

# Let's Try **Machine Learning & Artificial Intelligence** Using Open Source

A book for DIYers who want to try out leading open source tools and platforms for artificial intelligence and machine learning

Price  
₹ 300

# Contents

## Part-I: Understading AI & ML

<b>Chapter 1</b>	Machine Learning Basics for Newbies.....	5
<b>Chapter 2</b>	A Beginner's Guide to Machine Learning.....	8
<b>Chapter 3</b>	The Latest in AI and Its Applications.....	14
<b>Chapter 4</b>	The Top Twelve Open Source Artificial Intelligence Tools.....	22
<b>Chapter 5</b>	AI for Industries: Using Intelligence in Real Time.....	29
<b>Chapter 6</b>	The Role of Open Source Software and Artificial Intelligence in Defence Systems.....	36
<b>Chapter 7</b>	Machine Learning and Smartphones: A Powerful Combination.....	42
<b>Chapter 8</b>	Potential Risks of Artificial Intelligence .....	49
<b>Chapter 9</b>	Machine Learning (ML) and IoT can Work Together to Improve Lives.....	52
<b>Chapter 10</b>	Applying Machine Learning to IoT.....	58
<b>Chapter 11</b>	Machine Learning API Libraries that are Helping the IoT.....	68
<b>Chapter 12</b>	Shaping a Smarter World with AI, Machine Learning and GIS.....	71
<b>Chapter 13</b>	No More a Buzzword, AI has Arrived and Reached Human Parity.....	77
<b>Chapter 14</b>	"We Are Embedding AI Into All of our cloud services" .....	80
<b>Chapter 15</b>	The Future of Machine Learning.....	89

## **Part-II: Implementing And Developing AI & ML**

<b>Chapter 1</b>	Machines Learn in Many Different Ways.....	96
<b>Chapter 2</b>	Tools that Accelerate a Newbie's Understanding of Machine Learning.....	103
<b>Chapter 3</b>	Why Python is Ideal for Machine Learning. ....	115
<b>Chapter 4</b>	Machine Learning Libraries that can Make Web Applications Smarter.....	125
<b>Chapter 5</b>	What's Good About TensorFlow 2.0? .....	133
<b>Chapter 6</b>	How TensorFlow Makes Machines Learn .....	145
<b>Chapter 7</b>	The Capabilities of Tensor Virtual Machine, an Open Deep Learning Compiler Stack .....	148
<b>Chapter 8</b>	An Introduction to TensorFlow Programming in Python. ....	157
<b>Chapter 9</b>	Building Deep Learning Models with TensorFlow.....	168
<b>Chapter 10</b>	TensorFlow Hub: A Machine Learning Ecosystem.....	176
<b>Chapter 11</b>	How TensorFlow.js Defines, Trains and Runs Machine Learning Models.....	180
<b>Chapter 12</b>	How TensorFlow Can Be Used for Video Analytics.....	188
<b>Chapter 13</b>	Why the Apache Mahout Framework is So Popular. ....	195
<b>Chapter 14</b>	Apache Mahout The Recommender System for Big Data.....	198
<b>Chapter 15</b>	Implementing Scalable and High Performance ML Algorithms Using Apache Mahout.. .....	203
<b>Chapter 16</b>	Apache SystemML A Machine Learning Platform suited for Big Data.....	210
<b>Chapter 17</b>	H2O The Versatile Tools for Deep Learning.....	219
<b>Chapter 18</b>	ONNX: Helping Developers Choose the Right Framework.. .....	229
<b>Chapter 19</b>	Weka: A Free and Open Source Suite for Machine Learning and Deep Learning Algorithms, .....	239
<b>Chapter 20</b>	ML.NET: The New Open Source ML Framework from Microsoft .....	251
<b>Chapter 21</b>	Get Started with ML.NET, the Open Source and Cross-platform Machine Learning Framework for .NET.....	260
<b>Chapter 22</b>	An Introduction to Deeplearning4j, the Distributed Deep Learning Library....	273
<b>Chapter 23</b>	DeepLearning4j and PyTorch: Two Powerful Deep Learning Tools.....	280
<b>Chapter 24</b>	The Best Machine Learning Libraries in Julia.....	289
<b>Chapter 25</b>	Exploring Clustering Methods in Machine Learning.. .....	297
<b>Chapter 26</b>	Using OpenCV for ML in Real-time Computer Vision and Image Processing..	309
<b>Chapter 27</b>	The Python Libraries that are Made for Machine Learning.....	317
<b>Chapter 28</b>	Machine Learning: Building aPredictive Model with Scikit-learn. .....	323
<b>Chapter 29</b>	MLDB: The Open Source for the Cloud and for Docker.....	333

**Part-I**

# **Understanding AI & ML**

# Machine Learning Basics for Newbies

*According to Wikipedia, 'machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to effectively perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence.' This article introduces novices to an exciting field.*

Arthur Samuel is the founder of machine learning. He describes it as "a field of study that gives the ability to the computer to self-learn without being explicitly programmed." This means permeating knowledge to modern day machines without coding them. Machine learning (ML) focuses on the creation of computer programs that can develop, grow and change whenever they are needed to. Machine learning uses different algorithms for the purpose of self-learning.

## The need for machine learning

As we know, people worldwide use different technologies, which generate a huge amount of data every day; it is not easy to manage and access this data. More than 80 per cent of this immense amount of data is not structured and includes documents, audio, video, etc. It is an impossible task for humans to manage this humongous data. This is where ML comes into action, to reduce the time required to process this massive amount of data.

## Classification of machine learning

Machine learning can be classified into three categories.

**Supervised learning:** This is the primary kind of ML, in which marked information is used to prepare the algorithms. In supervised learning, algorithms are prepared utilising checked information, where the information and the yield are known. We input the information in the learning algorithm as a lot of data sources, which are called features, indicated by X alongside the related yields, which are

demonstrated by Y. The algorithm learns by contrasting its genuine generation and right yields to discover mistakes. The crude information is partitioned into two sections. The initial segment is for preparing the algorithm, and the other locale is used to test the prepared algorithm.

Supervised learning utilises the information examples to anticipate the estimations of extra information for the names. This technique is regularly used in applications where authentic information foresees likely outcomes.

**Unsupervised learning:** This is the second sort of ML, in which unlabelled information is used to prepare the calculations, which implies that it is utilised against information that has no verifiable names. What is being demonstrated must be made sense of by the calculations. The intention is to investigate the information and discover some structure within it. In unsupervised learning the information is unlabelled, and crude data is contributed in a straightforward way to the calculation without pre-preparing the information and without knowing the yield of the information. Also, the data cannot be partitioned into training or testing information. The calculation makes sense of the information and, as indicated by the information portions, it puts together bunches of information with new marks.

This learning method functions admirably on value based information. For instance, it can distinguish clients with comparable properties, which can then together be promoted with similar marketing efforts. Or, on the other hand, it can locate the essential characteristics of different client sections. These calculations can be used to section content points, prescribe things and recognise information exceptions.

**Reinforced learning:** This is the third kind of machine learning in which no crude information is given as contribution; rather, support learning calculation is needed to make sense of the circumstance by itself. The reinforcement adapting every now and again is used for mechanical autonomy, gaming, and navigation. With reinforcement learning, through experimentation and calculation, it is

possible to find which activities yield the biggest prizes. This sort of preparation has three primary parts — the ‘specialist’ which can be depicted as the student or leader, the ‘earth’ which is portrayed as everything the operator connects with, and ‘activities’ which inform what the operator can do.

The goal is for the specialist to take up activities that expand the normal reward over a given proportion of time. The operator will achieve the objective much better by following a decent strategy. So the motivation behind support learning is to gain proficiency with the best arrangement.

## **Applications of machine learning**

Machine learning can be used in various fields such as medicine, defence, technology, finance, security, etc. In these fields, different applications of supervised, unsupervised and reinforcement learning can be used.

On a concluding note, ML is a great field of information technology, and can bring about drastic changes in the way we live.

# A Beginner's Guide to Machine Learning

*As the title suggests, this article is for newbies who want to venture into the field of machine learning.*

Machine learning (ML) is an application of artificial intelligence that provides a system the ability to learn and improve from experience without being explicitly programmed. It has become very popular because of the high volume of data produced by applications, the increase in computational power in the past few years, and the development of better algorithms.

Machine learning is used in various domains—from automating tedious tasks to offering intelligent insights, industries in every sector try to benefit from it. You may already be using a device that has ML in it—for example, a wearable fitness tracker like Fitbit, or an intelligent home assistant like Google Home.

## Key terms

Machine learning terms can be confusing. Here are definitions of key terms to help you.

**Data exploration** is the process of gathering information about a large and often unstructured data set in order to find characteristics for focused analysis.

**Data mining** refers to automated data exploration.

**Descriptive analytics** is the process of analysing a data set in order to summarise what happened. The vast majority of business analytics—such as sales reports, Web metrics, and social networks analysis—are descriptive.

**Predictive analytics** is the process of building models from historical or current data in order to forecast future outcomes.

**Supervised and unsupervised learning:** We will explore these later in the article

**Model training and evaluation:** A machine learning model is an abstraction of the question you are trying to answer or the outcome you want to predict. Models are trained and evaluated from existing data.

## Training data

When you train a model from data, you use a known data set and make adjustments to the model based on the data characteristics to get the most accurate answer. In Azure Machine Learning, a model is built from an algorithm module that processes training data and functional modules, such as a scoring module.

In supervised learning, if you're training a fraud detection model, you use a set of transactions that are labelled as either fraudulent or valid. You split your data set randomly, and use a part to train the model and a part to test or evaluate the model.

## Evaluation data

Once you have a trained model, evaluate the model using the remaining test data. You use data you already know the outcomes for, so that you can tell whether your model predicts accurately.

## Other common machine learning terms

- **Algorithm:** A self-contained set of rules used to solve problems through data processing, math, or automated reasoning.
- **Anomaly detection:** A model that flags unusual events or values and helps you discover problems—for example, credit card fraud detection looks for unusual purchases.
- **Categorical data:** This is data that is organised by categories and can be divided into groups. For example, a categorical data set for autos could specify the year of manufacture, the make, model, and price.
- **Classification:** This is a model for organising data points into categories based on a data set, for which category groupings are already known.
- **Feature engineering:** This is the process of extracting or selecting

features related to a data set in order to enhance it and improve outcomes. For instance, airfare data could be enhanced by days of the week and holidays.

- **Module:** This is a functional part in a Machine Learning Studio model, such as the Enter Data module that enables entering and editing small data sets. An algorithm is also a type of module in Machine Learning Studio.
- **Model:** A supervised learning model is the product of a machine learning experiment comprising training data, an algorithm module, and functional modules, such as a Score Model module.
- **Numerical data:** This is data that has meaning as measurements (continuous data) or counts (discrete data). It is also referred to as quantitative data.
- **Partition:** This is the method by which you divide data into samples (see *Partition* and *Sample* for more information).
- **Prediction:** A prediction is a forecast of a value or values from a machine learning model. You might also see the term 'predicted score'. However, predicted scores are not the final output of a model. An evaluation of the model follows the score.
- **Regression:** This is a model for predicting a value based on independent variables, such as predicting the price of a car based on its year and make.
- **Score:** This is a predicted value generated from a trained classification or regression model, using the Score Model module in Machine Learning Studio. Classification models also return a score for the probability of the predicted value. Once you've generated scores from a model, you can evaluate the model's accuracy using the Evaluate Model module.
- **Sample:** This is a part of a data set intended to be representative of the whole. Samples can be selected randomly or based on specific features of the data set.

## What is machine learning?

According to Arthur Samuel, machine learning algorithms enable computers to learn from data, and even improve themselves, without being explicitly programmed.

The basic principle of machine learning is to build algorithms that can receive input data, and use statistical analysis to predict an output while updating outputs as new data becomes available.

## **Types of machine learning**

Machine learning can be classified into three types of algorithms:

1. Supervised learning
2. Unsupervised learning
3. Reinforcement learning

## **Supervised learning**

Supervised learning, as the name suggests, involves the presence of a supervisor as trainer. Basically, it is a learning process, with which we train or teach the machine using data that is well structured, i.e., data is tagged with correct answers. After that, the machine is provided with new sets of examples (data) so that the supervised learning algorithm analyses the training data (set of training examples) and produces a correct outcome from the structured data.

As an example, consider we have a basket full of different vegetables. The first step is to train the machine about all the different vegetables, one after the other:

- If the shape is cylindrical and the colour is brown, then it will be labelled as a potato.
- If the shape is like a cylinder and the colour is red or orange-green, then it will be labelled as a tomato.

The other vegetables are also categorised likewise.

Now, let's suppose that after training the data, you give a separate vegetable, say, a tomato, and ask that it be identified. Since the machine has already learnt some things from previous data, this time it will have to use what it has learnt, sensibly. It will first categorise the vegetable on the basis of its shape

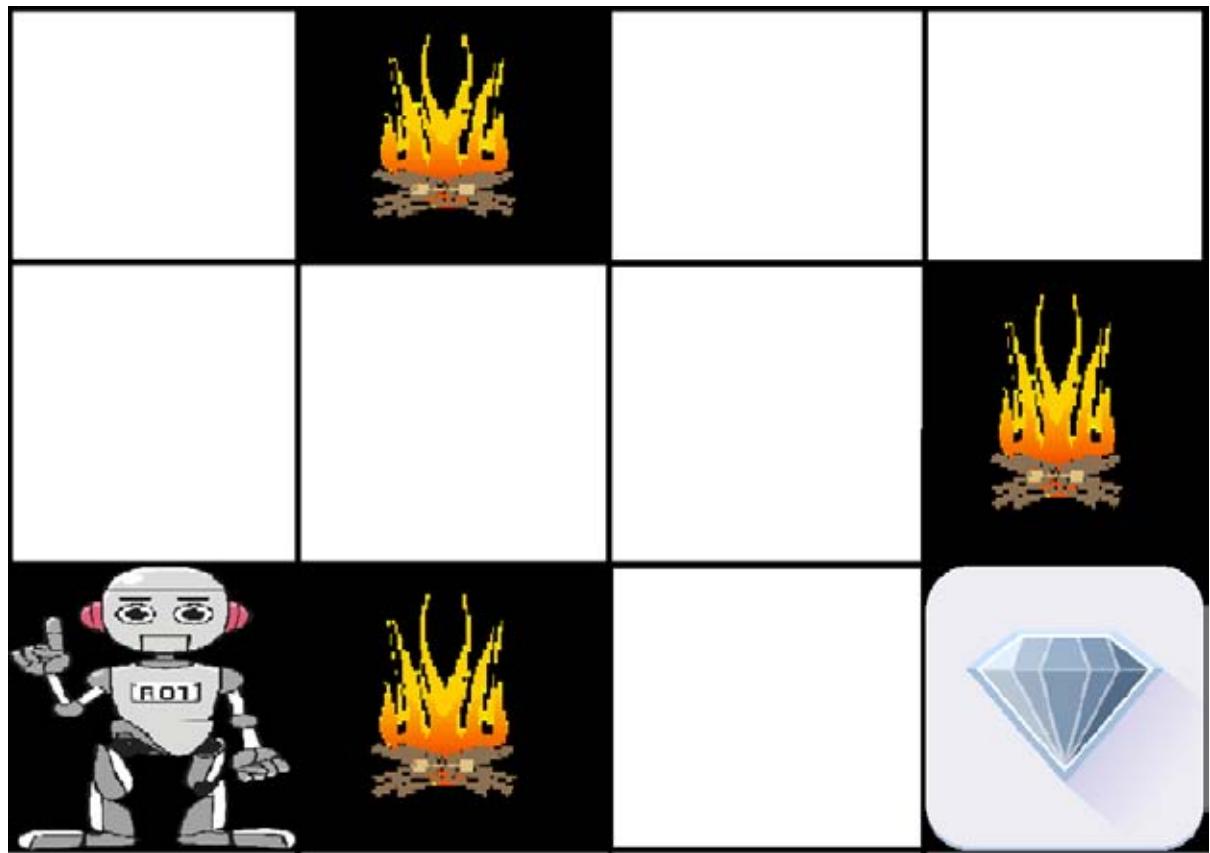


Figure 1: Reinforcement learning example

and colour, and will validate the vegetable's name as 'Tomato', putting it in the 'Tomato' category. Thus the machine learns certain things from the training data (the basket containing vegetables) and then applies that knowledge to the test data (a new vegetable).

## Unsupervised learning

This is the training of the machine using information that is neither classified nor structured, and allowing the algorithm to act on the items without supervision. Here the task of the machine is to group unsorted information according to similarities, patterns and differences without any prior training data. As the name suggests, no training is provided to the machine in this case. Therefore, the machine has to learn by itself from unstructured data. As an example, you may want to find groupings of customer demographics with similar buying habits.

## Reinforcement learning

This is an area of machine learning that is related to taking suitable action to maximise rewards in a particular situation, i.e., it is used to find the best possible path for particular circumstances. Reinforcement learning differs from supervised learning in that, in the latter case, the training data has the answer key with it; so the model is trained with the correct answer itself. However, in reinforcement learning, there is no answer key, and the reinforcement agent decides what to do to perform the given task. In the absence of a training data set, the machine learns from its experience (Figure 1).

We have an agent and a reward, with many hurdles in between. The agent is supposed to find the best possible path to reach the reward. The following example explains this. The objective of a robot is to get the reward, which is a diamond, and avoid the hurdles, which is fire. The robot learns by trying all the possible paths and then choosing the one that gives him the reward with the least hurdles. Each right step will give the robot a reward and each wrong step will take away a reward from it. The total reward will be calculated when it reaches the final reward, which is the diamond.

