a short distance in the direction of the greatest rate of decrease of the error. It is not as accurate as of the other two algorithms. This method of optimizing the weights is slow and does not produce an effective outcome.

b. Conjugate gradient method is an algorithm for optimizing the weight of the neurons in neural network. In the backpropagation, algorithm adjusts the weights
in the steepest descent direction (negative of the gradient). This is the direction in which the performance function is decreasing most rapidly. It turns out that, although the function decreases most rapidly along the negative of the gradient, this does not necessarily produce the fastest convergence.

In the conjugate gradient algorithms, a search is performed along conjugate directions, which produces generally faster convergence than steepest descent directions. In this section, we present four different variations of conjugate gradient algorithms. In most of the conjugate gradient algorithms, the step size is adjusted at each iteration. A search is made along the conjugate gradient direction to determine the step size, which minimizes the performance function along that line. For larger datasets with more than a few thousand weights, Conjugate Gradient is faster than BFGS, and uses less memory.

c. BFGS algorithm is Broyden-Fletcher-Goldfarb-Shanno algorithm which is a popular quasi-Newton method used to solve large scale nonlinear optimization problems by computing Hessian Matrices. BFGS uses the solutions and gradients from the most recent iterations to estimate the Hessian matrix. For the datasets with few variables, BFGS is usually faster than the Conjugate Gradient. However, for networks with more than a few thousand weights, BFGS requires a lot of storage and could lead to a shortage of memory.

### 9.4 Difference between Deep Learning and Machine Learning

In machine learning, training of data is based on ‘task-specific’ algorithm whereas Deep Learning is based on ‘feature-specific’ algorithm. This means that in deep learning machine recognizes and learn the patterns, however in machine learning domain expert classifies the pattern that machine should look for to reduce the complexity; as represented in Fig. 9.9.
The Deep learning technique uses an end-to-end approach for problem-solving, which means it requires only the input data for providing the end result. On the other hand, Machine Learning breaks the problem statement into several components, solves it individually and clubs at the final stage to give the result. To better understand it, consider a problem of Image Classification where an image is given as input data and various objects are to be classified; as shown in Fig. 9.10. When Deep Learning technique is used, it takes image as input and extracts the required features, classify the objects and categorizes all based on trained data to give the required output. If machine learning technique is used for the same problem and considering SVM algorithm is used for object classification, then it will require a bounding box detection algorithm for first identifying all the objects which will further be used as input to recognize the relevant object categories. Compared to Machine Learning, Deep learning requires significant computational power and dedicated GPUs for performing the desired task, this could be attributed to the computation involved in multi-layered neurons and their correlations.
Fig. 9.10 Object detection through Deep Learning [6]
10 Altair Knowledge Works™

Knowledge Works platform is a trademarked technology of Altair Engineering. Altair is a global technology company that provides software and cloud solutions in the areas of product development, high performance computing (HPC) and data intelligence. Altair enables organizations across broad industry segments to compete more effectively in a connected world while creating a more sustainable future.

Knowledge Works software is a desktop-based predictive analytics and machine learning solutions that will help you quickly generate actionable insight from the data. Knowledge Works data intelligence solutions allow individuals and organizations to incorporate more data, unite more minds with agility, and engender more trust in analytics and data science. The platform enables users to quickly and accurately capture and prepare data for any project, use data to accurately predict outcomes and produce insight and foresight, and visualize trends and insights that help to decipher and communicate across the business.

A fully integrated system for data access, cataloging, data prep, governance, automation, predictive analytics and visualization. It also provides Visual, ML-driven forecasting that autogenerates execution code for advanced data science models as well as Embedded visualization for easy interpretation of models. It also consists of Augmented analytics in which Machine learning provides recommendations for data prep and relevant data sources and determines the trustworthiness of data assets.

Knowledge Works platform is open and flexible which enables teams to use favored algorithms and coding languages without restrictions. It also gives a collaboration platform leveraging social tools such as likes, follows, comments and shares. There are numerous benefits involved with Knowledge Works, some are highlighted below:

- Better decisions are made in less time, based on analysis that leverages all available data.
- Organizations can predict trustworthy outcomes and prescribe actions in response to those outcomes.
- Reporting and analytics are consistent, effortless and error-resistant.
- Barriers to insight are shattered – people can instantly find the information they need for analysis and use it.
- De-duplicates two of the costliest enterprise resources: data and effort.
- Individual productivity is optimized – no one ever repeats the same data work twice.
- Team productivity is optimized – no one ever repeats analytics and reporting projects that someone else already completed.

Knowledge Works platform is a data intelligence platform where a layer of value is created between your data and your stakeholders. It involves data preparation, data science and data visualization. Knowledge Works empowers organizations to face challenges with trust, literacy, diversity and complexity of data and removes ambiguity from analytics and data science. There are variety of tools in the Knowledge Works platform such as:

- Altair Knowledge Studio
- Altair Knowledge Studio for Apache Spark
- Altair Knowledge Seeker
- Altair Monarch

10.1 Altair Knowledge Studio

Knowledge Studio is a data mining and predictive analytics software tool which enables advanced analysts to build and deploy predictive models using repeatable workflows. Knowledge Studio features cover all stages of the predictive analytics life cycle, from data preparation and profiling, variable selection, decision tree analysis, building predictive models, cluster analysis, principal component analysis, market basket analysis, building scorecards, applying reject inference techniques, model performance evaluation, building and deploying strategies. An entire workflow or any part of it can be deployed using automated SAS code generation.
The Optimization algorithms available in Knowledge Studio maximize or minimize any user-defined metric representing your business objective while maintaining relevant business constraints. Linear optimization on the dataset level handles user-defined linear objective functions with linear constraints, covering problems on the level of aggregating the values over a given dataset. Nonlinear optimization can be performed on both dataset level and record level. For details, see the Help topics in the Optimization section.

**10.2 Altair Knowledge Studio for Apache Spark**

Knowledge Studio for Apache Spark is a comprehensive data science platform integrated with Apache Spark technology to provide advanced data mining and predictive analytics on large-scale distributed data structures such as Hadoop HDFS, Amazon S3, and other storage types supported by Spark. Data can be loaded directly from Hadoop HDFS and ViewFs, Hadoop Archive, Amazon S3, FTP and Network Shares. Hive tables, Text (CSV) and Parquet formats are supported. Data manipulation, profiling, modeling (including interactive decision trees and strategy trees), model evaluation and deployment are supported on Spark. Python code is automatically generated for all operations. Custom programs for data science tasks using any available algorithm libraries in Python and PySpark can be written within Knowledge Studio workflows. Jupyter Notebook can be launched from within a Knowledge Studio workflow for a more advanced coding environment. All Knowledge Studio workflow tasks on Spark can be scheduled for automatic execution. See the Help section "Integration with Apache Spark" for more information.

**10.3 Altair Knowledge Seeker**

Knowledge Seeker is a data mining and predictive analytics software tool used by business analysts and advanced data scientists for data exploration, decision tree analysis, predictive modeling with decision trees, and strategy development. Knowledge Seeker capabilities include data preparation and profiling with advanced visualization, decision trees, strategy design, model performance evaluation, and deployment of decision tree models and strategies. The product provides the capability to create repeatable workflows and execute them with automatic data
refresh. An entire workflow or any part of it can be deployed using automated SAS code generation.

10.4 Altair Monarch

Monarch is a desktop-based self-service data preparation solution. Monarch connects to multiple data sources including structured and unstructured data, cloud-based data, and big data. Connecting to data, cleansing and manipulating data requires no coding. Monarch can quickly convert disparate data formats into rows and columns for use in data analytics. Over 80+ pre-built data preparation functions mean data preparation tasks can be completed quickly and error free. More time is spent on generating value from data as opposed to making data usable to begin with.

Monarch is designed for ease of work for data architects, data engineers, and business analysts. Monarch’s intuitive wizard driven interface has prebuilt functions to easily transform messy data into useable data sets for further analytic use. Models built in Monarch can be exported into common BI or other analytics platforms. Monarch is the fastest and easiest way to extract data from dark, semi-structured data like PDFs and text files as well as Big Data and other structured sources. It builds trust in your metrics with auditable change histories and clear data lineage tracking.

Monarch is a desktop-based self-service data preparation tool which eliminates the need for IT to extract data, that is often considered to be very time consuming and labor intensive. Automated repeatable processes with reusable models and workspaces allow workers to spend their time analyzing data instead of preparing it for analytic needs.
11 Altair Knowledge Studio™

Knowledge Studio is an open, flexible predictive analytics and machine learning platform designed for data scientists and business analysts alike. Data scientists who are experienced in coding with R, Python or other languages can reduce or eliminate repetitive tasks while building sophisticated models using favoured algorithms, using Knowledge Studio’s intuitive and interactive visual interface. Less experienced users can quickly build the multiple steps that are part of the data science lifecycle using wizards to complete a model’s workflow. There are numerous benefits of using Knowledge Studio; a few are:

*Single Interface:* Knowledge Studio requires only one interface to scale data science across the entire business, driving collaboration and governance as business analysts and data scientists work together to solve complex business problems.

*Connection to Data:* Studio has the capability to connect with any data regardless of the source. It can handle structured and unstructured data, cloud, open source languages and databases.

*Data Visualization:* Quick and easy data mining is a core requirement for machine learning. Powerful data visualizations in Knowledge Studio allow users to quickly see variables in the dataset and detect any errors. It enables effective variable selection and provides efficient illustrations of required variable transformations. Visual predictive power statistics enable the user to narrow in on variables essential for machine learning and eliminate redundant variables in the modeling process.

Model Data: Through Knowledge Studio it is easy to find insight from data using 80+ code-free nodes and advanced modeling techniques. These techniques include – market basket analysis, regression models (e.g., linear, logistic, constrained logistic, PLS), regularization, cluster analysis, multi-layer neural networks, decision trees, strategy trees, text analytics, and natural language processing. Various other models are also supported by the software.

Having known the benefits and utility of the Knowledge Studio software, now its imperative to delve deeper to get acquainted with the software.
Setting of Connect for Knowledge Data Mining Engine

The Data Mining Engine is the core component of Knowledge Seeker and Knowledge Studio that performs data mining and predictive analysis operations such as calculating summary statistics, creating and scoring a predictive model, etc.

The Knowledge Studio has two license configurations – Standalone License and Server License: as shown in Fig. 11.1. In the Standalone configuration, both the Data Mining Engine and the user interface component reside on the same computer. In the Client/Server configuration, the Data Mining Engine resides on a server.

In all configurations, when KS Workstation is launched, it automatically connects to the Data Mining Engine, checks the license, and opens the Working Directory with your projects in the Project Explorer (the left panel). The name of the Working Directory is displayed at the top of the Project Explorer panel. In the standalone mode, you normally would not need to connect to the Data Mining Engine manually unless the Working Directory specified in the current settings is not accessible or does not exist.
Getting acquainted with the Platform

To start Knowledge Studio software from Start menu, click on the Start button in the Taskbar at the bottom left corner, click on All Program folder, and then select KS Workstation for opening Knowledge Studio. (Desktop icon can also be double clicked to launch the software). After opening, Knowledge Studio platform will appear as shown in Fig. 11.2.

![Knowledge Studio platform](image)

The Knowledge Studio platform start up screen will contain the details of its installed version, working directory and option to start the new project. At the top, it has a multi-utility tool bar. The creation of New Project will launch a separate section as shown in Fig. 11.3.

The Project Platform

A project contains workflows and the objects are produced by executing operations in these workflows, i.e., datasets, models, reports, etc. Each project is represented by a folder in the file system. The platform contains Project Explorer, Work-Flow Process Bar, Work-Flow Canvas, Docking Panel along with Multi-Utility Tool Bar and Canvas Control at the bottom left.

**Project Explorer**

The Project Explorer (the left panel) shows all projects in the current Working
Directory, which is set using the **File | Set Working Directory** command. You can view and work with multiple projects in the Project Explorer at the same time. Each new project can be created by selecting **File | New Project** from the menu and is represented by a physical folder in your current Working Directory. Individual project components (models, model analyzers, etc.) are stored as files in the project folder. Conversely, every folder inside the current Working Directory is considered a project. In fact, files can be copied to a project folder and they will appear in the Project Explorer after the project is refreshed. Any project can be read by the KS Data Mining Engine of the same version on any supported operating system. The Project Explorer panel also displays the list of all projects in the Working Directory.
Fig. 11.3. Knowledge Studio Project Platform
Any project can be expanded by clicking on the triangle on the left-hand side to view its contents. The Working Directory name is displayed at the top level of the Project Explorer panel. All projects contained in the Working Directory are displayed beneath its name. In this example, there are six projects in the Working Directory. The hierarchy of objects within a project is displayed in a typical Windows Explorer-like fashion. At the top of the hierarchy is the project. Attached to the project are dataset objects, models, reports, generated code, model analyzers, and possibly some desktop documents.

The Project Explorer panel can also be automatically hidden (auto-hide) by clicking on the pin at the top right corner of the Project Explorer panel. To switch back from the auto-hide mode, open it and click on the pin again.

**Work-Flow Process Panel**

A work-flow process Panel defines a data mining process in terms of operational nodes and data flows. A typical process starts with extraction and preparation of the data which goes through modeling and ends with model deployment or outputting deployment code; it also has the option of exporting data. Similar to a program, a workflow is a description of interconnected operations together with their inputs and outputs which can be re-run multiple times with different inputs.

Altair workflow execution features allow to re-run individual nodes and segments of the workflow as well as the entire process. One of the most useful deployment features is automatic SAS code generation for all stages of the Scorecard building workflow. Code can be generated for the whole workflow, sub-processes, or individual nodes. Then it can be copied directly into SAS environment and run there. Other types of code such as SQL Function can also be generated.
Managing the Work-Flow

More than one process can be created in a single workflow canvas, one can also have multiple workflow canvasses in a single project. If one canvas gets too crowded, create a new one by either clicking the plus sign or by right-clicking on the Workflow tab at the top of the Node Group panel and selecting New from the context menu, as shown in Fig. 11.4. Switch between multiple workflows by clicking on their tabs. To change the order of the Workflow tabs, drag one of the tabs to the desired position relative to other tabs. You can also use the right-click context menu of the Workflow tabs to rename or delete workflows. The same commands are available under the Workflow menu.

While creating and running workflows you can also easily keep track of the objects produced by running them. The project folders and objects created inside them are displayed in the Project Explorer panel on the left-hand side. Aside from showing all projects in your Working Directory, Project Explorer allows you to view outputs and delete the outputs that are no longer necessary.

Work-Flow CANVAS

Canvas is the Visual Interface in the Knowledge Studio. The Altair Knowledge Works work-flow automation feature allows to easily construct such workflows in a visual, interactive way: by dragging, dropping and connecting process nodes on the visual canvas and specifying their inputs, outputs, and parameters. The process
defined by a workflow constructed this way can easily be re-run on updated or refreshed data inputs. Particular data transformations, modeling, and other operations can be defined in a Work-Flow Canvas and can also be repeated as long as the structure of the new input data are the same.

**Working with Work-Flow Canvas**

1. To Add Work-Flow nodes to Studio Canvas

A Node on the Canvas represents an operation or set of instruction which the user desires to perform in the Knowledge Studio platform. For instance, to import the text file into the Knowledge Studio, a text import node is inserted into the canvas.

![Nodes workflow arrangement](image)

Fig. 11.5. Nodes workflow arrangement

These nodes can be added into the Studio Canvas through Work-Flow Process Panel or Right-click on the Canvas Space and select the required tool for the desired operation. Fig. 11.5 shows an example of the nodes workflow arrangement that builds a decision tree, a logistic regression, a scorecard, it also scores it and analyzes its performance.
Methods to Add Nodes

There are several ways to add nodes to the workflow canvas:

i. **Drag-and-drop:** Drag the desired node from a functional palette (Connect, Profile, Manipulate, Model, Evaluate, or Action) to the workflow canvas.

ii. **Mouse Control:** Right-click on the canvas and select the desired node name from the hierarchical context menu, as shown in Fig. 11.6.

iii. Drag a connection arrow from the input node, as shown in Fig. 11.7, to an area on the workflow canvas. Release the mouse and select the desired node category and then the node name from the context menu.

Fig. 11.6. Adding Nodes through mouse to the Canvas Studio

Methods to Manipulate Nodes

Nodes can be moved around the workflow canvas simply by dragging and dropping.

- To move a single node, click inside the node (closer to the centre rather than the edge) and drag it to the desired location.

- To move a group of nodes, select the desired group first by dragging a rectangle over the area enclosing these nodes as shown in the Fig. 11.8.
The nodes in the selected group will become highlighted with a light-gray frame around each node. Then click inside any node in the group (closer to the centre rather than the edge) and drag it to the desired location. This will move the whole group, provided that it is still highlighted.

**Aligning the Nodes in Studio Canvas**

To align a group of nodes, select the desired group by dragging a rectangle over the area enclosing these nodes as shown above. The group becomes highlighted. Right-click on any node in the group and select "Align Horizontally" or "Align Vertically" from the context menu, as shown in Fig. 11.9.

Fig. 11.8. Manipulating group of nodes

Fig. 11.9. Aligning the nodes
Auto arranging all Nodes in a Workflow

To arrange the nodes in a workflow, and to align all connections in the horizontal direction, click the Auto-Arrange button in the toolbar. To undo the change, click the button again. Fig. 11.10 shows before and after auto-arranging of the workflow:

![Before Auto Arrange](image1)

![After Auto Arrange](image2)

Fig.11.10. Auto arranging the nodes

Supernodes usage and its creation

A supernode is a cluster of nodes merged into a single node. Supernodes represent a convenient way to organize workflows in the Canvas Studio. While building or viewing large and complex workflows with a lot of nodes and connections, a group of nodes can be merged into a single "supernode" to simplify the work-flow. This way whole sections of a workflow can be contained inside a single node for operational and viewing convenience. The functional schema of the workflow is preserved as the supernode's input and output connections are the same as those of its members. A supernode can be ungrouped at any time to restore the original representation of the corresponding group of nodes on the canvas.
Creation of a supernode,

For creating the Supernode, the selection of the desired group of nodes need to be done in any of the following ways:

- By dragging a rectangle over the area enclosing these nodes as shown in Fig. 11.11.

- by clicking on the desired nodes while holding down the CTRL or SHIFT key.

The nodes in the selected group will become highlighted with a light-Gray frame around each node. You can add/exclude nodes to/from the current selection by clicking on the (un)desired nodes while holding down the CTRL-key. Right-click on any node in the highlighted group and select Make Supernode from the context menu, as shown in Fig. 11.12.

A light-blue frame enclosing the selected group of nodes indicates the supernode. Any nodes that were not in the original selection are shown as partially shaded. In the Fig. 11.13, only the 6 tree nodes comprise the supernode. To merge (collapse) the group into a single node, click on the "minimize" control at the upper-left corner of the supernode frame, as presented in Fig. 11.14.
As a result, the highlighted group is merged into a single node labelled "SuperNode", as represented in Fig. 11.15. To rename a supernode, click on its label and edit the text as necessary. To ungroup a supernode, right-click on it and select Ungroup SuperNode from the context menu, as shown in Fig. 11.16. This can be done in any state of the supernode (compact or expanded).
Mini-Map Creation

An interactive mini-map of the workflow can be opened by clicking the mini-map button in the toolbar. A typical Mini-map formation can be seen in Fig. 11.17. It shows the current view of the workflow relative to the whole workflow and allows to easily navigate across large workflows that don't normally fit in a single view, as well as zoom in and out.

Fig. 11.17. Mini-map of the Work-flow

A transparent rectangular frame inside the mini-map serves as a movable lens or magnifying glass and indicates the part of the workflow shown in the current view. The fastest way to navigate to the desired part of the workflow is to click on that part in the mini-map. The frame will move, and the Workflow view will be updated immediately to show the part you clicked on. Alternatively, click anywhere within the frame and drag it around the mini-map frame, dropping at the desired position. The view of the workflow will be updated dynamically depending on the position of the frame.

Resizing of the mini-map as well as resizing of the "magnifying" frame by dragging on any of its corners can be done. Changing the frame size will cause
the workflow to zoom in or out. Each project can have only one mini-map. If you switch to another workflow within the same project, the mini-map will change accordingly to show the current workflow.