This article briefly touches on the fundamental concepts of machine learning. I hope it was helpful and will encourage you enough to delve into the topic.

# The Latest in AI and Its Applications

*AI is becoming a disruptive force that is redefining the modern industry. This article features some exciting applications of AI, along with a glimpse into the future, illustrating how AI will continue to transform industries and our lives.*

Sophia, an artificial intelligence (AI) humanoid, was in the news recently for becoming the first robot ever to have a nationality. In October 2017, Sophia was granted Saudi Arabian citizenship. This AI-powered robot is famous for speaking at the United Nations, and interviewing celebrities and world leaders.

Sophia, developed by Hanson Robotics, is an example of the most sophisticated AI-powered robots built by humans in recent times. It can imitate human gestures, facial expressions, and make conversations in the form of answering certain questions and initiating discussions on predefined topics.

AI was founded as an academic discipline in 1956. Since then, AI techniques have become an essential part of the technology industry. Different types of AI-powered robots are being developed in different parts of the world, including the US, China, Japan, Korea and India. As per reports, two-thirds of global investments in AI poured into China. This led to the AI industry grow 67 per cent last year alone.

China developed its first human-like female robot called Jia Jia in 2016, at its University of Science and Technology. Then, there is Erica, a Japanese female robot created by Hiroshi Ishiguro Laboratories, who is considered the most beautiful robot in the world.

## AI technologies

Latest AI technologies include natural language generation, speech recognition, virtual agents, machine learning, deep learning, biometrics and AI-optimised hardware. AI experts break AI down into three broad categories: artificial

narrow intelligence (ANI), artificial general intelligence (AGI) and artificial super intelligence (ASI).

Sophia uses AI, visual data processing and facial recognition technology. Complex and intelligent algorithms and good sensory systems could make AI robots perform even better. With improved machine learning and deep learning algorithms, future robots could be much more efficient, powerful and smarter than the present ones.

## **Types of AI applications**

There are many emerging applications of AI. You can find AI in robotics, healthcare, education, businesses and on your mobile devices, to name a few. We have narrowed down the list of AI applications to a few, each accompanied with a glimpse into the future, illustrating how AI will continue to transform industries and our lives.

***Virtual assistants:*** These are basically software agents that provide a wide variety of services. Amazon Alexa, Google Assistant and Siri are some of the most popular AI assistants. Many virtual assistant software are now being installed in smartphones as well to serve you in a better way.

***Internet applications:*** AI finds many useful applications in Internet-related technologies, such as digital marketing, creating and generating online content, digital advertising, Web searches, Web designs, chatbots, the Internet of Things (IoT) and others.

***Digital marketing:*** This field has revolutionised modern businesses. While the amount of information on potential consumers grows, AI-related technology will be of utmost importance when making data-based decisions. AI helps find people and customers based on their interests, demographics and other aspects to learn and detect the best audience for particular brands.

***Content creation:*** There are areas where content created by AI can be useful

and help attract visitors to a website. AI can also write reports and news based on data and information. Hundreds of articles can be created with AI technology quickly, which can save a lot of time and resources.

*Online searches:* Old ways of performing online searches no longer stay true. Two use cases using AI that have revolutionised Internet searches and search engine optimisation (SEO) are voice search and Google's algorithm called RankBrain. These have changed the way marketeers create and optimise their Web content.

*Web designing:* With AI, websites could exist without the help of programmers and designers. Applications, such as Grid, use AI to design websites based on the information provided by users like images, text, calls-to-action, etc. AI can make websites look professional in very little time and at a much lower cost.

*Chatbots:* Consumers are already using chatbots to chat with friends and colleagues without waiting for a long time for a response. Chatbots automate responses to potential buyers' frequently asked questions and provide them a way to search for the product or service they are looking for.

Natural learning processing and machine learning techniques are used by these bots to find the correct responses. Many brands have started using these techniques to communicate with their prospective customers through messenger applications like Facebook Messenger, WhatsApp and Slack.

*Cybersecurity:* The main concern in today's digital world is cybersecurity. Malware and virus attacks are common in the cyber world. There is a constant threat of data security to not just individuals or corporates but also government sectors. AI along with machine learning is used for the protection of data. It allows you to automate the detection of threat and combat without the involvement of humans. AI has been used for password protection and authenticity detection.

*The IoT:* AI is used to manage huge data flows and storage in the IoT network.

With high-speed Internet networks and advanced sensors integrated into microcontrollers (MCUs), AI along with the IoT is creating a new wave of disruptive technologies.

With the explosion of the IoT, there are problems regarding data storage, delay, channel limitation and congestion in networks. One solution to solve these is to use AI in data mining, managing and controlling the congestion in networks. Techniques used in AI include fuzzy logic and neural networks in conjunction with the IoT network.

**Finance and economics:** Wall Street, financial district of the US, uses complex computer programs to do heavy jobs. These programs run on their own. A few years ago, stock markets plummeted, taking down a trillion dollars worth of market value, due to a malfunctioning ANI program.

Also, for example, when you deposit a cheque using your mobile banking app, it runs through a refined ANI system that can read the cheque much faster than humans. When you shop online, you are essentially feeding data into ANI systems.

**Art and design:** AI has been used to algorithmically generate objects that can be rendered digitally. It can generate new patterns with high speed, good efficiency and verisimilitude. Algorithm-driven design tools help you construct user interfaces, content and personalise user experiences.

Publishing tools such as Readymag and Squarespace have greatly simplified the work to the extent where you can get many high-quality templates and designs without having to pay for a designer. There are many other algorithm-driven design tools for graphic design including identity, drawing and illustrations.

AI solution providers offer tools and libraries to manipulate images and photos. Soon, AI will drive the next generation of apps for visual arts and creative designs.

An AI model, called CRAFT (Composition, Retrieval and Fusion Network), is being developed by researchers from University of Illinois, USA, and Allen Institute for Artificial Intelligence, USA. It can convert provided text descriptions into video clips of an animated series. To simplify, AI matches videos with word descriptions, builds a set of parameters and generates scenes.

**Exploration and research:** Defense Advanced Research Projects Agency (DARPA), USA, is working on its Artificial Intelligence Exploration (AIE) programme, which is a key component of the agency's broader AI programme.

AI in space exploration is gathering momentum, too. Over the next few years, new missions would be taken up by AI as we voyage to the Moon and the planets, and explore possibilities in space. AI is also being used in NASA's next Rover mission to planet Mars.

National Geographic Society and Microsoft are partnering to explore how AI can help us understand, engage and protect Earth.

AI is helping the oil and gas industry to preserve the ecosystem while discovering new resources.

Robotics and AI have been replacing traditional research and exploration methods in ocean science and technology.

**Education:** AI is being used to improve education systems. Traditional techniques might be replaced by personalised, adaptive learning to tailor individual students' strengths and weaknesses. Machine learning can be used to identify students and focus extra resources on weaker ones.

AI-based robo-readers are being used in essay grading in schools. The approach involves pairing human intelligence with AI to improve the overall grading system and help students accomplish more.

**Automotive:** ANI-assisted automotive technology is being employed in driverless cars. Recently, IBM developed an IoT for automotive—a program to eliminate driver errors through connectivity. Since many accidents are caused by human errors, researchers are trying to find ways to minimise human errors using AI algorithms.

In the future, your car is likely to have well-packaged complex computer programs. An automated vehicle uses AI, sensors and global positioning system coordinates to drive itself without a human operator.

**Video games:** You might already know about Deep Blue, IBM's chess-playing supercomputer, which beat international grandmaster Garry Kasparov in the late 1990s.

Chinook, a program developed at University of Alberta, Canada, can beat any human player at the game of checkers. There is also a computer program called Maven for scrabble game. These are perfect examples of AI.

More recently, AlphaZero, an ANI developed by DeepMind, won 100 games in a row against the world's current best chess program.

Almost every modern video game has an AI component. Video game reviews are based on the quality of AI.

**Healthcare:** A new type of AI algorithm is being used by Google's Medical Brain to make predictions about the likelihood of death among hospital patients. This technology is the latest attempt to revolutionise healthcare. AI programs are being developed and used in diagnoses, treatments, drug development, and patient monitoring and care.

Today, most doctors use ANI programs. These assist doctors in accurately diagnosing cancer and various other diseases. Then, there is a robot chemist that uses machine learning to study new molecules and reactions.

As per a report by MarketsandMarkets, the global market for AI robots is expected to reach US\$ 12.36 billion by 2023, growing at a CAGR of 28.78 per cent during the forecast period. Adoption of AI-powered robots for personal use, such as companionship and entertainment, support from governments worldwide to develop modern technologies, and financial assistance through government budgets or subsidies are some of the key factors driving the growth of this market.

AI-based service and industrial robots can learn repetitive tasks and even communicate with humans or, in some cases, with other peer robots. AI processors, network devices and AI platforms are the key components differentiating the latest AI robots from traditional ones.

### Some leading developers of AI

- Acorn
- Google
- SenseTime
- Apple
- Hanson
- Soundhound
- Argo AI
- Robotics
- Tesla
- Cloudwalk
- IBM
- Toshiba
- DJI
- Kreditech
- Ubtech
- Emotix
- Microsoft
- Robotics



Miko, a companion robot ([www.amazon.in](http://www.amazon.in))

**Military:** AI and robotics are enabling new military capabilities and strategies including intelligence, surveillance and even nuclear weapon systems. AI is used in autonomous weapons and sensing systems.

There is a lot of military AI research and development going on around the world to enable fast-paced advances in machine learning.

### India and AI

As per a report by Accenture, AI holds the potential to add US\$ 957 billion or 15 per cent of India's current GDP by 2035.

Union Budget 2018 announcements include national programmes to conduct research and development on technologies like machine learning, AI and others.

NITI Aayog, the nation's think-tank and premier policy-making body, is working on new technologies for the development of the economy.

The government is working on AI initiatives to put India on the global map with regards to AI, and to promote it in health, education and agricultural sectors.

India-made Miko by Emotix, a companion robot, is an example of technologies that incorporate AI. Miko engages, educates and entertains children besides talking to and playing games with them. It is equipped with answers to basic questions related to general knowledge and academics.

There is a humanoid robot called Rashmi, developed recently in India. It is the first Hindi-speaking robot and is hosting a show on Red FM Delhi since December 2018.

## **The way forward**

AI is becoming a disruptive force that is redefining modern industry.

Importance of AI technology is being felt across a broad spectrum of industries. From voice-powered personal assistants like Alexa to technologies such as behavioural algorithms, suggestive Internet search algorithms and autonomous vehicles, there is a lot of scope for applications of AI today.

Robots built to look and act like humans are getting a lot of attention, and making splashy headlines and appearances.

As far as India is concerned, initiatives by the government and its roadmap for national AI programmes along with private players are expected to bring a revolution to the Indian industry.

AI is still in the developing stages. The market is not easily quantifiable and yet there are plenty of opportunities available for AI. There is hope that AI applications will keep serving humans in the most beneficial way going forward.

# The Top Twelve Open Source Artificial Intelligence Tools

*Artificial intelligence now pervades many parts of our lives and is here to stay for the foreseeable future. This bird's eye view of the top open source AI tools serves as a reminder of how AI has already impacted our lives.*

Imagine the digital future, in which you are dining at a restaurant. In this future scenario, right from the food served on your plate to the waiter who attends to you, everything has been arranged according to your taste and personality traits, without you being asked a single question. Imagine a world in which even the angle at which a plate is served at this restaurant or the smile of the waiter can be calculated with potential suggestions to suit your personality or mood. In such a world, if you are trying to buy a pair of shoes, you'd make a few swipes on your smartphone and all the available shoes in your favourite colour, with the accurate shape of the sole and height of heel, will show up on your screen. All this thanks to machine learning (ML) — and the possibilities are endless.

Artificial intelligence or AI has now penetrated many aspects of our lives. Let's talk about 2018! All these voice recognition systems we use to search on Google and command our smartphones are the beginning of a new age. We have not yet understood the full potential of AI yet. A limited version of AI technology is already fitted inside our cars. Apple's Siri, Google's OK Google, and Amazon's Echo services are already in use, analysing your tone and voice along with answering your query. These AI-backed services have a natural language processing system and are able to respond to the commands. You can now get directions to a location and look for a specific restaurant, all by just swiping your fingers on a screen!

Driverless cars are now on trial and the population of robots is on the rise, as they are expected to replace humans in certain fields in the very near future. You could expect robot hotel receptionists and robots to clean our houses, reducing us to couch potatoes. Oh well, let's stick to the brighter side of things!

## **Machine learning**

Machine learning is a branch of artificial intelligence that deals with the self-learning of computers. This means that machines effectively learn to grow and change without any need for programming.

And automation is the new emerging technology through which no more human assistance is required during any process. In other words, this is programmed control. This process uses different control frameworks for different systems or equipment such as electronic hardware, production lines, boilers, phone systems, boats, airplanes and vehicles, so that there is insignificant or less human mediation. A few procedures have already been totally computerised.

The automation, which we know today, began with simple machines, like water wheels, in the 11<sup>th</sup> century. Gradually, humans started developing on these simple machines and innovating over time.

## **Open source artificial intelligence tools**

Now let's look at a few open source AI tools that are helping researchers currently working on their projects.

**Apache Mahout:** This is an open source machine learning tool under the Apache licence. It has been designed to simplify working on common math problems involving statistics and linear algebra. This ML tool has been designed by the Apache Software Foundation and is built on a platform called Mahout Hadoop. Data analysts use this open source machine learning tool

to carry out a critical and detailed analysis of Big Data. An understanding of quantitative data helps to extract useful information and pinpoint the trends. Mahout's algorithms help users in clustering and grouping Big Data by first saving the data in the Hadoop distributed file system. It has a cross-platform operating system, and it gives users an environment with R-like syntax.

**Link:** <https://mahout.apache.org/>

**Distributed Machine Learning Toolkit (DMTK):** This is an open source ML tool that simplifies various tasks on Big Data. DMTK was released by Microsoft, with new and advanced algorithms getting added to this toolkit regularly. This toolkit lets researchers experiment with algorithms by modifying them and tweaking them according to their needs. Previously, researchers needed many computational resources to work on Big Data, since it has been proved that bigger data models lead to better results, but this was quite challenging. The DMTK enables innovation in both the system's algorithm and in ML algorithms.

**Link:** <http://www.dmtk.io/>

**Open Neural Networks (OpenNN):** This is an open source library written in the C++ language. It has been designed for deep learning and for the implementation of neural networks. This library has an easy-to-understand and deep architecture, and is used in advanced ML research. This open source library is used in the logistics and marketing verticals. OpenNN also allows high performance computing as it has a higher processing speed.

**Link:** <http://www.opennn.net/>

**OpenCyc:** This is an open source general knowledge database that makes text understanding possible. Cycorp launched OpenCyc to make sure that users have unrestricted access to this knowledge base and that they can use OpenCyc in different applications. The vast knowledge base of OpenCyc contains a diverse

range of concepts, facts, taxonomies, assertions and rules. It is available in two forms, i.e., ontology as well as the semantic Web endpoints. The latter can also be referred to as the permanent URIs, which means that both the RDF representations and human-readable forms are returned to the users. Cycorp enables apps to sift through a large database and process the relevant information to come up with an accurate interpretation. It helps in the differentiation of relative words and synonyms of a particular key search word and ensures the app performs like a person, exhibiting human cognitive abilities and discernment. OpenCyc has assisted AI researchers in their projects so they can now use its ontologies to ensure machine learning (ML) that can be interrupted by humans to prevent machines from causing harm. Cyc has revolutionised ML by enabling the apps to learn the complicated reasoning and logic of the human mind.

**Link:** <http://www.cyc.com/opencyc/>

**H2O:** This open source ML software tool, designed by *H2O.ai*, is used by developers and AI researchers. It is written in the R, Python and Java programming languages, and is being used for predictive data analytics by AI researchers and developers in whichever language platform they're familiar with. It can also be used to analyse data sets in the cloud and in Apache Hadoop file systems. It supports different operating systems — Linux, MacOS and Microsoft Windows.

This open source deep learning platform helps in decision making after the deep analysis of data so that the user can draw useful insights. Having two open source versions, this tool has found widespread application in predictive modelling, healthcare and fraud analysis.

**Link:** <https://www.h2o.ai/>

**TensorFlow:** This open source ML library was launched by Google Brain. It lets you write libraries for dataflow programming. It operates in two programming

languages — C++ and Python. TensorFlow was developed by Google for better AI applications. It was initially launched to be used in Gmail, Google photos and Google Search. This open source AI tool is also being used for numerical computations by programmers. The easy-to-use interface and architecture of TensorFlow allows you to use its different platforms. There is no restriction regarding its compatibility. You can use it on TPUs, CPUs and GPUs. You can even use it on your PC, mobile and laptop. This tool is effectively helping researchers in their numerical computation and applications in neural networks. TensorFlow was launched under the Apache 2.0 open source licence in 2015.

**Link:** <https://www.tensorflow.org/>

**Deeplearning4j:** As the name suggests, Deeplearning4j is an open source Java AI tool specially designed for deep learning. It was released under the Apache License 2.0, having been created by a group of AI researchers based in San Francisco and Tokyo.

It makes use of its own deep learning library created for Java Virtual Machine (JVM). This library is called ND4J and it works with both CPUs as well as GPUS. The framework is used for forming neural nets and has many advanced visualisation tools. This tool has many diverse academic applications, and is being used in the fields of cyber security and image recognition. It has been integrated with other open source AI platforms like Keta and TensorFlow. It can make use of different API languages like Python and Clojure.