**Docking Panel**

If there is a need to build or view large and complex processes in your Workflow canvas, some space can be saved by docking the Node Groups panel to the left. Just click on the arrow in the middle of the separator between the Node Groups panel and the canvas.

**Canvas Zoom / Pan**

The Work-Flow Canvas visual section can easily navigate across a large workflow using pan and zoom features when the workflow is larger than the application window. *Zoom in/out* using the standard zoom control with a slider provided in the Status bar at the bottom right corner, as shown in the Fig. 11.18. Zooming can be performed in several ways:

- Moving the slider or clicking the +/- icons on the zoom control.
- Scrolling the mouse wheel while pressing down the CTRL key.
- using the keyboard shortcuts
  - CTRL - (zoom out)
  - CTRL + (zoom in), and
  - CTRL 0 (restore zoom to 100%).

![Fig. 11.18. Zoom control](image)

*The pan* control allow to drag the workflow canvas around with the mouse instead of using the scroll bars. While holding down the CTRL key on your keyboard, click anywhere in the workflow canvas, hold the left mouse button down and start dragging to move the canvas.
Menu Bar

The menu bar consists of multi-functional tools to control the various processes of the Knowledge Studio. These tools can help in creating new projects, saving/exporting the files, control Printing and setting the preference and options. Fig. 11.19. shows the Studio Menu and tool bars

Setting up of Working Directory

The Working Directory is the location where your projects and the related documents are stored. It is sometimes referred to as the Mining Location. If already connected, you can change the current Working Directory at any time using the menu command File | Set Working Directory or by clicking the second icon from the left on the application toolbar; as presented in Fig. 11.20.
Assessing Data Sources

To start the analysis of your data and build models using Altair Knowledge Works software, you must first open or create a project and make the data accessible within the project in one of the following ways:

i. Import the data from the source file or database into a dataset in the Knowledge Works format.

ii. If the software license includes In-Database Analytics, connect to the desired SQL Server, Teradata, or Oracle database using the In-Database Analytics node.

Once the data is accessible within the current project, it is represented as a dataset node on the Workflow canvas and a dataset object in the Project Explorer. A dataset node can therefore represent a) a Knowledge Works data file; b) a database table.

Dataset nodes representing data files on the Workflow canvas are references to the physical dataset objects. When you delete a dataset node from the canvas, you only delete a reference. The referenced dataset object is not deleted and is still displayed in the Project Explorer panel. The physical dataset is deleted only when it is deleted from the Project Explorer panel. An in-database table can never be deleted from the source database using the Knowledge Works application interface.

Import from data sources of the following types of files are supported:

- Text
- Microsoft Excel
- R File
- SAS Data Files
- SAS Transport
- SAS Transport CPORT
- SPSS Native
- SPSS Portable
- Native Altair datasets (KDD file format)

To add a data import operation into the workflow, drag the node representing the desired source format from the Connect palette to the workflow canvas. Alternatively, right-click anywhere on the canvas and select the desired format from the Connect menu. Double-click the new node on the canvas to start the dataset import definition wizard, or right-click on it and select Modify from the context menu.
12 Machine Learning with Knowledge Studio

Machine learning is the branch of computer science which deals with building the predictive mathematical models by using statistical tools and programming language which can help a machine train on its data and predict specific outcomes based on this training. The details and stages have been extensively discussed in chapter 6. This module deals with the practical aspect of Machine Learning by using Knowledge Studio. Although basics remain the same, there are certain additional options and features available within the Knowledge Studio Software.

Knowledge Studio is a market-leading easy to use machine learning and predictive analytics solution that rapidly visualizes data as it quickly generates explainable results - without requiring a single line of code. Knowledge Studio includes prebuilt data preparation and data science functions and integrates with common programming languages. For creating any predictive model, Knowledge Studio follows the below-mentioned steps:

→ Step 1: Create a project
→ Step 2: Import data from the required file.
→ Step 3: Explore data using data profiling tools
→ Step 4: Create Partitions of a dataset for training and testing models
→ Step 5: Build a Machine Learning model
→ Step 6: Validate and score a model
→ Step 7: Evaluate model performance
→ Step 8: Build a scorecard
→ Step 9: Deploying of the built model

To understand the above-mentioned Machine Learning process by using Knowledge Studio, consider a problem for creating a predictive model.

**TASK** Consider a marketing campaign is to be planned to target persons with
income less than $50,000. A Census data that is available for a local zone. The data consists of details about individuals such as age, working class, education, marital status, relationship, gender, native country, income and capital income/loss. It has a discrete variable Income with two values ‘>50K’ and ‘<=50K’. Based on the census data, a Machine Learning model need to be created by considering Income as a dependent variable to predict the variation of income in different segments of population by using a Decision Tree Algorithm. To solve it through Knowledge Studio, all the above-mentioned steps will be followed.

Step 1: Create a project
A new project can be created by using command menu File | Create Project or using shortcut “Ctrl + N”. Alternate option can also be seen on the main window after the launch of KS Workstation.

Step 2: Import data from the required file
Specify the source file to import and the name for the new dataset. Enter the path to the file or use the browse button to browse for the desired source file. There are two steps for importing the data as mentioned below:

i. Right-click anywhere on the workflow canvas and select Connect | Select file Import from the context menu, as shown in Fig. 12.1.

ii. Drag the type of Import node from the Connect palette to the Workflow canvas, as presented in Fig. 12.2.
For the “Census” data, the excel file was imported into the Knowledge studio by dragging the excel into the Studio Canvas | Double click the import node | Source file
name as Excel | Locate and select Source file name as “Census info.excel” | Select Auto-Detect Encoding mode | Next | Save.

Step 2: Explore data using data profiling tools

Data Profiling is the method of extracting the statistical information available from the existing database. At the data profiling stage, descriptive statistics analysis is done in order to:

- Assess data quality and identify variables that can be discarded to reduce the data dimensionality
- Explore the relationships between the variables to prepare for predictive analysis
- Discover important trends in the data
- Identify dependent & independent variables for predictive modelling

Double click on the imported Census info node available in the Studio Canvas to look into the data for performing “data profiling” analysis.

![Fig. 12.3 Raw imported data](image)

The Fig. 12.3 shows the overview of the imported data. The Overview Report is a dataset view that displays the dataset structure and univariate statistics such as the mean, maximum, and minimum values, the number of missing values, etc. Customization of the Overview Report can be done by choosing which statistics.
to display or hide using the command **Tools | Options** from the main menu. To set
global preferences for new datasets, use **Tools | Preferences**. Note that
Preferences are not applied to objects already created.

To see the actual imported data select the “Data” tab at the bottom
of the Studio Canvas and select the “Option” tool
then press OK to visualise the complete list. However,
initially, only first 100 rows of data will be visible, to see more rows click on “Show
more…” at top-right

After the data is imported and overview of the data is visualized, which is
imperative to perform the data profiling, i.e. analysis of the data, that helps to
understand to know the nature of its spread and other statistical characteristics.
This can be done selecting the “Overview Report tab at the bottom and
clicking on “Calculate All” button placed above-left to the Canvas.

By default, it will calculate the “Unique count, % of missing values, Maximum,
Minimum, Measure of central tendency (Mean), Measure of Dispersion(Std.
Deviation)”. However, if more statistical calculations need to be performed on the
imported data click on Options button and select the type of
computation to be performed. Fig. 12.4 shows the different statistical operations
which can be performed on the imported data to gain a better insight into the data.
Based on this information, a mathematical model can be generated which can
predict the outcome.
Fig. 12.4. Statistical analysis for Imported Data

Fig. 12.5. Analysed Imported data with statistical characteristics

Fig. 12.5 shows analyzed imported data with statistical characteristics after performing the computation. This data content can be selected and copied to any other platform too such as Microsoft Excel, Word, PowerPoint Presentation, etc.
To understand the Attributes and univariate statistics displayed in Overview Report, Table 12.1 can be referred.

Table 12.1. Attributes and univariate statistics

<table>
<thead>
<tr>
<th>Name and Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>#</strong></td>
</tr>
<tr>
<td>Field Name</td>
</tr>
<tr>
<td>Field Label</td>
</tr>
<tr>
<td>Data Type</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Basic Measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardinality</td>
</tr>
<tr>
<td>Unique Count</td>
</tr>
<tr>
<td># of Missing Values</td>
</tr>
<tr>
<td>% of Missing Values</td>
</tr>
<tr>
<td># of Non-Missing Values</td>
</tr>
<tr>
<td>% of Non-Missing Values</td>
</tr>
<tr>
<td>Minimum</td>
</tr>
<tr>
<td>Maximum</td>
</tr>
<tr>
<td>Sum</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Measures of Central Tendency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
</tr>
<tr>
<td>Measures of Dispersion</td>
</tr>
<tr>
<td>------------------------</td>
</tr>
<tr>
<td>Median</td>
</tr>
<tr>
<td>Mode 1</td>
</tr>
<tr>
<td>Mode 2</td>
</tr>
<tr>
<td><strong>Standard Deviation</strong></td>
</tr>
<tr>
<td><strong>Range</strong></td>
</tr>
<tr>
<td><strong>Interquartile Range</strong></td>
</tr>
<tr>
<td><strong>Coefficient of Variation</strong></td>
</tr>
<tr>
<td><strong>Std Error Mean</strong></td>
</tr>
<tr>
<td><strong>Variance</strong></td>
</tr>
<tr>
<td><strong>Skewness</strong></td>
</tr>
<tr>
<td><strong>Median Absolute Deviation</strong></td>
</tr>
</tbody>
</table>
Graphical Visualization of the Data Statistic

The statistical characteristics of the imported data can also be visualized graphically by the help of “Dataset Chart” tab which is at the bottom of the canvas. The Dataset Chart is a view of the dataset that displays the distribution of data values for a given field in a chart format.

The graphical chart can be altered with ease by using various graphical interfaces, as shown in Fig. 12.6, available on the plotting window based on the user requirement.

Note:
- If the field is continuous, the distribution is displayed as a histogram.
- If the field is discrete, the distribution is displayed as a pie chart.

Various chart display settings including the chart type, labels, colors, and fonts can be changed using the Chart menu or the context menu that appears when right-clicking the chart. If the chart data need to be seen then select the Data Editor item from the context menu. You can change the field being displayed by choosing from the drop-down list labelled Field or using the navigation buttons to the right of the...
drop-down list.

By default, the chart for discrete variables shows a maximum of 10 categories. If the cardinality of a discrete variable is larger than 10, the remaining categories are mapped into a category designated by the symbol '###'. If you need to increase the maximum number of categories displayed, click the Edit Grouping button to change the threshold value.

Fig. 12.7. Range Editor for defining bin boundaries

Range Editor allows to define the boundaries of the bins for a continuous variable and the way its data should be grouped and distributed across the bins, Fig. 12.7. The range editing allows to combine ranges by using Group option, however, the ranges that are continuous can only be grouped. By using the Break option, a new interval can be added to the distribution by specifying breakpoints within an existing interval. If any error occurred during grouping or breaking the range, Reset Option can be used to return to the original ranges displayed when the range editor was first opened.

It is important to note that in the Dataset Chart, a numeric, date, time, or timestamp variable is treated as discrete if its cardinality is less or equal to 10. In this case, its distribution is represented in a Pie Chart by default. Otherwise, it is considered to be continuous and its distribution is represented in a Histogram by default. The Edit Ranges button is only available if the field to display is continuous.
This feature allows one changing the displayed ranges of a continuous variable. The dataset chart and the associated data can be copied and pasted into a Microsoft Office document. This allows the rapid and easy preparation of reports, presentations, and summaries.

Another option to know more about the dataset, the “Data” tab at the bottom, third to the left, can be selected. If this tab is clicked for the first time, then it will prompt to select the fields that is desired to be displayed. All fields are selected by default; exclude the fields that need not shown. It should also be noted that new calculated columns added in the Dataset Editor are not displayed by default in the Data tab if it had been opened before you added the new columns. Click the Options button in the toolbar to include the newly added columns in the Data view if required.

To visualize the distributions of the independent variables under each category, Segment Viewer tab is used. It helps to visualize the segmented variable. In another way, it can be said that to build an early hypothesis, the segment viewer is used. It specifically helps to determine whether the distributions of the independent variables vary across the segmented variable. If this variation is seen, then this may be a variable of interest for the analysis. The segment viewer can be seen in the bottom of the Studio Canvas. When the first-time segment viewer is viewed for a dataset, an Options dialog appears; as shown in Fig. 12.8. Among the available variables, independent variables need to be selected individually and marked selected by clicking on command or selecting all by clicking on command. If the Option dialog box does not appear automatically, then it can also be called up manually by using Tools | Options command. Fig. 12.9 shows the data distribution of the independent variable of the Census data by using the Segment Viewer.
To better understand the Segment Viewer analysed data, consider Fig. 12.9 for the matter of discussion. The graphical plot shows the relation of all the independent variables against the dependent segment variable along with its “Information value” and “Entropy Explanation”; both representing the significance of the variable for prediction. The Information value is a measure of the difference between a “true” probability distribution P and an arbitrary distribution Q. On the
other side, the Entropy Variance which is a non-statistical measure tells how much information about a DV value is contained in a random draw from the node dataset. High entropy means low information and vice versa. Entropy \( E \) achieves a maximum of 1 for a uniform (evenly spread) distribution and a minimum of 0 for a degenerate (peaked) distribution.

In the Fig. 12.9, if the independent variable “Sex/Gender” is to be considered, then it can be seen that the entropy of the data is 0.04 which means it has greater significance to predict the income of the individual. It also shows that persons having income => 50k are majorly male. Another variable “Relationship”, shown in Fig. 12.10, shows that the majority of husbands’/wives’ income exceeds 50k. Thus, a Hypothesis can be made that “Male” persons have a high probability of having income greater than 50k. Similarly, other variables can also be examined to conclude a strong Hypothesis based on which predictive model can be defined.

Another valuable information with the available data can also be comprehended by using the “Cross Tab”, available at the bottom of canvas. The Cross Tabs view helps to visualize the data in cross tables between two or more variables. The cross tables can be viewed in the chart form or a tabular/report form. The switch between the chart mode and the report mode by clicking the Show As Report icon in the toolbar or selecting View | Show As Report from the menu.

When it is activated for the first time, the Option window will automatically pop-up, shown in Fig. 12.11. Thereafter, the required independent variable needs to be selected and added in the column section whereas the dependent variable is
added in the row section. To substantiate the previously made hypothesis age, sex, relationship, and education were added to the column section and the income was added in the Row section.

![Cross Tab Option Window](image)

**Fig. 12.11. Cross Tab Option Window**

The Cross-Tab outcome, presented in Fig. 12.12, shows that the male personals who have bachelor education and are aged between 39 to 46 have a greater probability of having income $\Rightarrow 50K$. These relations among variables help to understand the data in a much better way.
**Character Analysis**, positioned next to Cross Tab, is another method of data interpretation that shows the frequencies of each category of a dependent variable across various ranges (bins) of independent variables. This view helps to reveal relative trends in the distribution of the DV categories by plotting them side by side, making it easy to compare the trend of one category against the others. The options dialog, initially appears by default and, provides a choice of displaying charts and report as a percent of column or percent of rows. The default is percent of column. The initial chart is displayed based on the dependent variable selected in Segment Viewer which can be changed by using the in the Options dialog.

The **Correlations** view of the dataset object allows one to explore how or to what extent fields in your dataset are associated with each other. It displays correlation coefficients (default option) or covariances for the selected fields. These statistics can be displayed in the matrix form, in pairs, or in a chart based on the requirement. Covariance and Pearson correlation coefficients measure the degree of linear association between pairs of variables. The values of correlation coefficients are always between -1 and +1. The value of +1 indicates that two fields are perfectly related in a positive linear sense, -1 indicates that they are perfectly related in a negative linear sense, and 0 means there is no linear relationship.
between the two fields. The correlation coefficient of the two variables is obtained by dividing their covariance by the product of their standard deviations.

After visualizing the various parameters of the imported data and knowing various statistical details, it now becomes important to create the machine learning model for predictive analysis.

**Step 4: Create Partitions of a dataset for training and testing model**

Dataset partitioning is an important preparation stage in the modelling process. A model is usually built on a training sample and then validated on a test (or validation) sample in order to ensure the adequacy of fitting the model and to test its predictive power. To introduce a partitioning operation into your workflow, drag the **Partition** node from **Manipulate** panel onto the canvas and connect the dataset to the node; as shown in Fig. 12.13.

![Fig. 12.13. Add Partition Node and connect with the Imported Data](image)

To define the partitions, double-click the Partition node or right-click on it and select Modify. This brings up the partition dialog, as shown in Fig. 12.14. The total data was partitioned into two parts – 70% Training and 30% Testing data for the complete description of its functions. Other partitioning ratios such as 60/40 or
80/20 ratio can also be considered based on the type, quality and quantity of the available data.

**Step 5: Build the Machine Learning Model**

For creating the predictive model, the machine learning algorithm needs to be added to Training data of the “Census info”. There are various types of machine learning models available in Knowledge Studio; few of the widely used models are
- Linear Regression, Logistic Regression, Decision Tree, Cluster Analysis, Random Forest and Deep Learning Analysis. The selection of the model is based on the type of data available and the kind of predictive analysis to be done. For the present task, a prediction needs to be made about the income of the personals from the census data, a “Decision Tree” model is most suitable for the desired outcome. To add a Decision Tree process node to the workflow canvas and connect with the input dataset, following steps can be used:

a) Through Model Palette

Open the Model palette available in the Work-Flow Process Panel and drag the Decision Tree node onto the workflow canvas. The Decision Tree node is gray until its input is defined. Connect the Census dataset to the Decision Tree node by dragging a connection arrow from the dataset to the tree node, as shown in Fig. 12.15.

![Fig. 12.15. Adding Machine Learning Model to the Studio Canvas](image)

b) Right Click in Studio Canvas

On the Studio canvas space, Right-Click | Model | Decision Tree, as shown in Fig. 12.16. Thereafter, connect the Training Census dataset to the Decision Tree node by dragging a connection arrow from the dataset to the tree node.
Once the Decision Tree model is added to the Training Data it needs to be modified as per the requirement. To do so, Double-click on the Decision Tree or Right-click on it and select Modify from the context menu. Thereafter, the Decision Tree wizard opens in which the following details need to be fed:

**Tree Name:** Income_DecisionTree (since prediction model needs to be made for Income).

**Dependent Variable:** Select “Income” from the list (as stated in the problem statement).

After defining the Dependent variable and Tree name, select “Next”; as shown in Fig. 12.17. The next step allows to set the Split Search Method and Measure.

The Split Search Method determines the iterative process of merging groups of values into bins until an optimal binning is achieved according to certain criteria of the split significance. This procedure locates the pairs of most similar groups and merges them together. Upon re-evaluation of the binning, this process is repeated. Merges are subject to the constraints defined by the type of independent variable (nominal, ordinal, or continuous) and whether or not the groups are in fact "similar" by comparison of the cost function to a merging threshold value. Special data structures are used to make the search for similarity between two groups as fast as possible.
The Knowledge Studio platform provides three Search Method options, and each has its own significance.

**The Cluster method** finds groups that maximize similarity within the groups and dissimilarity between the groups. Merging stops when either there is only one group left or there is no pair of groups that is "similar" enough with respect to the "merging threshold". The Cluster method tends to find more natural patterns than Exhaustive. This method is the application default.

**The Exhaustive method** performs an exhaustive search to find groups that maximize statistical significance. Merging stops when there is only one group left. While merging, the sequence of merges is recorded in a special history buffer, noting the "best", global, solution found to date. When the merging process is finished, a re-initialization to the start point is performed, and the sequence of merges recorded in the aforementioned "history buffer" is performed. This process stops as soon as the "best" history entry (i.e., the noted best solution) is encountered. Splits derived from the Exhaustive method display the most significant relationships.

**The Forward Exhaustive method** is optimized for ordered variables. It begins with no split points and adds them iteratively. It combines binary split search applied to the bins of the current (k-th) iteration and ternary splits on the bins obtained at the previous iteration (k-1). In addition, at each iteration it uses a discrete mesh of
potential split points to increase the chances of achieving the global minimum p-
value, rather than stopping at a local minimum. The Bisection Samples parameter
determines the number of nodes in this mesh. The default is 10. The admissible
values are nonnegative integers. Zero means no sampling (i.e., a complete sweep is
performed). The higher the value, the more likely the method is to find the optimal
global solution – at the expense of performance. These optimizations affect only
ordered variables – the variables whose Usage is 'ordinal' or 'continuous'. For splits
on nominal variables, the regular Exhaustive method is used.