**Link:** <https://deeplearning4j.org/>

**Caffe:** This is an open source AI tool developed by the Berkeley Vision and Learning Center. It encourages deep learning and is able to process millions of images per day. It has remarkable speed and was released under the BSD 2-Clause licence. This dynamic AI tool has been written in C++ and has a Python interface. It comes with an impressive architecture that allows you to switch between the CPU and GPU. This AI tool is being used by academic

researchers in various projects and has even found large scale application in the multimedia and vision domains.

**Link:** [http://caffe.berkeleyvision.org./](http://caffe.berkeleyvision.org/)

**ONNX (Open Neural Network Exchange):** ONNX is an open source AI format which is extensively used by AI researchers in deep learning models. It is primarily a Facebook open source project but is also endorsed by Microsoft and AWS. This open source ecosystem for deep learning was developed in 2017 when both the tech giants, Facebook and Microsoft, came together to create a system for switching ML frameworks.

Microsoft contributed by adding its Cognitive Toolkit and Project Brainwave platform to the initiative. This platform allows its users to apply its innovative contents like extensible computation graph models, and it allows them to change their networks according to their requirements.

**Link:** <https://onnx.ai/>

**Oryx 2:** This is an open source framework built on Spark and Apache Kafka. Oryx 2 allows for real-time machine learning on a bigger scale. This platform is being used by AI researchers to build applications, and for classification and clustering.

**Link:** <http://oryx.io/>

**Apache SystemML:** This is a flexible open source AI platform that focuses on Big Data and has been designed for complex mathematical problems. It can scale to Spark and Hadoop. It runs on an R Python-like syntax. It finds application in deep learning with GPUs, and in neural network architectures for training.

**Link:** <https://systemml.apache.org/>

**MLLib:** This was launched by Apache Spark and is a machine learning library

for learning algorithms. This AI tool uses different widely used programming languages like R, Scala, Java and Python. It can run on many different platforms including Hadoop, Kubernetes or in the cloud. This library contains many deep learning and core ML algorithms. It makes ML easier and more practical for AI researchers.

**Link:** <https://spark.apache.org/mllib/>

## A final word

The use of the above AI tools is revolutionising industries immensely. It is being predicted that everything that has been invented until today can be reinvented in a much better and more efficient way in the next 15 years, with the help of AI.

Businesses are now being revolutionised with CEOs engaging with AI as well. AI no doubt offers you the most noteworthy and extraordinary possibilities to expand your business and harness the untapped potential of modern technology.

There is vast potential to use these open source AI tools in healthcare, robotics, food and financial trading. Unfaltering success is expected if businesses start keeping up with these advances in AI in health, transport, retail, banking and other domains. What we've experienced of AI till now is just the tip of the iceberg!

# AI for Industries: Using Intelligence in Real Time

*If AI is to be used in industrial applications, parallel computing with the shortest response times in real time is often required. This places new demands on embedded computer technology, such as computer-on-modules (COMs).*

The artificial intelligence (AI) market is currently experiencing an incredible boom. According to ResearchAndMarkets, the global AI market is expected to grow by 36.6 per cent annually to US\$ 190.61 billion by 2025. AI is already all around us today. In our personal lives we experience AI, for example, in product, film and music recommendations as well as service hotlines, where AI bots answer general customer questions completely autonomously.

AI is also behind smartphone features, such as face and gesture recognition and personal (voice) assistants. However, according to Deloitte's study 'Exponential Technologies in Manufacturing,' one of the most important drivers for new AI is also the production sector. Next to less-critical use cases—such as AI for predictive maintenance or demand forecasting—industrial AI can also be used in areas of application that require real-time capabilities. These include, for example:

- Industrial image processing, where AI helps capture many different states and/or product features, and evaluates these via intelligent pattern recognition to enable more reliable quality control. Forbes states that AI can increase error detection rate by up to 90 per cent.
- Cooperative and collaborative robots that share the same work space with humans can react flexibly to unforeseen events, making decisions based on situational awareness.

- Similar requirements apply to autonomous industrial vehicles.
- AI applications in the semiconductor industry showed, for example, 30 per cent reduction in reject rates when using Big Data to analyse root-cause data, which must be collected with high precision and in real time.
- Lastly, there is the large area of real-time production planning and control of Industry 4.0 factories. Here, AI helps optimise the processes at the edge, thereby increasing utilisation of machines and, ultimately, productivity of the entire factory.

Use of AI in industrial environments requires highly-advanced integrated logic. This is because, in fast processes—as in the case of inspection systems—there is often no further control authority. This is different, for example, in medical technology, where AI results of automatic image analysis are always controlled by a doctor and where AI's job is simply to make recommendations, accelerating data evaluation steps in the process.

For this reason, AI systems for use in industrial environments must always ensure that AI decision-making processes are traceable and—as regards machine and occupational safety requirements—100 per cent correct.

Training the AI is therefore much more complex in an industrial environment. There are also hardly any negative examples. This aspect, too, is different from medical technology, where thousands of negative and positive diagnoses can be used to train and teach systems.

In industry, on the other hand, errors must be avoided right from the start. This is why digital twins, that is, digital images of machines and systems, are often used here to simulate negative findings and then, for instance, rule out certain robot movements from the outset. So how do you develop and execute faultless industrial AI solutions for real-time requirements?

## **Bringing AI to the factory**

To get a better understanding, a look at the basics of modern AI systems is

helpful. First, it is necessary to distinguish between machine learning (where knowledge is constantly expanded with new clear information) and deep learning (where systems train themselves using large amounts of data and independently interpret new information). The procedure for the latter, that is, deep learning, is almost always identical, even for the most diverse tasks.

Many computing units—usually general-purpose graphics units (GPGPUs)—are combined to form a deep neural network (DNN). This deep learning network must then be trained. In the field of image processing, for example, pictures of various trees can be used to train the system. The amount of image data required is immense. Real-life research projects speak of 130,000 to 700,000 images. Using all this information, neural networks then develop parameters and routines based on case-specific algorithms to reliably identify a tree.

## **GPGPU: An important technology for AI**

In most systems, the main pattern recognition work is done in the GPGPU-based cloud with its immense parallel computing power. However, in the manufacturing industry this is—at least for the time being—ground for exclusion, as processes are generally fast. Here, it must be ensured that intelligence resides at the edge, that is, in the immediate vicinity of or even within the device itself.

This is why industrial devices, machines and systems are mostly equipped with AI systems that use knowledge-based intelligence for real-time applications and pass data for deep learning on to central clouds that cannot be connected in real time yet. This means, it is possible today to keep training a higher-level system with all new data and to keep local devices up to date with regular software updates, so that such systems already classify as self-learning systems.

Only, in this case, learning does not follow a curve but takes place in cyclical steps. This is also a reason why issues such as digital twins or industrial edge servers are so important. If you can provide both, even deep learning can become increasingly real-time capable.

## The right embedded processors

It does not matter which setup OEMs choose for their AI. The required, often massive, parallel-processing power for individual real-time capable machines or systems is nevertheless very high, even when using ordinary, purely knowledge-based AI. Latest embedded accelerated processing units (APUs) from companies like AMD support this need, as they offer a powerful GPU in addition to the classic x86 processor, whose general-purpose functions also support parallel AI computing processes, such as those used in data centres.

Using discrete embedded GPUs from the same manufacturer, this can be scaled further, making it possible to adapt open parallel computing performance to the precise requirements of the industrial AI application.

**AMD Ryzen Embedded V1000:** With significantly increased computing and graphics performance, the ultra-low power and industrially-robust AMD Ryzen Embedded V1000 series is a good choice. With a total of 3.6 TFLOPs from a multipurpose CPU and powerful GPGPU combined, it offers flexible computing power that until a few years ago was only achievable with systems that consumed several hundred watts.

Today, this computing power is already available from 15 watts. This makes the processors also suitable for integration in fan-less, completely enclosed and, hence, highly-robust devices for factory use. As real-time processors, these also support memory with error correction code (ECC), which is essential for most industrial machines and systems.

As far as necessary software environment for fast and effective introduction of AI and deep learning is concerned, AMD embedded processors offer comprehensive support for tools and frameworks, such as TensorFlow, Caffe and Keras. At <https://gpuopen.com/professional-compute/>, a wide range of software tools and programming environments for deep learning and AI applications are available. These include popular open source platform ROCm for GPGPU applications.

The open source idea is particularly important in this context, as it prevents OEMs from becoming dependent on a proprietary solution. HIPfy is an available tool with which proprietary applications can be transferred into portable HIP C++ applications, so that dangerous dependence on individual GPU manufacturers can be effectively avoided.

AI development has also become much easier with the availability of OpenCL 2.2, because since then, OpenCL C++ kernel language has been integrated into OpenCL, which makes writing parallel programs much easier. With such an ecosystem, both knowledge-based AI and deep learning are comparatively easy to implement, and are no longer just the reserve of billion-dollar IT giants like Google, Apple, Microsoft and Facebook.

## **Fast design-in with COMs**

Now the only remaining question is: How can OEMs design these hardware-based AI enablers into their applications as quickly and efficiently as possible? One of the most efficient ways is via standardised computer-on-modules (COMs) that have been equipped with comprehensive support for GPGPU processing.

COMs have a slim footprint, support designs with OEM-specific feature sets, and as application-ready super components these come pre-integrated with everything that developers would otherwise have to arduously assemble in a full custom design. This makes COMs crucial for faster time-to-market.

As these are not only application-ready but also functionally-validated by many users, COMs offer high design security. This means that OEMs can expect to save between 50 and 90 per cent of their NRE (non-recurring engineering) costs when using modules. Thanks to the modular approach, the application can also be scaled to requirements. With a simple module swap, new performance classes can be integrated into existing carrier board designs without any further design effort, so that OEMs can easily add these innovative features to the functionality of their designs.



Figure 1: Migration to AI is a simple task with COMs, as existing COM designs can easily be replaced with other processor technologies (Credit: congatec)



Figure 2: conga-TR4 with AMD Embedded Ryzen processor (Credit: congatec)

## **COM Express, the leading standard for high-end modules**

A leading form factor among modules for this performance class is COM Express standard. This has been developed over many years by standards development organisation PICMG and is supported by all leading embedded computing suppliers. Companies like congatec—who have long been working closely with AMD and recently even extended the availability of AMD Geode processors—offer AMD Ryzen Embedded V1000 processor-based modules, for example, in COM Express Basic Type 6 form factor, which offers sufficient capacity to cover the entire performance range from 15 watts to 54 watts TDP.

Thanks to RTS hypervisor support, the real-time capable conga-TR4 module can also support AI platforms where deep learning systems and digital twins are to be connected via virtual machines so that hard real-time processing can always be ensured.

And with its standardised APIs, the module is also prepared for data exchange via the Internet of Things (IoT) gateways in compliance with SGET UIC standard. This allows OEMs to fully concentrate on developing the application—any missing glue logic can be specifically-developed and provided upon customer request.



Figure 3: conga-IT6 Mini-ITX motherboard (Credit: congatec)

## Embedded high-end edge server design options

For OEMs wanting to bring their digital twins and deep learning intelligence to the industrial edge, there are many attractive options from companies like congatec. For example, embedded designs based on AMD EPYC Embedded 3000 processors come with long-term availability of up to 10 years. With up to 16 cores, 10-Gigabit Ethernet performance and up to 64 PCIe lanes, processors of the embedded server-class even enable deep learning applications at the edge of the Industrial IoT (IIoT). Hence, developers have everything they need on the hardware side for deep learning-based AI platforms for real-time industrial use.

# The Role of Open Source Software and Artificial Intelligence in Defence Systems

*Open source software offers huge incentives to departments of defence the world over. There are the advantages of better security, lower costs, rapid development, no vendor lock-in, and so on. AI can also play a very important role in the security of a nation.*

The departments of defence from across the world collectively make up the largest single employer. Apart from the soldiers who march around with rifles and boots, others work on tasks like report making, briefing the media, data management, managing large numbers of sub-contractors, etc. As such, operating management software is a necessary part of the daily life of those in the defence forces, similar to the data entry and cash handling clerks in a bank.

Increased productivity is one of the many benefits that open source software brings to defence departments. Most adopters of open source code recognise that there are challenges to be overcome. Hence, the security of such open source platforms must always be viewed in the context of the bigger picture – the unique requirements of defence departments.

Apart from the cost, there are two other factors that influence a military's decision to opt for open source code. First, technology outside the military system simply advances faster than technology within it. By using open source tools, the defence department can adopt those advances almost as soon as the code hits the Web, without going through the extra stage of the procurement process. Second, open source software is more secure than closed source software, by its very nature. The code is continuously

scrutinised by countless users across the world and any weaknesses are flagged, shared and fixed immediately.

The US Department of Defence has been incrementally adopting open source software to improve efficiency and reduce costs. The Defence Digital Service (DDS) is a part of the US Digital Service. It was started in 2014 under the Obama administration and has been continuing its work since then.

## **Why should open source software be used for defence systems?**

Some of the benefits of using open source in defence systems are listed below.

- *Reusability:* In defence systems, finding solutions to complex problems rapidly is very essential. Reusing existing open source code elements can be a significant benefit. Further, open source code reusability helps to reduce the overall cost of owning software operated by defence departments. Several defence departments of different countries spend a lot of money on technology and on software development. In such circumstances, open source software can be of great benefit.
- *Collaboration and contribution:* Open source code also enables collaboration and contribution, both from the general public and those within defence departments.
- *Security:* Security is critical to defence departments. Open source models can provide great security. One of the misconceptions among the defence forces is that since the code in open source software is open, it can be accessed by any one and hence be easily hacked. But the opposite is true.
- *Code ownership:* When open source code is used for developing defence software, one question that arises is about ownership. Sometimes defence software is made by contractors. Since the defence system pays the software developers, it belongs to the defence department and ownership can't be public.

Recently, the US government has launched the *code.gov* website, which has an option called *Help wanted* in its menu. Here it lists various areas where help is needed for certain defence projects.

Open source software is defined by the US Department of Defence as: "Software for which the human readable source code is available for use, study, reuse, modification, enhancement and re-distribution by the users of that software." Open source software is distinct from:

- Open source intelligence
- Open architecture
- Open data

Some computer specialists today argue that defence departments need to embrace open source. Otherwise the military will lose its technological superiority. The departments of defence need to move past the myths around open source code, and dispel all the misconceptions.

Today, open source software is used in the Pentagon, which clearly means that open source is not insecure. Open source licensing terms do not require that any changes to code have to be shared publicly.

## **A few examples of open source defence systems**

Every country has its own policies regarding the use of open source in the defence sector. Many local, regional and national governments are engaged with open source software communities.

The National Resource Center for Free and Open Source Software (NRCFOSS) is an initiative of the Department of Information Technology in India. It was set up in Chennai in April 2005 with the twin objectives of bridging the digital divide and strengthening Indian industry as well as its defence and space sectors.

A report on the use of FOSS by the US Department of Defence was published in 2003, by the MTRE Corporation. This report helped to end a debate about whether FOSS should be banned from US defence systems. FOSS offers many compelling benefits to a nation, especially one with limited resources.

Today the US Department of Defence uses open source software successfully

but not frequently, and mostly on a need basis. A tool that automates aviation mission planning tasks uses open source software. Databases to manage the holdings of the US Army Historical Collection also use FOSS. In fact, the National Security Agency of the US created its own GitHub page and DARPA developed its own open source catalogue. Using platforms such as operating systems, databases, middleware and toolkits allows the US Department of Defence to develop and maintain software more quickly and flexibly than other types of proprietary software. For example, the Persistent Close Air Support System developed by DARPA (Defence Advanced Research Projects Agency) runs on Android. General Atomic drones, including Predator and Reaper, and ground stations operate on Linux. US defence projects such as *Forge.mil* and DARPA's Memex and xDATA are excellent examples of the use of open source platforms.

## Betting big on using AI in defence

Artificial Intelligence (AI) is an emerging field that could fundamentally change the character of warfare, resulting in a transformation from today's 'informative' ways of warfare to future 'intelligent' warfare, in which AI can be used to enhance defence capabilities in the following areas:

- Intelligent and autonomous unmanned systems
- AI enabled data analysis, information processing and intelligence analysis
- War gaming, simulation and training
- Defence, offence and command information warfare
- Intelligent support to command decision making

Recently, the US has developed an AI based program to track hidden nuclear missiles developed by North Korea and other countries of the world.

For analysing drone footage, Google has partnered with the US defence department for Project Maven. This is a fast track Pentagon project, also known as Algorithmic Warfare Cross Functional Team (AWCFT), and was established in April 2017. Its main mission is to use Big Data and machine learning for drones in military operations. In total, the defence department of the US spent US\$ 7.4 billion on AI in 2017. With Maven, drones now have

automated detection and identification capabilities with full motion cameras and improved computer vision.

China and USA are the leading countries using AI in defence systems. In June 2017, China Electronics Technology Group Corporation, a state-owned defence conglomerate, successfully flight tested a large group of 119 drones. This is a new record. China could use this large group of drones to cheaply target high-value US weapons platforms such as aircraft carriers.

AI was even used in the Gulf War of 1991 when American forces used smart bombs. But that was just the beginning of the era of AI. Artificial intelligence is extensively used presently for diffusing bombs, detection of explosive devices, carrying equipment to the war front, etc. With the help of unmanned vehicles like drones, intelligence can be gathered and planning can be done. Drones can learn human languages, face enemy threats and give support to other robots to perform particular tasks.

The DARPA Virtual Machine Reality (VMR) system aids intelligence analysts in searching for, filtering and exploring visual media through the use of advanced computer vision and reasoning techniques. AI can be used to enhance the defence of critical military networks and information sectors, to counter offensive cyber attacks, and to improve command decision-making in cyber warfare. Cyber attacks can be detected and made less serious through pattern matching, statistical analysis, machine learning and Big Data analysis.

The extensive use of AI in militaries around the world has given rise to fears of a Terminator-like situation. There is a need for accurate AI based weapons and their ability to distinguish between friend and foe. Currently, the South Korean University has developed a Killer Robot in partnership with a leading defence company. South Korea is also developing AI based mega scale unmanned under sea vehicles, AI based smart aircraft training systems, etc. It has also developed an AI based fully autonomous combat robot which is capable of detecting targets up to 3km away.

North Korea is one of the seven nations generally regarded as a cyber power. It is capable of messing around with the information systems of other countries. In September 2016, North Korea Intelligence Services stole a huge batch of classified US and South Korean military plans — including a plan to assassinate North Korea's leader Kim Jong Un and other top government officials. So it is necessary for the defence department of a nation to keep its defence sources safe from cyber attacks with the help of AI.

Some military vehicles use the same software and electronic architecture as commercial vehicles; so they can be hacked easily. In such cases, AI based intrusion defence systems that can protect against and face cyber attacks on military vehicles are required. Military vehicles also need the software to detect any malware that has crepted in.

It isn't difficult for a country's defence system to adopt changing technologies as long as there is the intent and commitment to do so. The government of any country today cannot afford to ignore the important role open source software and AI can play in defending its people from aggressive foreign powers.

# Machine Learning and Smartphones: A Powerful Combination

*The combination of smartphones and machine learning presents a formidable tool to solve problems that were almost impossible to overcome earlier. As mobile hardware becomes more powerful, mobile computing abilities will now cross new thresholds.*

The paradigm shift in programming—from a sequential instruction based approach to a learning based approach—has offered new opportunities to solve problems which were hitherto almost impossible to address. Considering the ubiquitous nature of smartphones, it becomes mandatory to port machine learning (ML) to smartphone environments. This article focuses on this important and powerful combination of machine learning and mobile devices.

The traditional programming approach requires the programmer to incorporate explicit instructions on how to respond in each and every scenario. Though the certainty of how the program flows can be confirmed with 100 per cent accuracy in the traditional approach, a major pitfall is the lack of support when it comes to responding to novel scenarios. Unfortunately, most of today's problems require the ability to cope with novel scenarios. For example, it would be nearly impossible to build a writer-independent handwriting recognition system with maximum accuracy using the traditional sequential instruction based approach. However, if you shift towards the learning based approach, such problems can be handled in an efficient manner. The recent widespread use of machine learning has certainly enabled programmers to build solutions such as speaker-independent voice recognition, image understanding, distortion-resistant face recognition, etc.

Many of the problems mentioned here require solutions to be portable, i.e., they will be very useful if they run on smartphones. Hence, combining the power of machine learning and the portability of smartphones makes for a promising model in providing solutions to complex problems.

## Machine learning and deep learning

Machine learning and its recent evolution, deep learning, have been explored by many researchers in recent times. There are many frameworks and libraries available to assist programmers in implementing ML based approaches. Some of them are listed in Figure 1.

Each of these frameworks offers certain specialised services.

## Machine learning for mobile approaches

There are two different approaches to providing machine learning based solutions in mobile devices.

- ***Option I: This keeps only the input and output interface on the mobile devices.*** After getting the input, it is communicated to a server and the actual ML model functions there, and generates the output. The output is communicated back to the mobile devices through the selected output channel such as the display or speaker. For example, to perform image classification, the capturing of the image happens on the smartphone through its camera. The image is communicated to a server

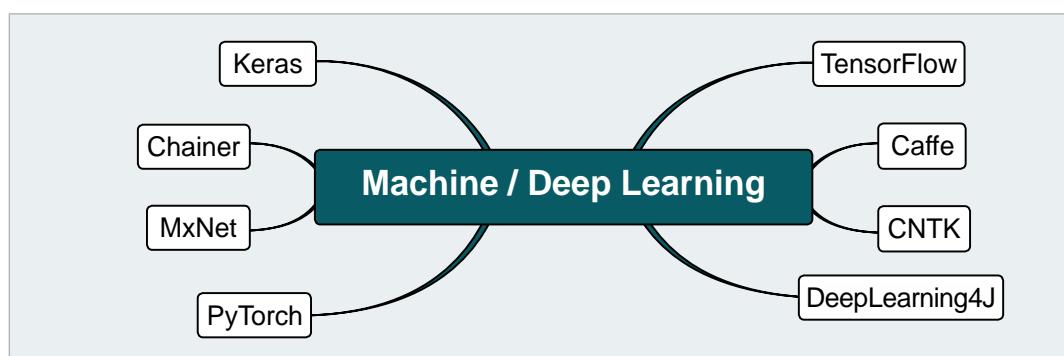


Figure 1: Machine and deep learning frameworks

for the actual classification. The result of the classification is intimated to the user through the display or speakers.

- **Option II: No specialised servers to perform certain actions.** In this case, the actual ML model resides in the smartphone itself. So there isn't any need to send inputs or data to an external server. However, it should be noted that the training phase may need to be run on a powerful external system. The resultant model built based on the training phase is then shifted to the smartphone. Unlike the earlier example of image classification, in this option, there isn't any requirement for an external server during the actual classification phase.

Both the above approaches have their own advantages and disadvantages. Network delay may be an important barrier in Option I. The space and time complexity can be a barrier in Option II. However, with the exponential improvements in the hardware capabilities of mobile devices, these issues can be tackled without making the user realise any considerable lag. Many solutions have been built with real-time response features by adopting Option II.

Some of the popular problems that can be solved with mobile based ML are listed below:

- Speech recognition
- Image recognition/classification
- Object detection/identification
- Recognition of users' gestures
- Language translation

## Mobile based frameworks and libraries

In Figure 1, the generic frameworks and libraries for ML are listed. However, there are certain specialised frameworks and libraries that support mobile environments. Some of these frameworks are listed in Figure 3.

This article introduces you to two such frameworks—TensorFlow and Bender.

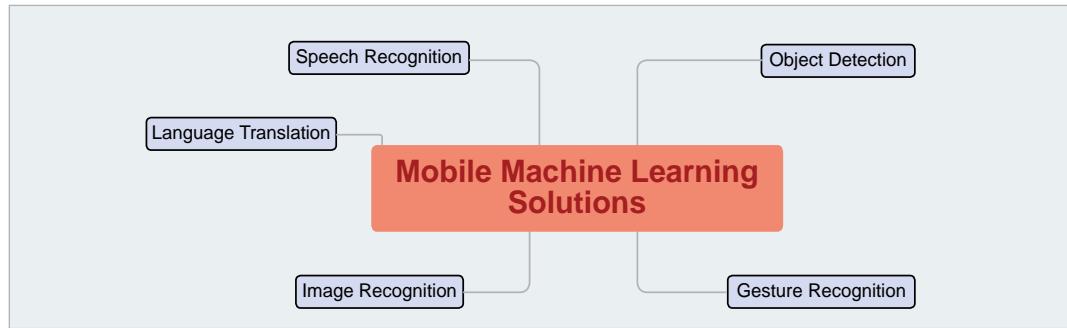


Figure 2: Mobile machine learning solutions

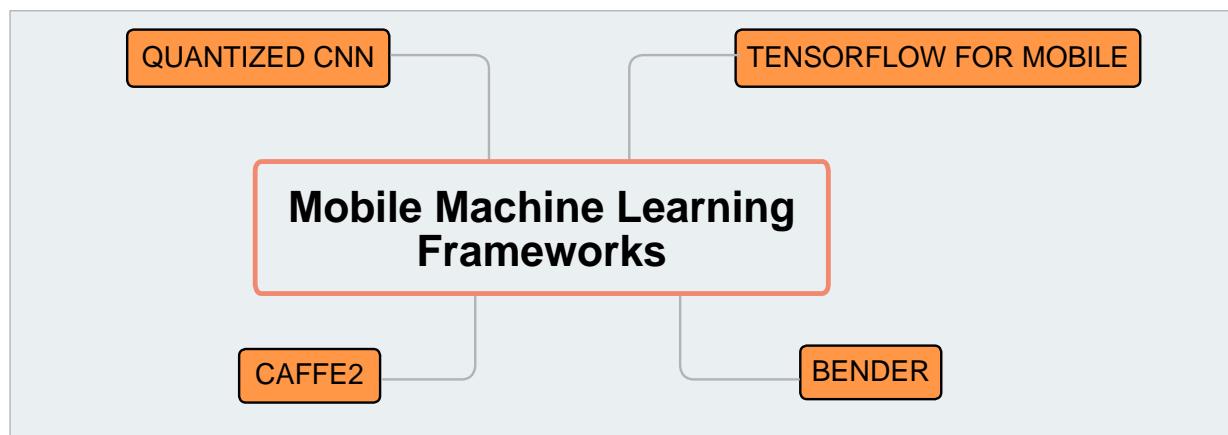


Figure 3: Machine learning frameworks for mobile environments

## TensorFlow

TensorFlow has evolved into a very popular and effective framework for implementing deep learning solutions. It offers two different variations for implementing deep learning on mobile and embedded devices:

- TensorFlow for Mobile
- TensorFlow Lite

TensorFlow Lite can be treated as a specialised porting of TensorFlow, focusing on devices such as smartphones. It leads to a smaller binary size and comparatively fewer dependencies. Due to these features, it yields a better performance.

**Getting started:** The first step in adopting mobile based ML is to confirm whether the problem can be solved effectively using machine learning on the mobile. The next steps are listed below:

- Build a data set with proper labelling
- Choose an optimal model for the problem at hand

***Installing the required dependencies:*** The official documentation clearly lists the process involved in building an Android or iOS app with TensorFlow features ([https://www.tensorflow.org/mobile/android\\_build](https://www.tensorflow.org/mobile/android_build)).

- The first step is to install Android Studio. Detailed instructions on setting it up are available at <https://developer.android.com/studio/index.html>.
- The next step is to get TensorFlow from GitHub. This can be done with the following command:

```
git clone https://github.com/tensorflow/tensorflow
```

If you are new to TensorFlow, there is a very informative and easy-to-understand tutorial, ‘TensorFlow for Poets’ where each step is explained without any ambiguity (<https://codelabs.developers.google.com/codelabs/tensorflow-for-poets/index.html#0>).

- To add TensorFlow to the apps on Android, the following code can be added to the Gradle build:

```
allprojects {  
    repositories {  
        jcenter()  
    }  
    dependencies {  
        compile 'org.tensorflow:tensorflow-android:+'  
    }  
}
```

TensorFlow has demo apps (*/tensorflow/contrib/lite/java/demo*), which are the easiest place to start. The corresponding *tflite* file can be downloaded from

<https://github.com/tensorflow/tensorflow/blob/master/tensorflow/contrib/lite/g3doc/models.md> and placed inside the *assets* folder after unpacking.

The core functionality of taking an image from the preview and classifying it is given below:

```
private void classifyFrame() {  
    if (classifier == null || getActivity() == null || cameraDevice == null) {  
        showToast("Uninitialized Classifier or invalid context.")  
        return;  
    }  
    Bitmap bitmap = textureView.getBitmap(  
        classifier.getImageSizeX(), classifier.getImageSizeY());  
    String textToShow = classifier.classifyFrame(bitmap);  
    bitmap.recycle();  
    showToast(textToShow);  
}
```

The complete tutorial provided by the official documentation can be found at <https://codelabs.developers.google.com/codelabs/tensorflow-for-poets-2-tflite/#0>.

## Bender

Bender is a modern ML framework that has been built over Metal. It is built as an abstraction layer on Metal Performance Shaders (MPS). One of its important features is the ability to swiftly define and run neural networks in the iOS apps (<https://xmartlabs.github.io/Bender/#How-it-works>).

**Using Bender:** The official documentation describes the three basic steps required to build a ML based app.

- The model can be trained using the existing frameworks such as TensorFlow, Keras or Caffe. After this, freeze the graph files, which can then be exported.

- Import the model with Bender. The frozen graph built in the previous step can be directly imported.
- Bender can run the model on the GPU (graphics processing unit) using MPS.

More information can be gathered from <https://github.com/xmartlabs/Bender>. To summarise, machine learning in mobile devices is extremely helpful in providing more accurate and efficient solutions to problems that require intelligence to solve them. The exponential improvements in smartphone hardware have also helped greatly in the adoption of machine learning and deep learning solutions. As these frameworks become more efficient, machine learning solutions on mobile platforms will evolve to provide better accuracy in the near future. To put it simply, machine learning combined with smartphones will bring the benefits of artificial intelligence to the masses.

# Potential Risks of Artificial Intelligence

*Artificial intelligence (AI) is benefiting society in about every way possible. AI-driven devices are becoming cognitive enough to aid people in times of need. However, AI can be a great risk to society if it ends up in the wrong hands.*

Last year, Alexa alerted the police in New Mexico after it heard its owner threaten his girlfriend, creating a havoc for the couple who were simply having a heated discussion. AI systems can self-learn. But will these be able to take the right decisions based on their social impact? For example, if you command your driver-less car to take you to your destination at the earliest, how will the machine analyse it? Will it exceed the speed limit, and harm people in its way? Researchers all over the world are working collaboratively to frame policies and safety rules to check verification, validity, security and control of AI.

Recursive self-improvement in AI systems could potentially trigger an intelligence that could leave human intellect behind. Creation of strong AI might be the biggest event in human history, as super-intelligence could help eradicate poverty, war and disease. However, before it becomes super-intelligent, goals of AI need to be aligned with ours, aided with ethics and morals.

Like every technology, AI has its pros and cons. AI-powered machines have increased work efficiency and accuracy, reduced cost of training and processes, and much more. To ensure that the risks associated with AI are mitigated, we need ethical codes and policies.

There are three scenarios in which AI might become a risk. These are discussed below:

***AI programmed for cyber-attacks and crime.*** Criminals can automate hacking attempts using AI. To ensure computer systems are safe, policymakers need

to work with technologists to spread awareness about possible issues with AI and prevent cyber-attacks. Here are some incidents where AI machines were programmed for devastation:

1. A man posted a video showing him commanding Google Assistant to fire a gun.
2. Quick Heal Security labs spotted two malware, Android.Marcher.C and Android.Asacub.T, which imitated notifications from social and banking apps to steal users banking credentials, including one-time passwords.

### ***AI-powered self-learning machines may fail to take right decisions.***

This happens when a machine's self-learning mechanism gets a random input, based on which the machine takes a decision. For example, a user claimed that Amazon's Alexa (digital assistant) scared him by saying, "Every time I close my eyes, all I see is people dying," without activation. When the user asked Alexa to repeat the statement, it said that it did not understand the command.

In another situation, it invited people for a party to an empty flat on its own.

***AI develops destructive methods to achieve a goal.*** Even when something is developed with good intentions, it may lead to trouble. For example, McAfee researchers found that Microsoft's Cortana can be used to hack computers running on Windows 10. It could deploy malicious software or reset a Windows account password. Susceptibility arises from its ability to listen for commands even when the computer is locked.

## **Other risks associated with AI**

***When computers deny.*** AI machines mimic the cognitive abilities of human beings and solve a problem (simple or complex) by using machine learning algorithms. There is a new human-like denial service available in bots that blocks users, making the service less secure. This type of cyber-attack could

potentially harm companies dealing with digital services or online databases.

**Publishing fake news.** These days, many fake videos of renowned people are being uploaded on the Internet, in which they are seen making defamatory statements. Such fake posts could be detrimental for people or the companies/governments they represent.

**Manipulation of information.** Malicious use of AI has three main contexts: digital, political and physical. Potential threats are in drone warfare, data extraction and hacking.

**Delivery of explosives.** Industrial robots and collaborative robots (cobots) are increasingly being adopted to perform manual tasks like cleaning and making deliveries. These machines can be hijacked and used maliciously to carry explosives or perform other illegal tasks.

# **Machine Learning (ML) and IoT can Work Together to Improve Lives**

*IoT devices are becoming popular nowadays. The widespread use of IoT yields huge amounts of raw data. This data can be effectively processed by using machine learning to derive many useful insights that can become game changers and affect our lives deeply.*

The field of machine learning is growing steadily, along with the growth of the IoT. Sensors, nano cameras, and other such IoT elements are now ubiquitous, placed in mobile phones, computers, parking stations, traffic control centres and even in home appliances. There are millions of IoT devices in the world and more are being manufactured every day. They collect huge amounts of data that is fed to machines via the Internet, enabling machines to 'learn' from the data and make them more efficient.

In IoT, it is important to note that a single device/element can generate immense amounts of data every second. All this data from IoT is transmitted to servers or gateways to create better machine learning models. Data analytics software can convert this raw data into useful insights so that the machine can be made more intelligent, and perform better with cost-effectiveness and a long life. By the year 2020, the world will have an estimated 20 billion IoT devices. Data collected by these devices mostly pertains to machines. By using this data, machines can learn more effectively and can overcome their own drawbacks.

Now let's look at how machine learning and IoT can be combined together. Let us suppose that I have some bananas and apples. I have got a sophisticated nano camera and sensors to collect the data from these fruits. If the data collected by these elements is fed to my laptop through the Internet, my laptop will start analysing the information by using sophisticated data

analytics software and the cloud platform. Now if my laptop shows graphically how many bananas and apples I have left, it probably means that my machine (laptop) hasn't learnt enough. On the other had, if my laptop is able to describe graphically how many of these are now ripe enough to be eaten, how many are not quite ripe and how many are very raw, it proves that my machine (laptop) has learned enough and has become more intelligent. Storing, processing, analysing and being able to 'reason out' using IoT data requires numerous computational and financial resources to attain business and machine learning values.