For the present task, the “Cluster” search mode and “Entropy Variance” measure
are considered and thereafter “Next” button is selected. The next step in the Split
Report page allows the user to select the variables to be used in the tree. As an
option, the strength of the relationship between the independent variables and the
dependent variable can be analyzed before making a decision on which variables to
exclude. Once all the necessary modifications are done in the Split Report page,
click on “Run” – this makes the decision tree node checkmark at the lower right
portion, turn green from red; as shown in Fig. 12.18. This represents the required
Decision Tree model had been successfully built on the training data.

Fig. 12.18. Set up of the Decision Tree on the Training Data

After setting up the Decision Tree model on the imported Census data, the
root node “Income_DecisionTree” can be double-clicked to view the basic
information about the data being analyzed. In the present case, it can be seen that
the dependent variable Income has two values: '<=50K' and '>50K'. With a total of
16,281 people in the dataset, 12,435 of them make less than or equal to 50K that’s 76.38%. The remaining 3,846 people (23.62%) earn more than 50K. Since the Analyze option in the wizard has already been performed, the first split of the tree is automatically displayed upon inserting the tree. However, if it does not appear, then right-click on the Root Node and select Find Split. The same can be done by clicking on the toolbar at the top of the Canvas Studio or using the key combination of ctrl + shift + F; as shown in Fig. 12.19.

Find Split searches for the independent variables (IV) that are most significant to the dependent variable (DV) based on criteria defined by the measure. The Independent variables that are found to be significant are called the split variable(s). The most significant split variable is displayed as child nodes where each child node represents a category or range found in the independent variable. The Find Split command will search for the significant relationships between our dependent variable Income and all the independent variables included in our analysis.

Fig. 12.19. Finding Split of Root Node for Decision Tree formation
This command can be used on any node of the tree, but it will start with the root node. For the current Task, **Find Split** of root node finds “relationship” to be the most significant variable with respect to the dependent variable Income. Looking at the >50K line for each of these nodes (Fig. 12.19), it can be noticed that the highest concentration of those earning over $50,000 is in the leftmost child node with the rule relationship = Husband/Wife node with 44.96% or 3,276 records. This is the node with the largest proportion of yellow.

Unlike Find Split, the Force Split command allows the user to choose the independent variable to split on and gives complete control over the bins of the split variable. In the default mode, Force Split does not use any algorithm to determine the most informative splits or rank independent variables. The split control is fully manual - the initial binning is "Equal Width" or "Equal Height", and the user is free to define their own custom bins starting from that point.

To activate **Force Split**, Right-click on the desired node of the tree that needs to be split and select Force Split from the context menu. Alternatively, select the Force Split command from the toolbar menu or use the keyboard shortcut CTRL + SHIFT + R. Fig. 12.20 shows the Force Split method and its outcome.

![Fig. 12.20. Force Split of Root Node for Independent Variable](image)
The **Optimal Binning** option in the dialog box is checked if an intelligent algorithm is to be applied to automatically produce the optimal binning for the selected variable - the most informative binning with respect to the dependent variable distribution. This is equivalent to performing the **Find Split** command followed by the **Go To Split** command and selecting the desired variable. Thereafter, Click Analyze to compute the relative significance of the variables with respect to the dependent variable; which is displayed in the Info column. The way Info is calculated depends on the selected measure.

From the Force Split Analysis, Fig. 12.20, it can be seen that “relationship” and “marital-status” has the most significant influence on the dependent variable “income” – i.e. 21% and 19% respectively. Thus, further analysis needs to be focused on these two independent variables.

Find Split and Force Split are the manual mode of Decision Tree analysis where the user has the freedom to explore the independent variable influence on the dependent variable “Income”. The user can explore different variable relations with the dependent variable by selecting **Next Split** on the toolbar menu or using keyboard command `ctrl + shift + T` and hover back to the previous relation with `ctrl + shift + H` until best or most significant relation is not identified.

---

**Fig. 12.21 Next Split option for exploring relations DV and IVs**
Fig. 12.21 shows the different relations explored between Dependent Variable (DV) and other Independent Variables (IVs) by using Next Split option. The Next Split command shows different relations sequentially, however, if the user wishes to jump to any specific variable to check the relationship with DV, the Go To Split command can be used. It can be opted through the toolbar menu or keyboard shortcut key ctrl + shift + O can be used. This Go To Split command gives flexibility to switch between different variables and corroborate on the analysed result.

The Find Split, Force Split and Next Split are the manual mode growing/expanding the decision tree of determining the relations between various Independent Variables with Dependent Variable. There is another advanced feature which Knowledge Studio provides to automatically grow the tree and examine all the relationships which are available with the imported data.

**Automatic Grow**

The Automatic Grow command allows to automatically grow a tree to the size specified in the automatic grow parameters. Each split on a node is the strongest statistical split for that node. The Automatic Grow can be started on any node. Any previous Find Split or Force Split results above the current node will remain intact in the tree. To use this command there must be at least one root node of a tree and the tree must be active. The attributes of the Tree variables can be modified in the Tree Attribute Editor at this step by using the Advanced button. The Automatic Grow command is called by using the selecting Grow | Automatic Grow from the menu. Alternatively, it can be called by the shortcut key Ctrl + Shift + A. Once the Automatic Grow command is called, a pop-up menu appears as shown in Fig. 12.22.
If the **Percentage option of Training data** is selected, the node size limits must be specified in percentage of the total. This option is selected by default and is recommended since it is logical to set these limits relative to the size of the input dataset. Select the option **# Records** if the requirement is to specify the node size limits in absolute numbers of records.

The **Non-Terminal Nodes** is the minimum number of records for a node to contain before the Find Split or Automatic Grow operations are allowed to split it. For the option "Percentage of training data", the default is 3%. The **Terminal Nodes** is the minimum number of records allowed for terminal nodes. For the option "Percentage of training data", the default is 1%. **Maximum Tree Depth** is the maximum number of levels allowed for the Automatic Grow command to grow the decision tree. The number of levels includes the root node, so the depth of a tree with a single split is 2. Enter the value of zero (0) for unlimited depth (in that case, the depth is limited only by the minimum node size constraints defined in this dialog). The default is 5.

After feeding the desired input to the Automatic Grow Pop-up window, click OK to generate the fully grown decision tree, as shown in Fig. 12.23, and each relation can be individually examined.
Fig. 12.23. Fully grown Decision Tree through Automatic Grow Command
While building the Decision Tree model, there are various Utility Tabs, as shown in Fig. 12.24, available in the Knowledge Studio platform for garnering the tree information. By default, Tree tab will be activated, but other options such as Tree Map, Node Data, Split Report, Node Report, Chart, and Profile chart can be handy to manage & control the Decision tree.

**Tree tab** is for decision trees which is the main analysis view. Initially, the tree view displays the root node, which represents all the records in the dataset. From there, records are split into groups called child nodes. These nodes, in turn, are further split until terminal nodes or leaves - which can no longer be split - are created.

**Tree Map** tab displays the tree without the detailed information for each node. This view is good for navigating through a large tree. All the functions can be performed that are available for the Tree view.

**Node Data** tab displays the data contained within the active tree node. The node is considered active when it is highlighted by a mouse click while in the tree view or tree map view. Data is listed in a worksheet-style window where the fields appear as columns and the records appear as rows. This information cannot be changed.

The **Split Report** tab is used for displaying all splits at the currently selected node of the decision tree. The default order is the order of the variable significance with respect to the dependent variable, depending on the measure. Although only the most statistically significant split is displayed in the tree, all statistically significant splits are calculated. The split report shows all the calculated splits in rank order. Variables with a significance less than the filter threshold will be displayed but will not have statistics computed. The **Node Report** tab provides the information about
the terminal nodes in the tree.

**Node Chart** tab helps to visualize the distributions of variables within a tree node. The chart look and functionality is exactly the same as in the Dataset Chart (for the Single Variable option) or Cross Tabs, except that it shows the variable distributions in a single tree node rather than the whole dataset. Refer to the Dataset Chart topic for details.

The **profile chart** shows the distribution of the dependent variable (DV) in the terminal nodes of the tree in the form of a bar chart. Each bar represents a terminal node in the tree, its width representing the size of the node. Profile Chart is completely interactive and synchronized with the Decision Tree view, and selection can be on the chart by left click to navigate the tree, right-click on any node to choose a command from the same pop-up menu as in the Decision Tree view, so any split operation such as Find Split can be performed. If a node is tagged in Profile Chart, it will be automatically tagged in the Decision Tree view, and vice versa.

After performing various types of analysis in the Decision Tree model and identifying the most significant variable that influences the Dependent variable (i.e. outcome), a decision can be made. But to know how good the model is built to predict the desired outcome, it is important to check how well the model is performing of the training data and score its performance.

**Step 6: Validate and score a model**

After creating a predictive model, it is reasonable to test the accuracy of its predictions before deploying it into production. This can be accomplished by applying the model to other datasets (different from the one the model was trained on) with known values of the dependent variable. Model predictions are then compared to these known values, and certain statistics are calculated to measure the model accuracy. This process is called validation.

As a standard modeling practice, prior to creating models, the dataset is split into two partitions: Training and Validation. The model is trained on the Training partition and then validated on the Validation partition. The validation process creates a new dataset that contains model predictions and comparison
fields. Creating some of these fields is optional. During the validation wizard setup, which fields to include need to be decided. The output of the validation process depends on whether the input models have a discrete or continuous dependent variable (DV).

Before initiating the model validation process, a model instance is required to be created. However, if not created before activating the validation node, it will be notified and automatically created after getting the approval. The steps to activate the Validation node is Right Click on the Studio Canvas | Evaluate | Model Validation. Then select the Decision Tree Node and drag the arrow to the Validation Node. If the model instance is missing then a pop-up window, as shown in Fig. 12.25, will open and consent is needed to create the model instance.