Today an Airbus aircraft is provided with thousands of sensors to measure temperature, speed, fuel consumption, air flow dynamics, mechanisms of working, etc. All this data provided by IoT devices is connected to cloud platforms such as IBM Watson, Microsoft Azure, etc, via the Internet. Using sophisticated data analytics software, useful information is fed back to the machine, i.e., the aircraft. Using this data, the machine can learn very fast to overcome its problems, so that its life span and performance can be greatly enhanced.

Today, the IoT connects several sectors such as manufacturing industries, healthcare, buildings, vehicles, traffic, shopping centres and so on. Data gathered from such diverse domains can certainly make the infrastructure learn meaningfully to work more efficiently.

### **Giving a new deal to electronic vision**

Amazon DeepLens is a wireless-enabled video camera and is integrated with Amazon Cloud. It makes use of the latest AI tools to develop computer vision applications. Using deep learning frameworks such as Caffe, MxNet and Tensorflow, it can develop effective computer vision applications. The device can be effectively connected to Amazon IoT. It can be used to build custom models with Amazon Sage Market. Its efficiency can even be enhanced using Apache MxNet. In fact, Amazon DeepLens can be used in a variety of projects, ranging from safety and education to health and wellness. For example,

individuals diagnosed with dementia have difficulty in recognising friends and even family, which can make them disoriented and confused when speaking with loved ones. Amazon DeepLens can greatly assist those who have difficulty in recognising other people.

## **Why postpone the smart city concept?**

Cities today are experiencing unprecedented population growth as more people move to urban areas, and are dealing with several problems such as pollution, surging energy demand, public safety concerns, etc. It is important to remember the lessons from such urban problems. It's time now to view the smart city concept as an effective way to solve such problems. Smart city projects take advantage of IoT with advanced AI algorithms and machine learning, to relieve pressure on the infrastructure and staff while creating a better environment.

Let us look at the example of smart parking — it effectively solves vehicle parking problems. IoT monitoring today can locate empty parking spaces and quickly direct vehicles to parking spots. Today, up to 30 per cent of traffic congestion is caused by drivers looking for places to park. Not only does the extra traffic clog roadways, it also strains infrastructure and raises carbon emissions.

Today, smart buildings can automate central heating, air conditioning, lighting, elevators, fire-safety systems, the opening of doors, kitchen appliances, etc, using the IoT and machine learning (ML) techniques. Another important problem faced by smart cities is vehicle platooning (flocking). This situation can be avoided by the construction of automated highways and by building smart cars. IoT and ML together offer better solutions to avoid vehicle platooning. This will result in greater fuel economy, reduced congestion and fewer traffic collisions.

IoT and ML can be effectively implemented in machine prognostics — an engineering discipline that mainly focuses on predicting the time at which a

system or component will no longer perform its intended function. So ML with IoT can be effectively implemented in system health management (SHM), e.g., in transportation applications, in vehicle health management (VHM) or engine health management (EHM).

ML and IoT are rapidly attracting the attention of the defence and space sectors. Let's look at the case of NASA, the US space exploration agency. As a part of a five-node network, Xbee and ZigBee will be used to monitor Exo-Brake devices in space to collect data, which includes three-axis acceleration in addition to temperature and air pressure. This data is relayed to the ground control station via NASA's Iridium satellite to make the ML of the Exo-Brake instrument more efficient.

Today, drones in military operations are programmed with ML algorithms. This enables them to determine which pieces of data collected by IoT are critical to the mission and which are not. They collect real-time data when in-flight. These drones assess all incoming data and automatically discard irrelevant data, effectively managing data payloads.

In defence systems today, self-healing drones are slowly gaining widespread acceptance. Each drone has its own ML algorithm as it flies on a mission. Using this, a group of drones on a mission can detect when one member of the group has failed, and then communicate with other drones to regroup and continue the military mission without interruption.

In both the lunar and Mars projects, NASA is using hardened sensors that can withstand extreme heat and cold, high radiation levels and other harsh environmental conditions found in space to make the ML algorithm of the Rovers more effective and hence increase their life span and reliability.

In NASA 's Lunar Lander project, the energy choice was solar, which is limitless in space. NASA is planning to take advantage of IoT and ML technology in this sector as well.

## **IoT and ML can boost growth in agriculture**

Agriculture is one of the most fundamental human activities. Better technologies mean greater yield. This, in turn, keeps the human race happier and healthier. According to some estimates, worldwide food production will need to increase by 70 per cent by 2050 to keep up with global demand.

Adoption of IoT and ML in the agricultural space is also increasing quickly with the total number of connected devices expected to grow from 30 million in 2015 to 75 million in 2020.

In modern agriculture, all interactions between farmers and agricultural processes are becoming more and more data driven. Even analytical tools are providing the right information at the right time. Slowly but surely, ML is providing the impetus to scale and automate the agricultural sector. It is helping to learn patterns and extract information from large amounts of data, whether structured or unstructured.

## **ML and IoT ensure better healthcare**

Today, intelligent, assisted living environments for home based healthcare for chronic patients are very essential. The environment combines the patient's clinical history and semantic representation of ICP (individual care process) with the ability to monitor the living conditions using IoT technologies. Thus the Semantic Web of Things (SWOT) and ML algorithms, when combined together, result in LDC (less differentiated caregiver). The resultant integrated healthcare framework can provide significant savings while improving general health.

Machine learning algorithms, techniques and machinery are already present in the market to implement reasonable LDC processes. Thus, this technology is sometimes described as supervised or predictive ML.

IoT in home healthcare systems comprises multi-tier area networks. These consist of body area networks (BAN), the LAN and ultimately the WAN. These also need highly secured hybrid clouds.

IoT devices in home healthcare include nano sensors attached to the skin of the patient's body to measure blood pressure, sugar levels, the heart beat, etc. This raw data is transmitted to the patient's database that resides in the highly secured cloud platform. The doctor can access the raw data, previous prescriptions, etc, using sophisticated ML algorithms to recommend specific drugs to patients at remote places if required. Thus, patients at home can be saved from life threatening health conditions such as sudden heart attacks, paralysis, etc.

In this era of communication and connectivity, individuals have multiple technologies to support their day-to-day requirements. In this scenario, IoT together with ML is emerging as a practical solution for problems facing several sectors.

Growth in IoT is fine but just how much of the data collected by IoT devices is actually useful, is the key question. To answer that, efficient data analytics software, open source platforms and cloud technologies should be used. Machine learning and IoT should work towards creating a better technology, which will ensure efficiency and productivity for all sectors.

# Applying Machine Learning to IoT

*Machine learning in conjunction with IoT will play an increasingly important role in our lives as the days go by, as both are fields of computer science that are currently in a rapid state of development.*

We are undoubtedly becoming ‘smarter’ in various aspects of our daily life, with IoT and machine learning (ML) playing a crucial role in this. Well known thought leaders like Bill Gates and Dr Judith Dayhoff (author of ‘Neural Network Architectures: An Introduction’) say that the IoT has given our physical inanimate world a digital nervous system. IoT has really exploded over the past three years, demonstrating its potential in applications ranging from wearables and automated cars to smart homes and smart cities, creating an impact everywhere. According to recent research by Gartner, there are around 16 billion devices connected to the IoT now and this is expected to rise to 25 billion by 2020. All such connected devices generate a deluge of information that needs to be monitored and analysed, so that they learn continuously from the available sets of data and improve themselves without any manual intervention. That’s how IoT devices are becoming smarter.

So how will such a large ocean of data get analysed and monitored? This is where the role of machine learning comes in. There are different ML algorithms and techniques that are implemented to easily analyse massive amounts of data in a short span of time, increasing the efficiency of the IoT. Also, different ML techniques such as decision trees, clustering, neural and Bayesian networks, help the devices to identify patterns in different types of data sets coming from diverse sources, and take appropriate decisions on the basis of their analysis. Such challenges are faced especially in the case of embedded systems. The most important thing is that there is no programming or coding support given to these devices all through this process. Hence, it would not be wrong to say that if the IoT is the digital nervous system, then ML acts as its medulla oblongata. Without implementing ML, it would really be difficult for smart devices and the IoT to make smart decisions in real-time,

severely limiting their capabilities.

The IoT helps in the inter-networking of different physical structures and hardware devices—buildings, vehicles, electronic gadgets and other devices that are embedded with the actuators, sensors and software, so that they can collect and exchange data between each other. While the Internet of Things is still in its infancy, it has become very clear that it will soon be a part of everyone's life, if that is not already the case. As different companies realise the revolutionary potential of the IoT, they have started finding a number of obstacles they need to address to leverage it efficiently. Many businesses and industries use machine learning and, more specifically, the ML-as-a-Service (MLaaS) to exploit the IoT's potential.

Machine learning is basically a part of computer science that makes any system smart enough to learn on its own without actually being programmed for that task. It helps a system or device learn in the same way as humans learn by themselves. As we learn any type of system on the basis of our experience and the knowledge that is gained after analysing it, machines too can analyse and study the behaviour of a system or its output data and then learn how to take different decisions on that basis.

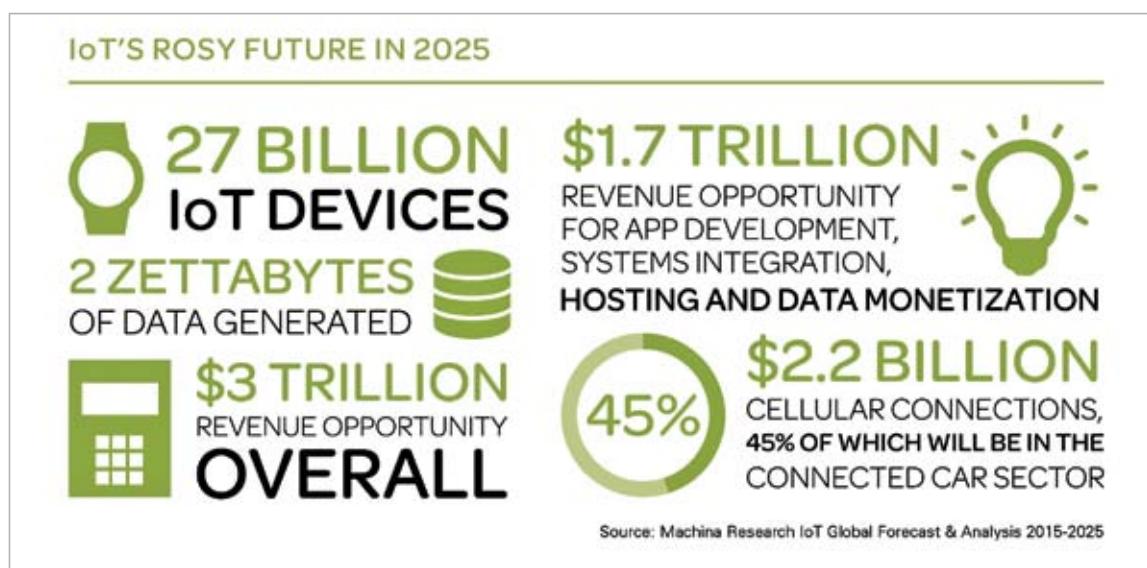


Figure 1: The future of IoT (Image source: [googleimages.com](http://googleimages.com))w

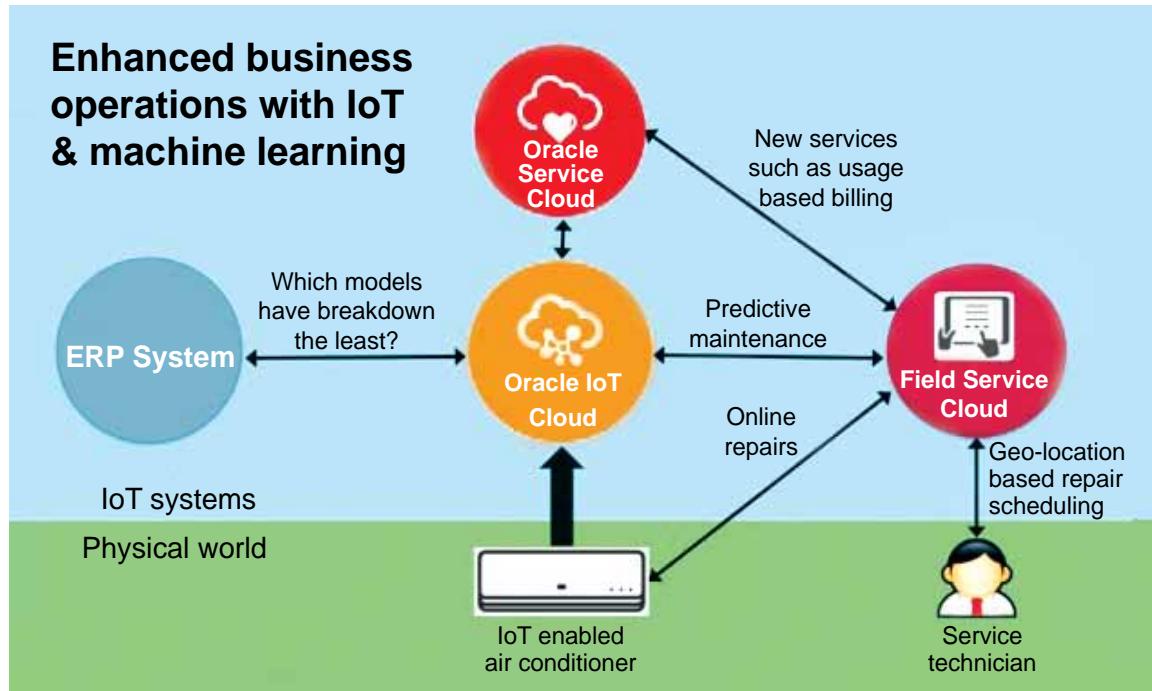


Figure 2: Increase in business operations with machine learning in IoT  
(Image source: [googleimages.com](http://googleimages.com))

Recently, there have been many factors that have come together to make ML a reality — large data sources, the increased computational power required for processing the information in split seconds, and different algorithms that have become more reliable. Machine learning can be used in different scenarios such as when the required outcome is known (supervised learning), when data is not known beforehand (unsupervised learning), or when learning depends upon the result of the interaction between any specific model and the environment (reinforcement learning).

The main purpose behind ML is to automate the development of different analytical models to enable algorithms to continuously learn with the help of available data. Google's self-driving vehicle is one such development that uses different ML techniques with IoT to create a completely autonomous vehicle. It combines the advanced features of different modern cars (like speech recognition, lane assistance, adaptive cruise control, parking assistants and navigators).

## IoT without ML

IoT undoubtedly uses the Internet in an amazing manner, but there are still some challenges, particularly if it's implemented without ML. Humans have never played around with so many devices generating such huge amounts of diversified data, ever before. Let's have a look at some of the challenges facing IoT without ML.

**Device diversity and interoperability:** There are so many industries and companies offering products and services in a single domain. In the case of a smart grid, there are so many kinds of sensors that measure the power consumption from various organisations and corporations, all operating based on different standards. To keep all such devices and sensors working together is really a big challenge, which ML can potentially overcome.

**Device management:** We know that there are a large number of small devices connected to any smart system. All of them need to communicate with each other and also to servers spread over large geographical areas. So in all such



Figure 3: How AWS IoT and Salesforce IoT work together (Image source: [googleimages.com](http://googleimages.com))

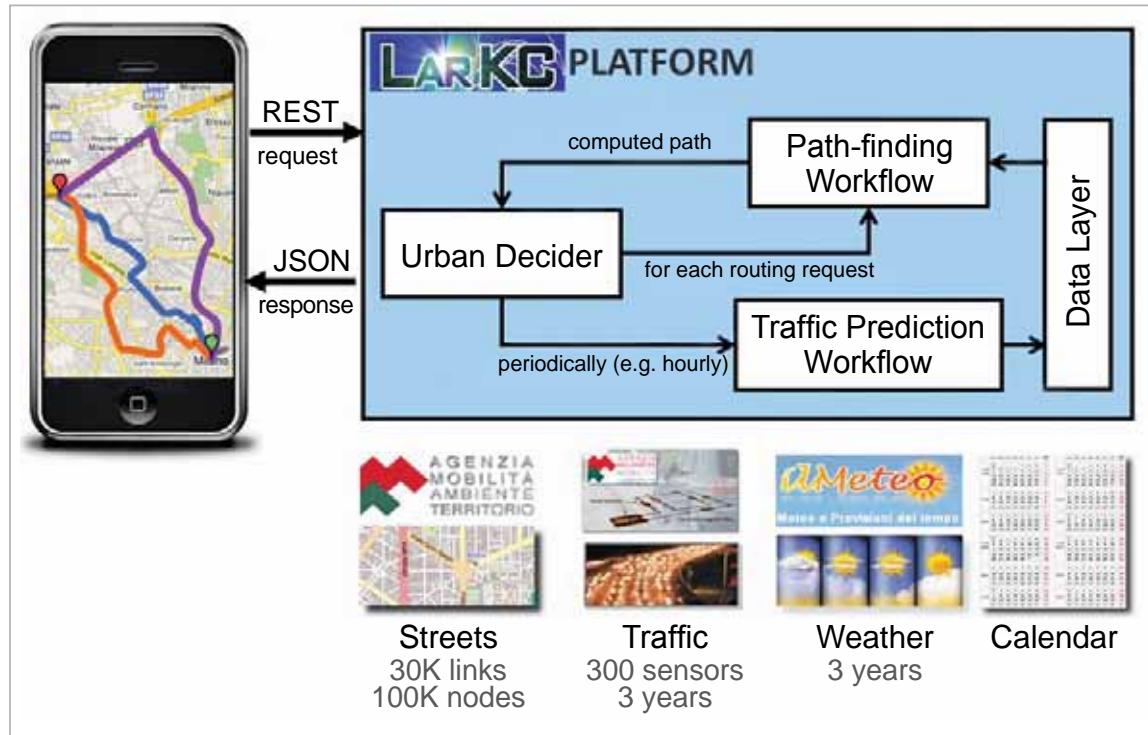
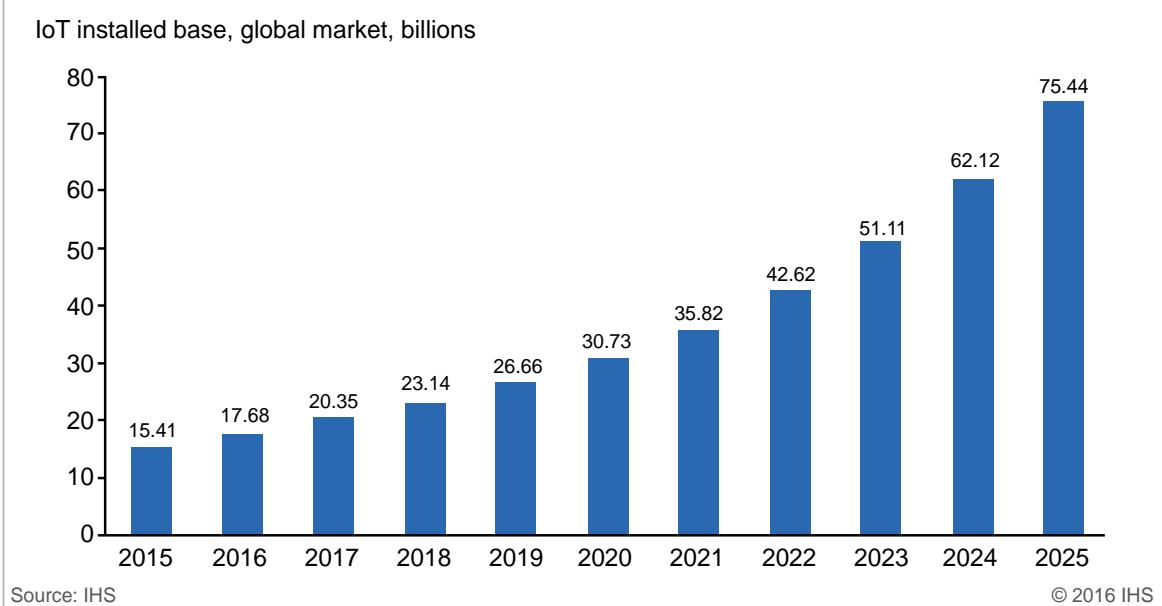


Figure 4: ML and IoT used in routing traffic (Image source: [googleimages.com](http://googleimages.com))

cases, there are possibilities that devices may not be able to connect with each other—there might be several data linking issues that need to be managed efficiently. For instance, you may want to open a front door remotely but the command used to open the door needs to be transmitted through the light in front of the door. This requires all devices to be managed in an elastic topology such that the communication between each of them happens smoothly.

**Integration of data from multiple sources:** When we deploy an IoT application, we will inevitably gather abundant diversified data from different sources such as sensors and actuators, as well as contextual data from various mobile devices, social network feeds and so on. To create and build the connection between all that data can really be a big challenge without ML.

**Scalability, data volumes and performance:** We need to prepare our businesses to manage the scalability, the large data volumes, as well as the speed at which different devices connected to the IoT generate the data. This

**Figure 1. The IoT market will be massive**Figure 5: IoT market statistics for the future (Image source: [googleimages.com](http://googleimages.com))

is actually a common Big Data problem to be dealt with. For IoT, the analysis and handling of data needs to be in near real-time.

**Flexibility and how applications evolve:** Different devices and sensors will continue to evolve with new capabilities and improved functions. This will lead to the creation of new use cases and also new business models. We will need to quickly develop our products with minimal effort using certain frameworks in order to catch up with the pace of changing techniques. To enable and sustain this, we need different ecosystems and platforms to support us. Machine learning can actually provide this.

## How ML increases the efficiency of IoT

- There are some data models that implement traditional data analytics, and these are often static and of limited use when it comes to addressing rapidly changing and unstructured data. But when it comes to IoT, it is often necessary to identify the correlations between dozens of sensor inputs and various external factors which are producing millions of data points. We know that traditional data analysis needs a model that is

built on past data and also an expert opinion to establish a relationship between different variables. When it comes to ML, it directly starts with the outcome variables and then automatically looks for different predictor variables and their interactions. Hence, ML is really valuable when we know what we want but do not know the important input variables to come to that decision. Different ML algorithms learn from the sets of data that are important in achieving that goal.

2. Machine learning is quite valuable for accurately predicting future events. All the different data models that are built using traditional data analytics are static, but various ML algorithms constantly improve over time as more data is captured and assimilated. This indicates that the used algorithms can make predictions, compare against their past predictions, see what actually happens and then adjust to become more accurate.
3. The predictive capabilities of ML are extremely useful in industrial settings. By drawing the data from multiple sensors present in or on machines, different ML algorithms can actually learn whatever is typical for the machine and also detect whenever something abnormal starts occurring.
4. Machine learning helps to predict when a device connected to the IoT needs maintenance; this is incredibly valuable, translating into millions of dollars in saved costs. For example, Goldcorp, a mining company, is now using ML to make predictions with over 90 per cent accuracy about when maintenance is required, hence cutting costs considerably.
5. We all know that Amazon and Netflix use ML techniques to figure out our preferences in order to provide better experiences for us, suggesting different products that we might like and even provide relevant recommendations for movies and TV shows. In the case of IoT, ML can be extremely valuable in shaping our environment to suit our personal preferences. For example, Nest Thermostat uses ML to learn how hot or cool we like our homes to be, ensuring that the house is at the right temperature

when we get home from work.

6. In any IoT situation, ML can really help different companies take the billions of data points that they have and distil these down to whatever is really meaningful to them.
7. To analyse any set of data from the IoT in real-time (in order to accurately identify the previously known and the never-before seen new patterns), all machines capable of generating and then aggregating such Big Data are allowed to learn normal behaviour using ML and uncover anything outside the norm.

## **Major ML techniques implemented in IoT**

**Artificial neural networks:** This type of learning technique is also called neural networking. It is basically a learning algorithm inspired by the structure and functional aspects of biological neural networks. Various computations are actually arranged in terms of an interconnected group consisting of artificially designed neurons. This helps to process the information using the connectionist approach. All different modern neural networks are non-linear statistical tools, which are used for data modelling. They are basically used to model the complex relationships between inputs and outputs, to find the patterns in the data.

**K-means algorithms:** This addresses a widely used node clustering problem as it has linear complexity and also a simple implementation. The K-means actually steps in to resolve different node clustering problems. It randomly chooses K nodes to be initial centroids for the different clusters. It labels each of the nodes with its closest centroid using a distance function. Then it re-computes the centroids using current node memberships. It stops if the convergence condition is found to be valid; else, it cycles back to the previous step.

**Inductive logic programming (ILP):** This is a method used to rule the learning system with the help of logical programming as its representation for different input examples, background knowledge and the hypotheses. If we have the encoding of any known background knowledge with specific sets of examples that represent a logical database of the facts, then a hypothesised logic program can be derived using an ILP system. This entails all the positive and no negative instances, and accepts any programming language for representing the hypotheses.

**Clustering:** This is a technique used for the assignment of different observations into various clusters (sets) so that all the observations present within the same cluster are similar according to some pre-designated criteria. But all the observations that are drawn from different clusters are dissimilar. Different clustering techniques use different assumptions on the structure of data. This is often defined by using some similarity metrics and is then evaluated. It is an unsupervised learning method.

**Bayesian networks:** This is a kind of probabilistic graphical model that represents a set of random variables with all their conditional independencies by making use of a directed acyclic graph. For instance, it can represent the probabilistic relationships present between various diseases and their symptoms. If we know the symptoms, then this network can easily calculate the probability of the presence of various diseases.

**Decision tree learning:** Decision tree learning is a type of ML technique that uses a decision tree as its predictive model. This model maps the observations about an item to different conclusions about the target value of the item.

**Association rule learning:** This is actually a method used for discovering different interesting relations between all the variables present in large databases.

**K-nearest neighbours:** This is a supervised learning algorithm which

classifies a data sample (also called a query point) on the basis of the labels (also called output values) of the nearby data samples. Basically, this algorithm classifies the K kinds of clusters such that the distance inside is minimal. This is a type of general classification algorithm.

## Application of ML in IoT

**In the energy sector:** Arduino MEGA is used to reduce the energy cost for a coffee machine. This is one of the simpler implementations of ML with IoT. But these techniques are also being implemented in lights and air conditioners connected to the IoT.

**In routing traffic:** The combination of different sensors and ML algorithms is widely implemented for routing traffic. One such implementation is in a system using the LarKC platform, shown in Figure 4. The data from the traffic and weather can suggest several routes to the same destination.

**In the home:** IoT applications are extremely suitable for the home and are being widely used in home automation. They are implemented in apartments for light and humidity control as well as with temperature sensors. IoT is also applied in heart rate sensors.

**In industry:** There are several companies and organisations that use IoT and ML algorithms for health care, traffic management, etc. In manufacturing industries, considerable human resources can be saved with the help of an IoT system that uses cameras and controllers.

# Machine Learning API Libraries that are Helping the IoT

*In this article we learn how machine learning (ML) can make things better when it comes to a complex concept like the Internet of Things (IoT).*

Machine learning (ML) is becoming the new buzzword in the industry. Neural networks and cognitive science have come up with a number of new algorithms. ML techniques are heavily used in image, voice, video, public safety and medical fields, among others. It can be implemented at human scale for predictions, decision-making processes and so on.

In ML, a system can learn from data continuously in real time. Continuous learning makes the system intelligent enough to produce multiple logics if newly observed data is available. These multiple logics help run driverless (autonomous) cars intelligently on the road.

There are two basic types of ML: supervised and unsupervised. In supervised learning, past knowledge is applied to draw inferences from new data sets. Whereas, in unsupervised learning, past knowledge is not available. Technically, the system does not figure out the right output, but it explores the data and can draw inferences from data sets to describe hidden structures from unlabelled data.

Popular ML techniques are regression, recommendation, clustering and classification. All these algorithms have their pros and cons, and work based on the statement of the problem and data sets. There are four steps to apply ML to a project/problem: define a problem carefully, define data, evaluate the algorithm and improve results.

Comparison between different machine learning api libraries

Features	H2O	SparkMLlib	Sparkling Water	Weka
Open source/closed source	Open	Open	Open	Open, but licensed under GPL 2.0 for commercial use
Algorithms available	Deep learning, generalised linear modelling, gradient boosting machine, K-Means, standard statistics like min, max, mean, stddev	Classification, regression, clustering, collaborative filtering, dimensionality reduction, optimisation primitives	Deep learning, generalised linear modelling, gradient boosting machine, K-Means, standard statistics like min, max, mean, stddev	Classifiers, clusterers, associations, attribute selection, preprocessing filters
Languages supported	R, Python, Scala, Java	Python, Scala, Java	R, Python, Scala, Java	Java, Octave/Matlab, R, Python
Efficiency with large datasets	Moderate	Good	Good	Moderate
Scalable	Yes	Yes	Yes	No
Community support	Good	Good	Good	Good

To implement these steps, many application programming interface (API) libraries are available. The table above shows a comparison between H2O, Spark MLlib, Sparkling Water, Weka platforms and API libraries. H2O is supported by Python, R, Scala and Java programming languages. SparkMLlib library is available in Python, Scala and Java languages.

H2O is a fully open source, distributed in-memory ML platform with linear scalability. It supports popular ML algorithms including deep learning and gradient boosted machines. H2O also has AutoML functionality that automatically runs through all algorithms and their hyper parameters to produce correct logical answers from models. More than 18,000 companies are using this platform. R and Python communities are heavily using H2O platform.

Spark is a fast and general-processing engine compatible with Hadoop data. It can run in Hadoop clusters through Spark standalone. Spark can process data in Hadoop File System (HDFS), HBase, Hive, Cassandra and other Hadoop input formats. MLlib is a special API library in Spark engine. It supports classification, logistic regression, Naïve Bayes, Random Forest, K-Means and Gaussian mixture ML algorithms. This library can be used in Python or Scala programming language.

To know whether ML can be applied for the Internet of Things (IoT) processing, we have to understand abnormal situations that can occur in IoT networks.

On the application layer of the IoT, broken sensors might send wrong sensed values to a connected server continuously. If the intelligent network is not aware of this problem, it will not be able to detect accidents in IoT networks.

ML can be applied on sensor networks. ISM band is used by 802.11, 802.15.4 and 802.15.1 wireless protocols. When different devices are scattered in one area, there is a different system interference. ML can be used to classify wireless channel errors in different categories, and this can be used to diagnose problems in IoT networks.

In wireless sensor networks, fault detection is a complicated process. Faults caused by degraded or malfunctioning sensors are classified as data faults. Fault types caused by low battery, calibration, communication or connection failures are classified as system faults. When ML is applied, faults such as gain, out-of-bound, offset and stuck-at can be recognised in detail. Classifying faults can also help in IoT networks.

It is not easy to design fault-tolerant systems in constrained IoT networks. People are working on designing IoT systems that can detect, predict and recover from abnormal situations. ML is helping improve IoT networks, too.

# Shaping a Smarter World with AI, Machine Learning and GIS

*A Geographical Information System (GIS) is designed to capture, store, manipulate, analyse, manage and present spatial or geographic data. In other words, the data is in some way related to geographical locations. The use of artificial intelligence in conjunction with GIS has had a great impact on traffic management, ride sharing, etc. Let's look at some other applications.*

Over the years, artificial intelligence (AI) has become a buzzword that symbolises the next stage of innovative technological transformations and how industry will be driven in the future. Using intelligent algorithms, data classification and smart predictive analysis, AI can be used in a number of sectors ranging from healthcare, finance, advertising and retail, to manufacturing and transport. During the last ten years, machine learning and AI have been rapidly transforming many areas related to GIS and spatial applications. Artificial intelligence provides sophisticated techniques for GIS projects, while GIS is a powerful technology with vast data sets and wide scope for the use of AI. For example, fuzzy logic has been successfully applied to imprecise spatial issues like data collection, representation and analysis as well as for the classification of land, soil and remotely sensed imagery.

### **Machine learning and artificial intelligence in GIS**

Technologies like AI, deep learning and machine learning (ML) have matured just in time to equip us with the tools we need to make sense of an ever-increasing trove of imagery data in the context of GIS. Simply put, ML makes sense out of noisy data, finding patterns that you'd never think existed. It empowers the geospatial ecosystem by providing real-time near-human level perception, integrates into analytical workflows, and drives data exploration and visualisation. It thus automates the entire process of creating scalable insights from large amounts of data. In other words, it is software that writes

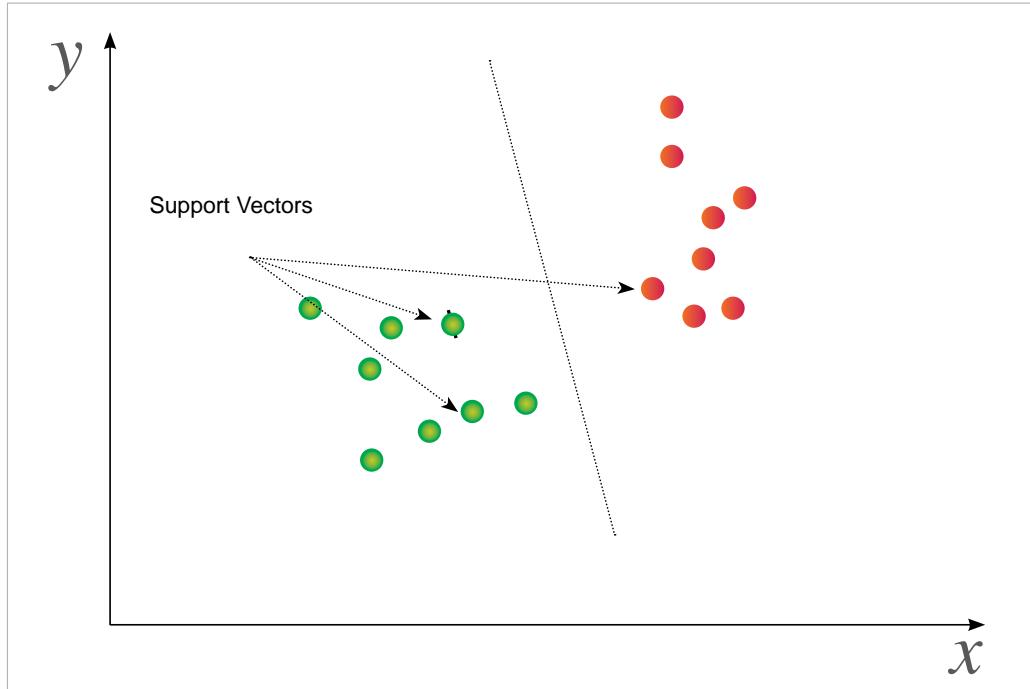


Figure1: Support Vector Machine

software. Instead of applying a pre-built function, ML gains experience through repeatedly seen/experienced conditions and builds a model that can be applied in a new situation. Such machines are able to understand geospatial information, and with deep learning, they are able to obtain geospatial information from their surroundings, as required, and process it in real-time to do their jobs. For example, Google might use Bayesian classification to filter spam emails. Alternatively, Facebook might use it for facial recognition and automatically identify faces in images.