![Fig. 12.25 ‘Model Instance’ requirement for Validation](image)

After activating the Model Validation node and connecting with the model instance Decision Tree node the data set needs to be connected with the Validation node – to do so, Click on Training data and drag the arrow to connect it with the Validation node; shown in Fig. 12.26.
Thereafter, to define the validation parameters, double-click the Model Validation node or right-click on it and select Modify. A pop-up window will open where Target dataset name needs to be specified. The name of the input model, input dataset, and the name of the dependent variable are also displayed on the first page.

The next step provides the option of Field Mapping the field names in the model to the ones in the validation input dataset. In some situations, the names of model variables may not match the names of the corresponding fields in the input dataset, so changes need to be done manually on the map fields by clicking in the right column (Dataset Field Names) and selecting the right fields to match the model variables in the left column.

After Field Mapping, the Validation Field appears by where the inclusion of items needs to be made, Fig. 12.27. The validation fields produced by validating models with the discrete dependent variable (DV) are different from those produced by models with continuous DV. For example, for a Decision Tree model
with discrete DV, the scoring fields are: DV Prediction, DV Probability of Prediction, DV Value Probability, DV Prediction Correct, Node ID, and Node Number. Since the DV for the present task is considered as **Income**, thus the name appears as shown in Fig. 12.27.

![Fig. 12.27 Validation Field](image)

At the Field Selection step, choose the fields from the input dataset that need to be included in the validation output in addition to the new score/probability fields. The fields required by the model are highlighted in bold font and cannot be excluded. By default, all fields in the validation input dataset are selected to be included in the output.

After all the required details are selected/fed in the Validation pop-up window, Click Save to save the settings, or click Run to save the settings and run the validation process immediately. The output dataset node will be created in the workflow after running the validation node. It can be viewed and manipulated like any other dataset.

After successfully deploying the Model Validation, the detailed report can be generated by double click on the newly formed **income decision tree validation** node and check how well the built model performs. Fig. 12.28. shows the generated report of Model Validation for the Training Data. The top section of the Validation Report displays the name of the input dataset and the input model, as well as the validation date and time. The Confusion Matrix and the Statistics table show parameters characterizing the prediction accuracy of the model. The Confusion Matrix shows how many of the predictions produced by model validation were
correct and how many were wrong, i.e., did and did not coincide with actual DV values.

The validation for the Training data, Fig. 12.28, show that 83.32% of data were correctly Predicted. However, these data outcomes were already known, therefore the Model Validation should also be evaluated for the Testing data (30% of total data) which were created during Data Partitioning to actually know the performance of the built model.

To evaluate the model for Testing Data, Model Validation needs to be added again to the Studio Canvas, and connected with two pre-existing nodes, i.e. Testing Node and Model Instance of Decision Tree, as shown in Fig. 12.29. Similar modifications need to be done for newly added Model Validation which were followed earlier and the generated report, Fig. 12.30, needs to be evaluated.
In the Validation Report of the testing data shown above, the validated model has the dependent variable with values <=50k and >50k. The Confusion Matrix shows that the actual ‘<=50k’ value was correctly predicted in 3414 cases (91.09% of all values) and wrongly predicted as ‘>50k’ in 334 cases. The ‘>50k’ value was correctly predicted in 692 cases (60.92% of all values) and wrongly predicted as ‘<=50k’ in 444 cases. The overall correct prediction for the testing data came out
to be 84.07%, which shows the built model performs well in predicting the outcome.

The statistical details mentioned in the report can be understood as:

**Total Records:** The total number of input records.

**Correctly Predicted:** The number of records with correctly predicted target outcome. It should be noted that this depends on the cut-off value for discrete models and the valid hit range for continuous models.

**Percentage:** The percentage of records with correctly predicted target outcome.

**Valid Records:** This is the number of records for which the probabilities could be successfully calculated and therefore predictions were produced. Invalid records are the records in the input dataset for which the probabilities could not be calculated because the values in some fields were unknown to the model.

**K-L Divergence:** The Kullback-Leibler (K-L) divergence, or relative entropy, is a quantity which measures the difference between two probability distributions.

**Cross-Entropy:** Cross entropy between two probability distributions measures the overall difference between the two distributions. This measures how well a distribution approximates another distribution.

***Step 7: Evaluate model performance***

Model Analyzer allows one to extract information about the models that are created and then display the information in the form that can be used for analysis. Charts and statistics can be created for evaluating discrete and continuous models and these charts can be used to compare the performance of different models. It can also compare model performance using validated datasets (i.e., datasets obtained as a result of model validation). A model analyzer allows one to:

- evaluate how well the built model can classify the data
- validate how well the model works with different data
- compare different models to see which model provides the best lift.
Before using the model analyzer, a predictive model must be already created on a project. Once a model is added to the model analyzer, information is calculated and displayed in the form of charts. To compare several models in Model Analyzer, another model can be added at any time using the menu command Tools | Add Model. To remove a model from Model Analyzer, select Tools | Remove Model.

If you have a validated dataset (a dataset that is a result of validating a model), Model Analyzer can estimate the performance of the model on the basis of this dataset. The necessary condition is that the dataset must have a field with actual values of the variable of interest and the field with values predicted by the model (DV prediction field or Scorecard field). In Angoss, datasets with these fields are generated as a result of validating a model. However, the origin of these datasets does not really matter, it can be even used on datasets with prediction fields generated in other applications. The most important thing is the presence of the fields with predicted values to compare to actual values.
To add a Model Analyzer operation to your workflow, drag the **Model Analyzer** node from the **Evaluate palette** and connect the input dataset to it. Alternatively, **Drag** a connection arrow from the input dataset node to any area on the Workflow canvas background. From the context menu that appears when the mouse button is released, select **Evaluate | Model Analyzer** as in the Fig. 12.31. A new Model Analyzer node will be created, and the wizard will open. If the wizard does not open automatically, double-click on the Model Analyzer node or right-click on it and select Modify from the context menu.

![Fig. 12.31. Model Analyzer activation for evaluation of model](image)

In the Model Analyzed window, ‘**Dependent Variable**’ if the field that contains the values of the dependent variable of the validated model. ‘**Predicted Value**’ is the target value predicted by the model that produced the input validation dataset. And the ‘**Probability/Score**’ is the field that contains the probability of the predicted value or the scores of a scorecard.

In the present task, the Dependent Variable is **Income** That has two
categories, "<=50K" and ">50K". The Income Prediction field contains values predicted by the model (either "<=50K" or ">50K"). The probability of the model predicting the value "<=50" or ">50" is displayed in the fields Income <=50 Prob and Income >50K Prob, respectively. The Income Predict Prob field contains the probability of the predicted value (i.e., the probability of the value in the Income Prediction field). For the task, the Predicted Value of the dependent variable <= 50K is selected and RUN button is clicked for evaluation.

Once the Model Analyzer is successfully added to the Studio Canvas, a Green Check mark will appear on the node. Thereafter, to see the outcome, a double click on Model Analyzer will enable Model Analyzer View where various plots can be seen by activating different tabs, as highlighted in Fig. 12.32, to know the performance of the build model.

A Cumulative lift chart is a measurement tool that helps to analyze the effectiveness of the built model. This chart is used to determine if the model has any value or not. It also compares different models to determine which is the best and see how the model behaves on new data. The cumulative lift is based on the probabilities of the category of interest plotted against the percentage of the total
population. Fig. 12.32 shows that if approximately 50% of the population is targeted, then there is a probability of reaching the people with Income <=50K is 92.67%.

**Lift Chart** is a measure of the effectiveness of a model calculated as the ratio between the result obtained with the model and the result obtained without the model (i.e., by random record selection). The result obtained without a model is based on randomly selected records and is represented by the random curve in the lift chart (the horizontal line).

The x-axis of the Lift Chart shows the percentage of the total number of records sorted in descending order by the score, i.e., by the probability that the model assigns to a prediction of the selected value of the DV (the target value). At the points where the model curve is below the random line, Fig. 12.33, the lift factor is smaller than 1, which means that the records of the model contain fewer target values than a random sample with equally distributed target values.

![Random Line](image)

**Fig.12.33. Lift Chart of the Model Analyzer**

The **Cumulative Lift Report** tab shows the values of lift and cumulative lift at the decile level for each model in the Model Analyzer. If more than one model was added to the Model Analyzer, select the desired model label from the drop-down list at the top of the view.

The **Lift Report** tab presents the values of lift at the decile level for each model in the Model Analyzer. If more than one model was added to the Model Analyzer, select the desired model label from the drop-down list at the top of the view.
The **K-S chart** tab shows the maximum spread between two curves, where the first curve (the upper curve) represents the dependent variable (DV) category of interest and the second curve (the lower curve) represents the remaining dependent variable categories. The upper curve is the cumulative lift of the DV category of interest and is identical to what is displayed in the Cumulative Lift Chart view. The vertical line between the two curves graphically shows the point where the maximum difference, or spread, occurs. The calculated value at the point of maximum difference is the K-S statistic.

The **ROC curve** (Receiver Operating Characteristic), also referred to as Relative Operating Characteristic, allows to evaluate the model accuracy by estimating the sensitivity and specificity of a model in terms of the true positive rate and the false positive rate. The ROC curve is created by plotting the True Positive Rate (TPR, or Sensitivity, on the vertical axis) versus the False Positive Rate (FPR, or 1 − Specificity, on the horizontal axis) while the cutoff used to construct the Confusion Matrix is varied from 0 to 1. Both TPR and FPR vary within the interval [0,1].

The best case (a perfect classification) would occur for a point plotted at (0,1), i.e., the upper left corner, where the Sensitivity is the highest (=1) and the False Positive Rate is the lowest (=0). In general, it is better if a point is above the diagonal line (where the True Positive Rate is higher than the False Positive Rate) than below the diagonal line (where the False Positive Rate is higher). *The higher the curve is above the diagonal, the better the model is.* Fig. 12.34 shows the ROC of the evaluated model. The plot show that the curve is well above the diagonal line, and its Area Under Curve (AUC) is 0.8771 (i.e. 87.7%). This means the built model is predicting the outcomes very well and can be considered for implementing in actual application.
The GOF Statistic tab view of the Model Analyzer displays the results of the Hosmer-Lemeshow test for goodness of fit. It is applied to Logistic Regression models. The statistic, however, can also be calculated for decision trees and other models with binary dependent variables. The test compares the rates of the actual (observed) and predicted (estimated) target outcome in the subgroups of the validated dataset that are the deciles of the target outcome probabilities. The closer the predicted and observed outcome rates in subgroups, the better the model performance.

The Profit Curve is a view within the model analyzer that provides a quick Return-on-Investment (ROI) calculation based model performance (cumulative lift chart) and business specific costs and expected returns.

Step 8: Build a scorecard

For Decision Trees and Predictive Models, scoring is the process of predicting the values of the dependent variable (DV) for records in a new dataset. Scoring is usually done after completing the validation process, in which assessment is done about how well the model performed on test data. The score is the
probability that the predicted outcome will occur. In contrast to validation, in scoring the target dataset is assumed to have no DV field, the correct outcome is not known, so no comparison of actual vs. predicted outcome takes place.

During the scoring process, additional fields are created. These fields can be either part of a newly generated dataset, appended to the target dataset, or exported to an external file or database table. Not all of the generated fields are mandatory. During the scoring wizard setup, you choose which additional fields you want to include.

Scoring can be performed on all types of models - Decision Trees, Regression Models, Cluster Analysis, Scorecards, Market Basket Analysis, etc. For interactive models - Decision Trees, Strategy Trees, and Market Basket Analysis - scoring is applied to model instances since it is important to know what the state of the model was at the time of scoring.

To add a model scoring operation to the Studio Canvas of Workflow, Drag the Scoring node from the Action palette onto the Workflow canvas and connect the input model and dataset nodes to this node. If the model is a Decision Tree, a Strategy Tree, or a Market Basket Analysis process node (rather than a model instance node), it will prompt to create a model instance first. The alternate method of adding a Scoring node is to Right-click anywhere on the workflow canvas and from the context menu command select Action | Scoring, as shown in Fig. 12.35. This will create a new scoring node. Connect the Training dataset and Decision Tree model instance nodes with the Scoring node.
Step 9: Validate and deploy the model

To define the scoring parameters, double-click the Scoring node or right-click on it and select Modify. The required selections need to be made in the Scoring Pop-up window and **RUN** the analyzer. Fig. 12.36 shows the Scoring outcome report of the built model.

**Fig. 12.36 Scoring Report of the Model**
The top section of the Score Report displays basic information such as the name of the input dataset, the name of the input model, and the date scoring was performed. It also shows the total number of records provided as the scoring input ("Total Number of Scored Records") and the number of records for which a valid score was produced ("Valid Records"). The latter may be less than the former in the case where a required field in the scoring dataset has values that did not occur in the corresponding field of the training dataset (in this case, the produced score is a missing value).

**Step 9: Deploying the Built Model**

Using automatic code generation, models can be exported to other formats for deploying outside of the Altair Knowledge Works applications and for presentation purposes. The model deployment code in SQL, LOS (Language of SAS), R, Python, SPSS, Java, or PMML can be run in the corresponding analytic environments, databases or decision engines that can interpret any of these languages. Code generation nodes are provided in the Action palette.

To generate code for any model node in the workflow of the Studio Canvas, **Drag a connection arrow** from the Decision Tree model node to any area on the Workflow canvas background. Releasing the mouse button brings up a context menu. From the context menu, select **Action | Generate {code}**, where {code} is the type of code you want to generate, such as SQL, SAS, etc. A new Generate {code} node will be created and the wizard will open. Alternatively, it can also be activated by Drag the Generate {code} node, where {code} is the desired code type, from the Action palette onto the workflow canvas and connect the model to it.

Fig. 12.36 shows the generation of the Code in the form of Python Programming Language for the Decision Tree Model.
The developed model can also be deployed by uploading it to KnowledgeMANAGER or RealTimeScoring.

KnowledgeMANAGER™ is a secure web-based framework for model storage, comparison, monitoring and deployment. It provides visual dashboards, automated alerts for monitoring model performance and detecting performance deterioration, enable users to replace or update (retrain) the models, etc.

Fig. 12.36. Python Code generation for model deployment
KnowledgeMANAGER at requires a user account for it, where uploading of models to KnowledgeMANAGER is directly done from the project workflow. To upload a model, open Project Explorer and right-click on the model you wish to upload. From the context menu, select the command Upload to KnowledgeMANAGER, as shown in Fig. 12.37.

![Fig. 12.37. Uploading to KnowledgeMANAGER for model deployment](image-url)

Altair Real-Time Scoring Engine integrates with operational systems to provide the ability to deploy models and strategies in real-time and produce on-demand scores, decisions, or recommendations as well as apply real-time actions or treatments. This also requires a user account and uploading can be done by opening Project Explorer and Right-Click on the model you wish to upload. From the context menu, select the command Upload to RealTimeScoring.

This concludes the solution of the Task which started for developing a marketing strategy for income group of <=50k. After following all the 9-Steps for building the predictive Machine Learning, the Workflow Canvas appear as shown in Fig. 12.38. The similar approach can be made for creating other types of Predictive Models also; such as Logistic Regression, Cluster Analysis, Random Forest, Deep Learning, etc.
Fig. 12.38. Workflow nodes in Studio Canvas
13 Conclusion

Artificial intelligence, commonly termed as AI, is concerned with the design and development of intelligence in machines artificially. The primary goal of AI is to create systems that can work intelligently and independently. Intelligence is something that characterizes humans, and it is expected from an intelligent machine, that it mimics humans in the best possible manner for performing any task. To define intelligence, 4 main characteristics of humans are considered – Act Humanly, Think Humanly, Act Rational, and Think Rational.

The year 1950 was speculated to be the most significant year for the induction of Artificial Intelligence. The year 1956 was considered as the birth year of “AI”. During a conference at Dartmouth College, New Hampshire, the term Artificial Intelligence was first used by John McCarthy. Dr Alan created the famous “Turing Test” that is used to determine intelligence of a machine. The turing test states that “if a machine can perform tasks which is indistinguishable from human work performance, then the machine can be considered to be an Intelligent Thinking Machine”.

AI has immense potential to redefine future technology and has seen exponential growth in recent years compared to its inception year and has already been touching our lives in several ways. The main question is - if AI was present for over half a century, why has it gained so much importance in recent years? Why AI is the most talked about topic and what is the main reason it has made its insurgency in recent times? The most notable reason that could justify its insurgency is the rise of computation power. Artificial Intelligence requires a lot of computing power and the recent advances have made complex calculations much easier through advanced computers. Due to this the complex deep learning models can easily be deployed and processed by high-power GPUs.

To categorize a machine as intelligent, it is always be compared with human intelligence. Based on this, Artificial Intelligence is categorized into three different evolutionary – Narrow-Intelligence, General-Intelligence and Super-Intelligence.
Artificial Narrow-Intelligence is also known as a weak AI which involves the application of AI only to a single task. It lacks self-awareness and can perform a specific task exceptionally well and repeatedly for a substantial period of time. General-Intelligence is also known as “Strong AI”. A machine can be categorized under General-Intelligence if it can perform any task as intelligently as a human being can carry out. The Artificial Super-Intelligence term refers to those machines which have the ability to surpass human intelligence. This implies that if a machine has greater creativity, logical thinking, general wisdom and problem-solving ability than a human, it will be considered an Artificial Super-Intelligent machine. In reality, there is no such machine, and it will certainly take a while to achieve the Super-intelligence stage.

The field of Artificial Intelligence (AI) uses several terms such as machine learning (ML), deep learning (DL), and data science (DS). Although, they seem to be indistinguishable many a time due to their similarity in application and non-availability of any concrete definition which can clearly distinguish these terms are mutually inclusive, and it is imperative to understand how they differ from each other. Artificial Intelligence is the technology that is concerned with the automation of intelligent behaviour and attempts to build intelligent entities. Artificial intelligence can be considered as a broad umbrella under which machine learning and deep learning techniques work to achieve the objectives of AI. Machine learning is a subset of artificial intelligence where we teach a machine how to make decisions with the help of input data by using statistical tools. In other words, we can say that the method of training computers to learn patterns from a set of data, and is commonly used for making decisions or predictions is known as Machine Learning. Deep learning is a further subset of machine learning where intelligent algorithms try to mimic the human brain. It implies that the machine learns by imitating the human way of gaining knowledge. Data science is a multi-disciplinary field where scientific methods and algorithms are used to get insight from any unstructured data. Data science is related to Big data, which requires powerful hardware and computational systems along with efficient algorithms for finding solutions with the bulk of data. Data science can be considered mutually inclusive
with Artificial Intelligence and Machine learning, but the aspects of deep learning remain unhindered.

In recent times, artificial intelligence has been widely used in developing machines and robots that are efficiently utilized in various fields such as healthcare, marketing, business analytics, autonomous robots, etc. The field of application of AI is not constrained to any specific domain, and it can be used anywhere where good amount of data can be generated. Machine learning is predominantly used in Computer science and Information technology; vital applications - Facebook face recognition tags, YouTube & Netflix recommendations based on the usage pattern of the user, Email Spam and Malware Filtering, Virtual Personal Assistants, etc., are just a few among many. With advancement and exploration in the field of AI, it is now deemed fit for much wider application areas too such as the domin of Mechanical Engineering. The influence of AI and its implementation in automation with respect to the modern mechanical engineer is conceivably a hot topic in the industry. Some of the important facets of Mechanical Engineering where Artificial Intelligence presently plays a crucial role are – Product Design, Material Science engineering, Fault Diagnostics of mechanical system and Reverse Engineering.

Artificial Intelligence is considered to be a predictive tool in which statistical and mathematical models are created through various computer programming languages. A programming language is a way of representing an idea and communicating with the machines through algorithms. The selection of a language for programming a machine is mainly dependent on the affinity to the language a programmer is acquainted with. Some of the significant names in AI programming languages are Python, Java, MATLAB, R-programming, Lisp, Prolog, etc. Python is the most preferred and effective language for application in the field of artificial intelligence. The main reason behind this is, that the syntaxes in python are very simple and easy to learn by the majority. Apart from Python, there are a variety of programming languages which are being used by users in the field of AI.

The earliest known conventional definition of a machine learning states that “A computer program is said to learn from experience E with respect to some
class of task $T$ and performance measure $P$ if its performance at a task in $T$, as measured by $P$, improves with experience $E$. “Machine learning is a method to predict or classify the outcome of the future data through a computer model by recognizing a pattern from the past data set which gradually improves with experience. Machine learning involves creation of predictive mathematical models by using statistical tools and a programming language which can help a machine train on its data and predict specific outcomes based on this training. Machine Learning process comprises of 7 major steps – Data Acquisition, Curation, Exploration, Model Creation, Model training, Evaluation, and Prediction. All these 7 stages are followed in machine learning process for performing the predictive analysis and each stage has its own importance.