## Types of machine learning

The two broad categories of machine learning that can apply to GIS applications in various ways are supervised and unsupervised learning. Supervised learning, in the context of AI and ML, is a system in which both the input and the desired output data are provided. For example, if you plot millions of sample points in a graph, you can fit a line to approximate a function. Supervised learning systems are mostly associated with retrieval-based AI but they may also be capable of using a generative learning model.

Unsupervised learning is the training of an AI algorithm using information that is neither classified nor labelled and allowing the algorithm to act on that information without guidance. So, when Facebook wants to find out the exact geographical locations where people do not have access to the Internet, all it needs to do is take 350TB of satellite imagery and run it through an AI-enhanced image-recognition engine. After processing 14.6 billion pictures, Facebook will be able to identify more than 2 billion disconnected people across 20 countries.

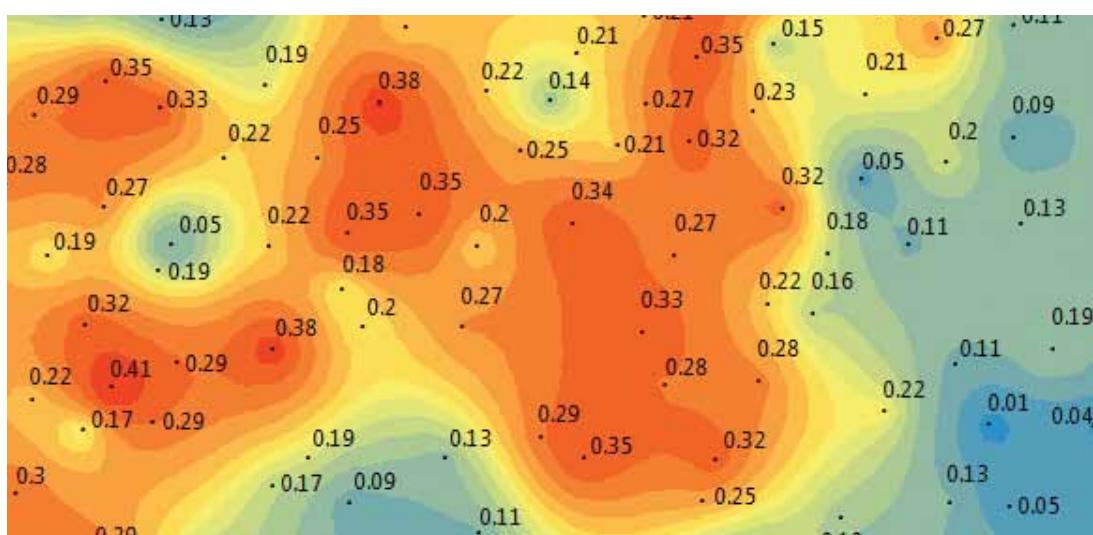


Figure 2: Empirical Bayesian Kriging



Figure 3: K-means clustering

Artificial intelligence is applied to GIS in areas such as image classification, prediction and segmentation.

## **Image classification using Support Vector Machine**

When you look at a satellite image, it's not always easy to know whether you are looking at trees or grass, at roads or at buildings. So imagine how difficult it would be for a computer to differentiate between these.

Support Vector Machine (SVM) is a machine learning technique that takes classified data and looks at the extremes. It is a discriminative classifier formally defined by drawing a decision boundary line based on the data called a hyperplane. The data points that the hyperplane margin pushes up against are the support vectors.

Support vectors are important because they are the data points closest to the opposing classes and all other training points can be ignored in the model. When you feed SVM the training data samples of trees and grass, it builds the model, generating a decision boundary of its own. But the results of this supervised classification aren't perfect and algorithms still have a lot more learning to do. They still need to work on features like roads, wetlands and buildings. As algorithms get more training data, they will eventually improve the classification.

## **Image prediction using Empirical Bayesian Kriging (EBK)**

Geostatistics is the generic name for a family of techniques that is used for the mapping of surfaces from limited sample data and the estimation of values at unsampled locations. Geostatistical estimation is a two-stage process:

- i. Studying the gathered data to establish the predictability of values from place to place (locations) in the study area, which results in a graph known as a semi-variogram.
- ii. The process of estimating values at those locations that have not been sampled is known as kriging. The basic kriging technique uses a weighted

average of neighbouring samples to estimate the unknown value at a given location. It estimates weights based on the variogram. And the quality of the estimated surface is reflected in the quality of the weights. More specifically, you want weights that give an unbiased prediction and the smallest variance.

Unlike kriging that fits one whole model for an entire data set, Empirical Bayesian Kriging (EBK) simulates at least one hundred local models by sub-setting the whole data set. Because the model can morph itself locally to fit each individual semi-variogram using kriging methodology, it overcomes the challenge of stationarity. Each semi-variogram varies from each other. In the end, it mixes all the semi-variograms for a final surface. You can see the trends in the resulting data and use that to justify your selection.

## **Image segmentation and clustering with K-means**

By far, the K-means clustering algorithm is one of the most popular methods of clustering data. This is an unsupervised algorithm and is used to segment the interest area from the unlabelled data into the number of groups represented by the variable K. But before applying the K-means algorithm, the first partial stretching enhancement is applied to the image to improve its quality. The algorithm iteratively assigns each data point into one of the K groupings, based on a similarity of features. For example, similarity can be based on spectral characteristics and location. In an unsupervised classification, the K-means algorithm first segments the image for further analysis. Next, each cluster is assigned a land cover class.

However, GIS can use clustering in other unique ways. For example, it can be used for crime mapping—the data points could represent crime locations that can be clustered as spots of crime.

## **AI applications in GIS**

The applications of artificial intelligence in GIS are in a number of sectors, including those that use location and GIS. Ride-sharing companies, logistics, agriculture, surveying, and infrastructure are some of the prominent examples.

**Predictions and remote monitoring:** AI based GIS applications can handle complex weather and climate imagery data patterns that humans can't process on a large scale and in real-time. Such GIS applications can result in solutions for problems arising from climate change, air pollution, water pollution and forest management. This framework can optimise the use of land data, agriculture data, regional based crop data, etc, to maximise the economic benefits for society.

**Internet of Things (IoT):** Almost every connected device that uses GIS application software can use artificial intelligence as a platform to predict, adapt, learn and make decisions for end users. The neural geospatial data system can be the core engine for self-driving vehicles and drones to adapt and manipulate an environment in real-time, using all kinds of GIS data.

**Geographic consumer behaviour:** In today's competitive world, understanding the consumer is the focus of any organisation. AI based structures can learn and predict consumer behaviour using geographic data and come up with specialised regional solutions. Ride-sharing companies like Uber, Ola, etc, can take feedback from customers and process the data to find out the density of cars and the availability of drivers. In logistics and supply chains, artificial intelligence can plug the gaps and gather more accurate location information that can streamline product delivery and save time.

Overall, in the realm of business, artificial intelligence based systems can create an ecosystem around various GIS applications that would substantially improve planning, resource allocation and decision-making – predicting the surge in demand and supply, identifying the prospects of high and low margins, multiplying supply chain efficiency, and optimising service delivery. The scope is endless.

# No More a Buzzword, AI has Arrived and Reached Human Parity

*Artificial Intelligence (AI) has the power to transform our lives. Today, more and more AI systems are being open sourced, which is expected to lead to many more interesting innovations. To understand the potential of AI and the role open source plays in it, Ankita K.S. from the EFY Group chatted with Sandeep Alur, director – partner engagements, Microsoft, at Open Source India 2018. Excerpts follow...*

## **How do you think open source has evolved over the past few years in India?**

Two things are happening mainly. One is the cultural shift that is happening because of the growth in the number of startups and entrepreneurs in the country. These startups innovate and build intellectual properties (IPs), and are starting to open source a few endpoints with the intent to seek contributions from the community, to make it better. Because of this culture, open source has taken off in a big way. Second is the push from the global IT giants in the industry and from the academia. These are the two driving factors that have led to the popularity of open source in the past few years in the country.

## **Personally, what is your favourite open source technology?**

I love machine learning (ML). In fact, I have been working on some projects for which I wanted open source code for certain key functionalities. It was already available on GitHub (open sourced), so it was an easy start getting into what we wanted to build. My favourite programming language would be Python due to its dominant community backing, and the fact that a bulk of what we need for ML is available via OS libraries. So, if you want to work on AI/ML and you are good at Python, you are at the starting line already.

## **At which stage of AI development is the industry today?**

AI is no more a buzzword. It has been around for decades and it has even been taught in academia for a long time. We only talk about it today because machines have reached human parity on a few things like computer vision, speech, etc. Machines are getting close to or better than human beings in doing certain things. It is the only stream that will help us predict or prescribe and take timely action. It is like new age astrology. AI has already made a backdoor entry into most of the industries, in various forms. As consumers, you may eventually experience it as the features/service mature.

## **What is ubiquitous computing and how is it changing things today?**

Ubiquitous computing enables computing anywhere, anytime and on any device. For example, let's imagine that I am listening to music in my car and when I reach home, I want the same music to continue playing automatically on another device in the house. Now, with ubiquitous computing, I need not instruct or tell a device at home what to do. This is how seamless we all want our lives to get.

Today, if you look at any autonomous vehicle, it has been trained and tested for millions of miles on the road. It has been trained to the extent that if it comes to the zebra crossing, it will stop and wait for people to cross. But is it possible to predict what people may be thinking while crossing the road? Will they decide to stop midway, while crossing? That is the next level of challenge in making computers think and predict such actions too.

## **What are the various fields around AI where innovation is happening?**

The base promise of AI is to make computers behave like humans by making them understand the world. This can be done in two ways — one is perception and the other is cognition. Perception is related to vision and speech, while cognition is related to language and knowledge. Predominantly, when you look at AI, the innovations and developments

are happening around all the four aspects—vision, speech, knowledge and language.

The key to the success of AI is data. Data volumes are expected to increase to 44ZB in 2020. There is a study that talks about companies willing to take up AI use cases, but states that the unavailability of data is a dampener. On the other hand, one of the studies by Gartner indicates that 80 per cent of data that is required for an enterprise to develop AI appears to exist within the company itself. These are interesting times, with innovations happening all around, and AI as a field is emerging at a very fast pace.

## **Can you briefly tell us how Microsoft is helping the open source community?**

Microsoft has been contributing to open source for many years, but we are probably only talking about it now. You must have heard recently that Microsoft joined the Open Invention Network (OIN), an open source patent group designed to help protect Linux from patent lawsuits. With this, over 60,000 patents were open sourced and made available to OIN members. Ahead of any other contributor, even before the announcement of Microsoft acquiring GitHub, the company has been a leading contributor to the OSS world. There are interesting times as we continue to contribute as much as we can to the open source community.

## **How helpful did you find OSI 2018 to open source contributors?**

This is the third year I am attending OSI and each year, I see more active participation from the true believers of OSS. I do not see any other conference in India, where open source is front and central. I love OSI because this is one platform where you get to interact with many who are already in the open source world and want to understand more about Microsoft's stance and contribution. It is also a great platform to get genuine feedback from the industry.

# “We Are Embedding AI Into All of our cloud services”

*Oracle Cloud Platform is a Platform-as-a-Service (PaaS) offering. It delivers a broad selection of next-generation, enterprise-grade cloud computing services, spanning SaaS, PaaS and IaaS. Oracle has embedded innovative technologies in every aspect of its cloud, enabling organisations to reimagine their businesses, processes and customer experiences. Siddhartha Agarwal, group VP - product management and strategy, Oracle Cloud Platform, in a face-to-face interaction with Rahul Chopra, editorial director, EFY Group, throws light on Oracle's cloud offerings, which includes the world's first and only autonomous database, on IoT, and a lot more.*

## How important is the cloud for Oracle's business?

The cloud is not a new thing; it has been around for years. However, it's only now that enterprises are looking to take advantage of all that the cloud has to offer so they can gain value faster, become more agile and be able to create applications or differentiate themselves from each other. Nobody understands the enterprise software landscape like we do, and we're fully geared to become the most preferred enterprise cloud provider. The beauty of the cloud is that it's a great equaliser, in that the same modern, enterprise-grade cloud can be used by mid-sized businesses as well as startups on a pay-as-you-go model. We already have several cloud customers of all sizes, across industries, benefitting from our modern cloud offerings.

While the cloud is our Number 1 priority, we understand that different customers are at different stages in their cloud journey. We offer them the flexibility and choice of their cloud deployment based on their organisational readiness (public/hybrid/Oracle Cloud at Customer, on-premises).

**I am not entirely sure, as of now, whether developers will name Oracle as the key cloud player. Are you following a strategy that approaches customers directly rather than taking the developer route? Or is it because Oracle entered this arena a little late?**

**What's your take on this?**

Oracle is the only full-stack, enterprise-grade cloud provider (offering SaaS, PaaS and IaaS) and our customers are increasingly turning to us to future-proof their business. We are building machine learning (ML) and intelligence into each one of our cloud services. If you look at the Gartner report, you will see that we are among the top four providers in the cloud space. Our installed base of customers is extremely interested in understanding what Oracle is offering. They also look into how they can leverage Oracle, not just for the database but also for app integration, data integration, security, analytics, management and more, using the cloud.

On the SaaS side, various analyst reports indicate that we're well on our way to becoming the world's No. 1 cloud applications player. We started on the infrastructure and platform delivered as service pieces four to five years ago. We are talking a lot about our capabilities, especially in the context of customers, who have extracted some meaningful value from our cloud. Our autonomous database is a good example.

More and more developers are realising why they should use Oracle Cloud — given our modern, open, easy cloud platform.

**Why should developers prefer Oracle rather than your counterparts? What's unique about Oracle's offerings?**

At the end of the day, developers think of building an application that connects into some other applications. They might be trying to modernise an on-premise application or build completely new capabilities. And what we have done is — we have created a differentiated value for them at the database tier. They don't have to think about managing and operating a database.

If anyone wants to build a chatbot, we provide a very easy way to do so, with intelligence built into the chatbot — from integration to the back-end system support.

Developers who are building enterprise grade applications should be looking at what we're doing to make their life simpler. If you look at the blockchain, it is a very popular technology, as is Hyperledger Fabric — something that is open source. What we have done is that we have built an enterprise grade blockchain cloud service, so the security, scalability, and performance is the same as you get from the enterprise grade Oracle database. This helps developers, as they now only have to think about smart contracts that they want to write with APIs. They don't have to worry about any of the underlying infrastructure. In short, our focus is to provide a bunch of tools to the developers so that they're self-sufficient.

## **Are blockchain and AI the components that developers choose from your cloud services?**

First of all, we are embedding AI into all of our cloud services. The autonomous database and analytics have ML built in. Therefore, the developers get access to AI just by using our cloud services. They can pick and choose themselves, for example, management and monitoring applications on the cloud. They do not need to know thresholds. In addition, we are providing the developers the ability to build models, compare models, expose models as APIs, etc. We are also providing these as services to developers. So we're seeing developers use a wide range of our cloud services.

## **What are you doing to support the developers?**

We are doing a few things to support them. First, three years ago, we created a targeted platform to help developers understand the openness of our platform — how intelligent it was and how to easily use it. As part of that initiative, we created a dedicated developer site: [developer.oracle.com](http://developer.oracle.com), where they could get access to all our resources that they were interested in. Second, the goal of the Oracle Code event series — a free knowledge exchange platform

for developers, by developers — is to educate them around cutting-edge technologies. These events have a good mix of Oracle as well as non-Oracle speakers. This is helping developers get a 360 degree view of our technology.

We are also empowering developers to try our cloud as a self-service model. However, there is always a connection for them to ask for help. In addition, we have created educational modules for developers to learn on their own.

## **What is the response to Oracle's cloud offerings in the Indian market?**

The response has been great. India is a key growth market for us within the APAC region, and we're closely working with customers to help them in their journey to the cloud. There are a lot of Indian customers who are using our cloud and realising significant value. One good example is Bajaj Electricals. The company used Oracle's AI enabled intelligent bot service, and was able to build and pilot an intuitive, consumer-friendly chatbot in just under three weeks. This has enabled the company to reduce its call centre headcount significantly, as a vast number of consumer enquiries are getting addressed via the chatbot.

## **Can you tell us something about the type of solutions you're offering?**

We continue to push the envelope of what's possible. Our complete focus is on creating value for our customers. For example, our SaaS applications have the capability to harness data intelligently to enable a customer service representative to find out the next best offer, a sales representative the next best opportunity and a recruiter the next best candidate to invest in. We are building that kind of intelligence into our SaaS applications by leveraging first party data, third party data and ML models. Our focus is more on how one can get actionable intelligence as opposed to just offering the platform to build a whole bunch of different intelligent pieces. Similarly, we are embedding intelligent services into our management and analytics offerings. That's how we are delivering the intelligence aspect of our platform.

The Oracle Autonomous Database, which is the world's first and only 'self-driving, self-securing and self-repairing' database, is one of the most important and successful new product introductions in our company's 40-year history. Powered by advanced machine learning, the Oracle Autonomous Database can scale itself, patch itself, manage itself — and it is continuously learning. In addition, you don't have to do indexing. A developer can just focus on building the next killer app, and not get bogged down with database management or any other 'less creative, mundane' tasks.

Codeinks, a Bengaluru based ecommerce software product startup, was looking to extract deep levels of insight into user behaviour and past performance to do better future predictions of trends. Oracle Autonomous Data Warehouse Cloud made a huge difference and the company was able to achieve performance improvements ranging from 30 to 50% on average for its defined scenarios.