In Machine Learning, the term learning refers to the training of the algorithmic model by using previous data for determining a pattern in it. In general, there are three types of learning – Supervised Learning, Unsupervised Learning and Reinforced Learning. Supervised Learning is a method of training the machine learning model with the help of an input vector and its supervisory signals. The input vector refers to the dataset that acts as the input to the system whereas the supervisory signal is related to the output data. This learning deals with training a model with a dataset that is labelled, i.e. whose outcome is already known. The commonly used algorithms in supervised learning are Linear regression, Logistic regression, Linear discrimination analysis, Decision trees, Bayesian logic, Support vector machines (SVM), Random forest, etc. The main applications where supervised learning is used are – spam detection, handwriting recognition, speech recognition, computer vision, biometric attendance, weather predictions, etc. Unsupervised learning is a data-driven learning process where the outcome is dependent on the characteristics of data and considers the probability densities of the given input. Some of the widely used algorithms are: K-means, C-means, Hierarchical, Mixture Models, Gaussian Mixture, Hidden Markov Model, Principal Component Analysis (PCA), Linear Discrimination Analysis (LDA), etc. The Reinforced learning deals with training the model to take a sequence of decisions by the agent considered to be in a complex environment and a balance is tried to
achieve between the unknowns and already known details. In this method, the learner is known as an agent and everything outside of the agent is considered as the environment. The agent interacts with the environment continuously by taking certain action steps and the environment responses to these actions by giving a new situation to the agent. The environment also provides a reward to the agent that is a step close to correct action and the agent tries to maximize this reward over time. The common types of algorithm that are used in this learning method are Monte Carlo, Q Learning, Deep Q Network (DQN), Proximal Policy Optimization (PPO), SARSA, etc.

The machine learning technology makes a machine smart and intelligent and gradually society is becoming more reliant on these intelligent machines that ease human’s effort. Despite having so many success stories and its revolutionizing capabilities, machine learning still has several limitations and required substantial efforts to overcome these. The main limitation is the data requirement. If Machine Learning is expected to perform like a human, an enormous amount of data is needed to train the model. Since data acquisition is not easy, the applications of machine learning are limited. Data Quality Dependency also affects the Machine Learning process. The poor-quality data will always result in an erroneous and inaccurate outcome. Another limitation of Machine Learning is that there is always a possibility of data biases. The machine learning dataset will never have parity among all variables. Some predictors will have dominating features compared to others which will have a greater influence on the target variable. Thus, the predicted outcome will always be biased.

Deep Learning, a subset of Machine Learning, also known as Deep Structured Learning or Deep Neural Network uses networks of artificial neurons for extracting high-level insightful features from raw input data. Due to the usage of artificial neurons for processing of information and distributed communication nodes, deep learning is also widely known as Artificial Neural Network (ANN). Although ANN is designed to mimic the human brain which is dynamic and analog, the erstwhile tends to be static and symbolic. The Artificial Neural Network is a mathematical model used for computation that simplifies and simulates the
working of the human brain. This model uses a large network of neurons positioned as layers and inter-connected with some weighted function. In the neural network, computation in neurons is based on bias, weight and activation function. The activation function in the network is a mathematical gate between the current input neuron and outgoing next neurons. The type of activation function decides whether the present neuron should be activated or not, and is dependent on the weighted sum involved with the neurons. Compared to Machine Learning, Deep learning requires significant computational power and dedicated GPUs for performing the desired task, this is attributed to the computation involved in multi-layered neurons and their correlations.

Knowledge Works software, a trademarked technology of Altair Engineering, is desktop-based predictive analytics and machine learning solutions that will help you quickly generate actionable insight from the data. Knowledge Works data intelligence solutions allow individuals and organizations to incorporate more data, unite more minds with agility, and engender more trust in analytics and data science. The software enables users to quickly and accurately capture and prepare data for any project, use data to accurately predict outcomes and produce insight and foresight, and visualize trends and insights that help to decipher and communicate across the business. It also provides Visual, ML-driven forecasting that autogenerates execution code for advanced data science models as well as Embedded visualization for easy interpretation of models. The Knowledge Works software platform also consists of Augmented analytics in which Machine learning provides recommendations for data prep and relevant data sources and determines the trustworthiness of data assets. There are a variety of tools in the Knowledge Works platform such as - Altair Knowledge Studio, Altair Knowledge Studio for Apache Spark, Altair Knowledge Seeker and Altair Monarch.

Knowledge Studio is a data mining and predictive analytics software tool which enables advanced analysts to build and deploy predictive models using repeatable workflows. Knowledge Studio features cover all stages of the predictive analytics life cycle, from data preparation and profiling, variable selection, decision tree analysis, building predictive models, cluster analysis, principal component
analysis, market basket analysis, building scorecards, applying reject inference techniques, model performance evaluation, building and deploying strategies.

Knowledge Studio for Apache Spark is a comprehensive data science platform integrated with Apache Spark technology to provide advanced data mining and predictive analytics on large-scale distributed data structures such as Hadoop HDFS, Amazon S3, and other storage types supported by Spark. Data can be loaded directly from Hadoop HDFS and ViewFs, Hadoop Archive, Amazon S3, FTP and Network Shares.

Knowledge Seeker is a data mining and predictive analytics software tool used by business analysts and advanced data scientists for data exploration, decision tree analysis, predictive modeling with decision trees, and strategy development. Knowledge Seeker capabilities include data preparation and profiling with advanced visualization, decision trees, strategy design, model performance evaluation, and deployment of decision tree models and strategies.

Monarch is a desktop-based self-service data preparation solution. Monarch connects to multiple data sources including structured and unstructured data, cloud-based data, and big data. Connecting to data, cleansing and manipulating data requires no coding. Monarch can quickly convert disparate data formats into rows and columns for use in data analytics. Over 80+ pre-built data preparation functions mean data preparation tasks can be completed quickly and error-free. More time is spent on generating value from data as opposed to making data usable to begin with. Monarch is a self-service data preparation tool which eliminates the need for IT to extract data, that is often considered to be very time consuming and labour-intensive. Automated repeatable processes with reusable models and workspaces allow workers to spend their time analyzing data instead of preparing it for analytic needs.
14 Reference

Ref. [1]

Fig. 1.3 (a) A photo of a conversation with the ELIZA chatbot, [https://en.wikipedia.org/wiki/File:ELIZA_conversation.jpg](https://en.wikipedia.org/wiki/File:ELIZA_conversation.jpg). This file is licensed under the public domain and is ineligible for copyright.

Fig. 1.3 (b) A photo of Kasparov-29 by Owen Williams, [https://en.wikipedia.org/wiki/File:Kasparov-29.jpg](https://en.wikipedia.org/wiki/File:Kasparov-29.jpg). This file is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported license.

A photo of Deep Blue by James the photographer, [https://commons.wikimedia.org/wiki/File:Deep_Blue.jpg](https://commons.wikimedia.org/wiki/File:Deep_Blue.jpg). This file is licensed under the Creative Commons Attribution 2.0 Generic license.

Fig. 1.3 (c) A photo of Stanley2, [https://commons.wikimedia.org/wiki/File:Stanley2.JPG](https://commons.wikimedia.org/wiki/File:Stanley2.JPG). This file is licensed under the public domain and is ineligible for copyright.

Fig. 1.3 (d) A photo of Honda ASIMO (ver. 2011) 2011 Tokyo Motor Show, by Morio. [https://commons.wikimedia.org/wiki/File:Honda_ASIMO_(ver._2011)_2011_Tokyo_Motor_Show.jpg](https://commons.wikimedia.org/wiki/File:Honda_ASIMO_(ver._2011)_2011_Tokyo_Motor_Show.jpg). This file is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported license.

Fig. 1.3 (e) A photo of IBM Watson w Jeopardy, by Atomic Taco, [https://commons.wikimedia.org/wiki/File:IBM_Watson_w_Jeopardy.jpg](https://commons.wikimedia.org/wiki/File:IBM_Watson_w_Jeopardy.jpg). This file is licensed under the Creative Commons Attribution-Share Alike 2.0 Generic license.

Fig. 1.3 (f) A photo of Sophia at the AI for Good Global Summit 2018, by ITU Pictures from Geneva, Switzerland, [https://en.wikipedia.org/wiki/File:Sophia_at_the.AI_for_Good_Global_Summit_2018_(27254369347)_cropped.jpg](https://en.wikipedia.org/wiki/File:Sophia_at_the.AI_for_Good_Global_Summit_2018_(27254369347)_cropped.jpg). This file is licensed under the Creative Commons Attribution 2.0 Generic license.

Ref. [2]

Fig. 7.3. A photo of EM-Gaussian-data, by Chire, [https://commons.wikimedia.org/w/index.php?curid=17085713](https://commons.wikimedia.org/w/index.php?curid=17085713). This file is
licensed under the Creative Commons Attribution-Share Alike 3.0 Unported license.

Ref. [3]

Fig. 9.1. A photo of Blausen 0657 Multipolar Neuron by Bruce Blaus, https://en.wikipedia.org/wiki/File:Blausen_0657_MultipolarNeuron.png This file is licensed under the Creative Commons Attribution 3.0 Unported license.

Ref. [4]

Fig. 9.4. A photo of Neural network bottleneck architecture, https://commons.wikimedia.org/wiki/File:Neural_network_bottleneck_architecture.svg. This file is licensed under the public domain and is ineligible for copyright.

Ref. [5]

Fig. 9.5. A photo of Svm separating hyperplanes, https://commons.wikimedia.org/wiki/File:Svm_separating_hyperplanes_(SVG).svg. This file is licensed under the public domain and is ineligible for copyright.

Ref. [6]

Fig. 9.10. A photo of Detected-with-YOLO--Schreibtisch-mit-Objekten, by M Theiler https://commons.wikimedia.org/wiki/File:Detected-with-YOLO--Schreibtisch-mit-Objekten.jpg. This file is licensed under the Creative Commons Attribution-Share Alike 4.0 International license.