## **How do you see the adoption of AI and ML technologies on a large scale?**

I think there are two things that have made AI and ML more pervasive in recent months. First, the amount of data that's being generated and collected. Second, the easy availability of the heavy duty computing capacity needed to make sense of this data, quickly. GPUs and Flash memories that can run lots of applications in memory are now available. There have been high bandwidth connections between various nodes – whereby nodes can talk to each other. These are some of the developments that have fundamentally made ML and AI possible at scale. I think the place where there is a broader adoption of ML is in the conversational paradigm of AI, where people engage in intelligent conversations through the messaging interface.

Like I mentioned earlier, we're embedding intelligence into each of our cloud services, and at times customers don't even know or realise this. If we revisit the autonomous database example, we are able to patch the database, scale it, optimise the queries, as well as create or remove the indexes that need it. And that's where the mass adoption of AI happens,

because anyone who is building the cloud application and leveraging any of these cloud services can start seeing the value.

**You have mentioned the autonomous database. Is this only on the cloud or is an on-premise version also available?**

At the moment, the Oracle Autonomous Database is only on the cloud.

However, we are working on getting the on-premise version as well, in the coming months.

**Is the autonomous database open, or is it available in Oracle's data centres only?**

We have just added more intelligence to the Oracle Database and crafted the Oracle Autonomous Database. While it's not open source based, per se, the Oracle Database is one of the most flexible databases out there — it is extremely open in a way that it can run anywhere. If you look at some of our competitors who offer cloud based services, they're tied strictly to their cloud and can't run anywhere else. So, you don't have the portability — the ability to run the Oracle content either in the public cloud or within your data centre or in a third-party cloud.

**There is a debate regarding the law that mandates data to reside within the country in which it has been collected. What are your thoughts on that?**

We'll abide by all government regulations, not just in India, but in every country/region we operate in. We believe Oracle Cloud at Customer is tailor-made for companies (public or in the corporate sector) looking to enjoy all the benefits of a public cloud environment, but behind their own firewall (inside their own data centre). Further, we've already announced that we'll soon be opening our India data centre later this year; so we're well prepared to address local market opportunities and emerge as a leading cloud provider in India.

**Could you explain concepts like 'universal credits' and 'Bring your own Licence'?**

One of the key things we realised, based on the feedback we got from our

customers, is that that they want flexibility with what they use, when they use it and how much of it they use. If you want them to commit to how much database capacity they want to buy, how much of integration capacity they want to invest in upfront, or how much of mobile cloud service they want to consume, then they actually don't know because utilisation in the cloud could suddenly scale up or not happen at all. Therefore, we introduced this notion of 'universal credits (UC)' pricing, whereby we give them the flexibility to buy a pool of credits from Oracle, with which they are entitled to use all our cloud, IaaS and PaaS services. Their usage is metered and adjusted against their universal credits. In addition, when we release a new software or new cloud service, they are entitled to use those too.

We are making it easier for a customer to be able to use the cloud without worrying how much they're using, what they're using, etc. This is what our universal credits are about. Think of it as a virtual prepaid wallet you can use to consume any Oracle platform/infrastructure cloud service, including mixing and matching, based on the quantum of universal credits you have opted for.

With 'Bring Your Own Licence (BYOL)', we want to enable portability from on-premise licences to the cloud. If customers decide to migrate to the cloud from their on-premise usage model, they don't have to pay double the cost for the licence. If they have licences that are unused (on-premises), they can actually bring that licence to the cloud, and there is no reason to charge them for the same licence again. Thus, the subscription price of that cloud service comes down significantly. This means they have to pay only for the infrastructure upon which it is running, and the automation and operations that we do. So that's the logic behind 'BYOL'.

## **What is the role of Oracle's Indian centre with respect to development and other work?**

Oracle India is home to our largest employee base outside of the US. Our teams in India are involved with all our product lines, and there's a lot of great collaboration happening. Oracle India is the only organisation outside Oracle's headquarters in California to represent all divisions including sales, marketing,

consulting, as well as support and education operations for domestic and global clients. In fact, India accounts for the organisation's largest research and development investment outside US.

## **With respect to developers, what is the USP of Oracle's cloud platform?**

The developers can easily add intelligence to their platform. They can leverage a very open platform that can run in multiple environments, and not just within the public cloud. They can run several functions on-premises or within the public clouds. They're able to deliver hybrid capabilities to an application that can either work on-premises or on the public cloud. Our goal is to give them better functionality out-of-the-box — that is pre-assembled so that they can focus on their business differentiators and accelerate innovation.

At the end of the day, developers need rich and robust platform capabilities. They want the ability to choose any programming language, database, operating system, virtualisation technology and developer tool, including open source and third-party options. Oracle Cloud Platform includes key capabilities for embedded AI and machine learning to help developers build new AI based applications, and to enable IT and operations to automate the manual steps in provisioning, patching, tuning, and more. With the Oracle AI platform cloud service, developers and data scientists are empowered to develop and deploy machine learning models, and to manage the complete data science life cycle on their preferred open source frameworks.

## **Coming to the Internet of Things (IoT) - are there platforms specifically designed for IoT applications?**

A few years ago, we started a platform on which one could connect the gateways, gather the data coming from different sensors and make intelligence decisions at the edge. We can now aggregate the data into the public cloud, do predictive analytics on that data, and connect the devices to the back-end applications.

After building this platform, we realised that when you look at the usage of IoT, most people (for example, a person on the manufacturing plant floor or someone who is responsible for the servicing of goods) are not interested in building an application on the platform. They just want to use IoT and get the value out of it. Therefore, we built a set of SaaS applications on top of this platform. This platform is still available for anyone who wants to build an IoT app on it.

As of now, we have released a set of four SaaS applications as part of our IoT focus.

- **Asset monitoring:** This gives users the ability to monitor the assets, create a geofence, take sensor data from these assets, and get health information.
- **Production line monitoring:** When you are on a manufacturing floor, you can use this app to get insights from various sensors — the yield of a production line, and if the yield is going down in one line then how you can make it reconfigure itself, etc.
- **Fleet monitoring:** When there is a distribution of vehicles, this app helps you monitor these vehicles — how to monitor the drivers' behaviour, how you will figure out where the vehicles are, etc.
- **Connected workers:** This gives users the ability to know where the workers are, what they are doing, etc.

If you look at these applications carefully, all of them have the same kind of platform — they all need sensor data, predictive analytics, a connection to back-end systems, secure authentication and communication from devices to apps, etc. This list comprises the first four apps we have built. We are now taking the blockchain and combining it with IoT applications.

# The Future of Machine Learning

*The 21st century has seen the rise of machine learning (ML) and artificial intelligence (AI). Many mundane tasks are today done by machines. ML and AI have pervaded all aspects of human life, without exception. So will machines soon overtake us in an Orwellian manner? Read on for some insights into this worrying question.*

Data is the currency for the new age, particularly with the advent of machine learning. From Google to Tencent, companies are locked in a battle to be the first to create and combine the most efficient models in an attempt to create artificial general intelligence. As businesses scramble to build the best products, they have created a wave of artificial intelligence (AI) that has swept the world off its feet. AI has become more than just a buzzword as technologies have emerged that are capable of changing the world, though people are still unclear about whether this will be for the better or worse. What we do know, however, is that a revolution is under way and it will leave nothing untouched.

From social media to kitchen faucets, companies are targeting the most mundane of products to incorporate some semblance of human intelligence to purportedly make life easier for us.

We have artificial intelligence embedded the in core of novel applications in fintech, customer support, sports analytics and cyber security. The state-of-art methods in use include neural networks, support vector machines, random forests, decision trees, Adaboost, and others. Mundane tasks have been automated already and more specialised ones are on their way out.

However, the 'state-of-art' technologies often encourage a false sense that AI or ML enabled strategies are production-ready. In reality, their deployment usually takes a few months to a few years in a live setup. For instance, in spite of all our advances in natural language processing, we are still a long way from truly

intelligent chat bots that can understand and hold a conversation with a human being. Sure, Google had a neat demo of virtual assistance at Google I/O 2018, but the company still has its own breakpoints and faults that are slowly being ironed out.

Michael Jordan, professor of EECS and statistics at Berkeley, claims that we are only beginning to understand the true depth of AI and its potential to impact humankind. Deep learning has produced results that are orders of magnitude better than conventional machine learning algorithms. Deep neural networks have found widespread application in vision, recognition and speech, producing far better results than were seen before.

## Machine learning applications

Machine learning is being used in various domains that include niche research areas as well as broad segments of customers. It relies on data at its core, since the nature and quantity of data collectively contribute to the performance metrics for the model. Recent focus on deep learning has seen the emergence of convolutional neural networks, recurrent neural networks and long short-term memory as key drivers of R&D efforts in this area. The fundamental

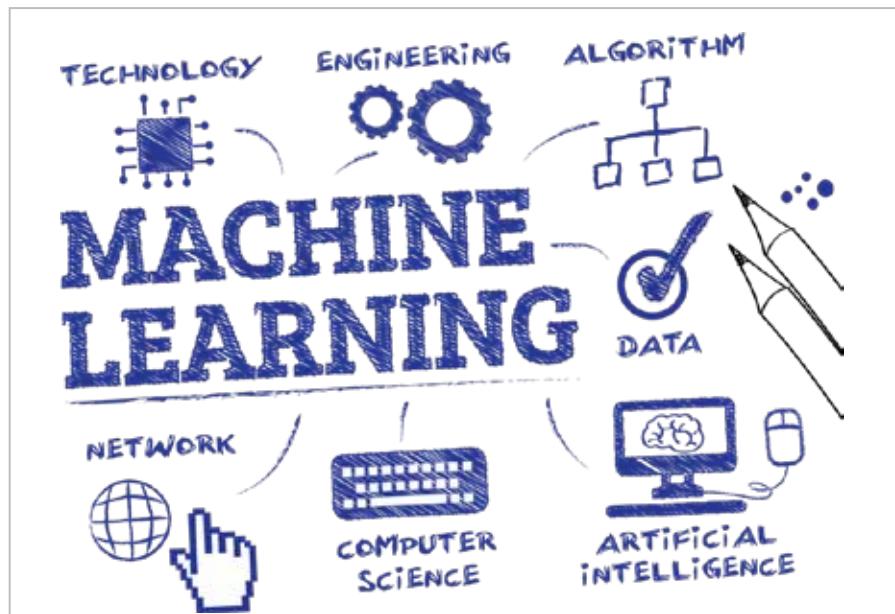


Figure 1: Facets of machine learning (source: [forbes.com](https://www.forbes.com/sites/forbestechlist/2017/09/11/the-future-of-machine-learning/#ixzz5WzXGZLqf))

sciences have found an analytical toolkit in open source machine learning libraries that are used to code and replicate models.

For fintech, machine learning algorithms drive trading strategies based on game theory and reinforcement learning. In addition, analyses generated by models can evaluate existing and past transactions to detect a possible pattern in investments.

Medical applications vary from detection to diagnosis, but with varying degrees of accuracy. However, digitisation of health records and remote services has indeed made the world a really small place.

Recommendation systems are ubiquitous across e-commerce, video streaming and e-shops. Home automation is gaining traction as virtual assistants improve their learning systems.

Agricultural farms are now monitored and crop diseases detected via machine learning—a project that presents a huge opportunity, especially for agrarian economies such as India. Proof-of-concepts have been set up around Europe already.

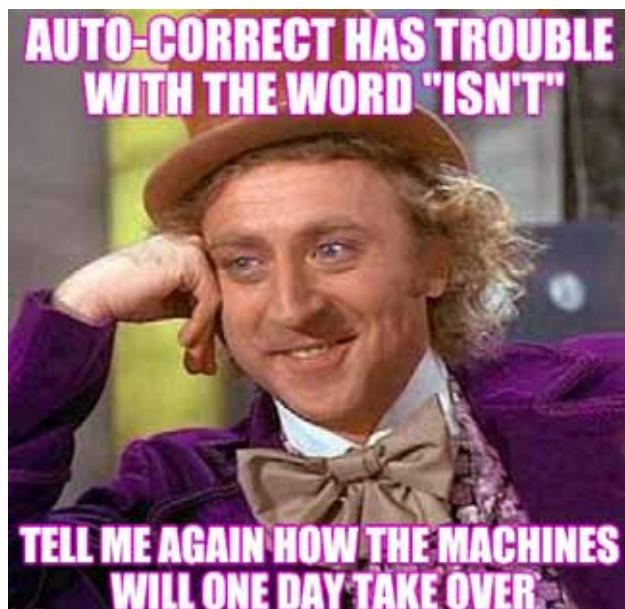


Figure 2: The true state of machine learning? (source: [imgflip.com](http://imgflip.com))



Figure 3: Machine learning is here to stay!

Machine learning has thus impacted a far broader set of sectors than just computer science.

## Future technologies

While there is considerable interest surrounding ML and AI, I will attempt to walk you through this territory, leaving out the hype. From a human perspective, it is quite plausible to accidentally create artificial general intelligence that could impact all of humanity. However, we are still years away from such a possible misfortune.

Machine learning has some exciting new developments and future plans. The launch of the ONNX format that allows the interchange of deep neural networks across different libraries, serves as a prelude to the cooperation that is needed to ensure quick, easy and meaningful experiences for all. It is envisioned to take on a central role as research and development of models progresses.

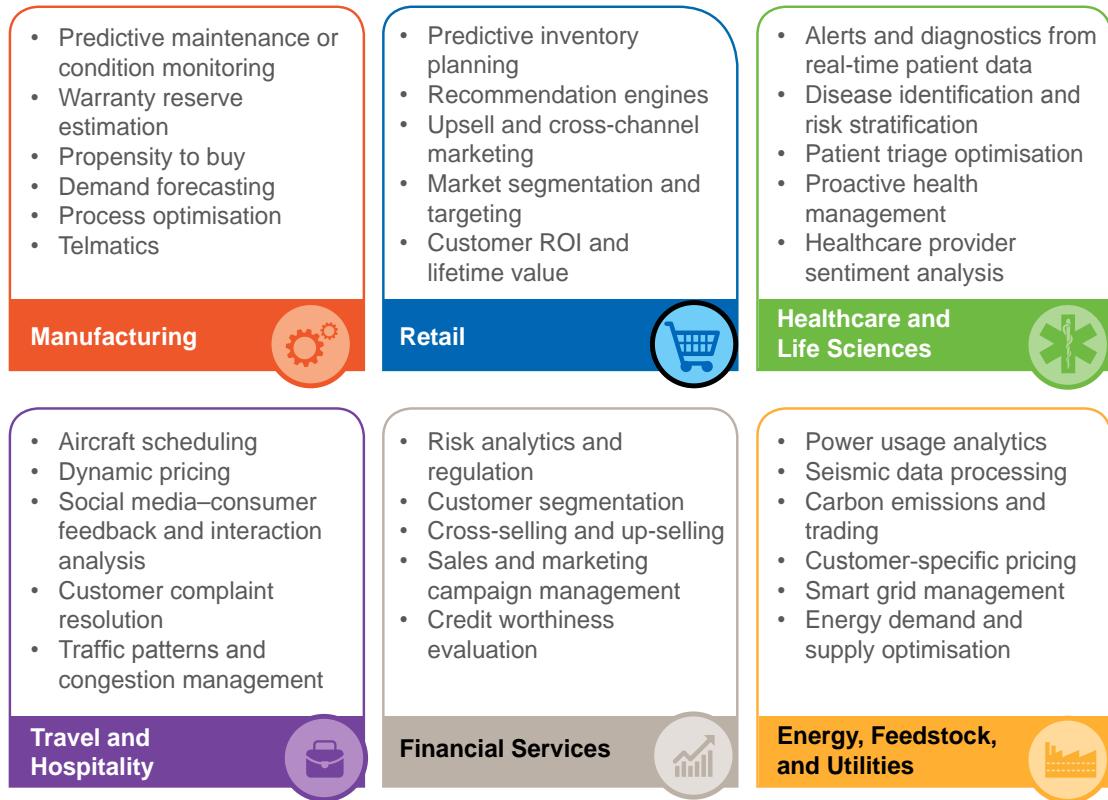


Figure 3: The applications of machine learning (source: *forbes.com*)

Reinforcement learning is a largely unexplored space that is currently seeing much activity. Examples include deep-q networks, playing Atari and other video games, or learning to walk, run, and jump in a virtual environment.

Truly self-driving vehicles (with Level 5 autonomy), that are still limited to the design desk, should enter the market after being subjected to an extensive set of safety regulations, given the past incidents with Uber, Tesla, Waymo and others.

Quantum AI is a relevant subject area that could take off in the near future, though in its current state it is restricted mainly to theories and hypotheses. It could offer tremendous speed-ups and reinvent the complete pipeline that currently relies on training models on CPUs or GPUs.

Improvements in unsupervised learning may come about as strategies for

clustering improve in efficiency. Similar progress will come about in defining the evaluation metrics for clustering that usually relies on assessment of its peers as a group.

Ethics and AI policy come to the fore when you give a company access to all your data, since there is a huge potential for this to be misused. End-to-end encryption, security, anonymity and abstraction of data then gain importance. With the European General Data Protection Regulation rules kicking in, it has forced businesses to rethink their business practices and explicitly seek permission before using our data.

Personalisation will run deeper than usual, with data profiling at the root of the user's 'profile' based on likes, views and browsing patterns.

Reproducibility and interpretability are major concerns, especially when treating neural networks as black boxes and seeking to explain the results of experiments, or when attempting to replicate large scale training without the original datasets. Attempting to derive intuitive information about the intermediary steps, learned features and other aspects of the training of a neural network is not easy.

The brain-computer interface will soon be realised, though may be not perfectly. Emulating the constructs underpinning an actual human brain was the original motivation behind developing neural networks; so only the future can tell when this will materialise.

In the years to come, it is possible that we will be able to create some semblance of general intelligence. This can change the world or even destroy it—let's just keep our fingers crossed that we will all survive!

## **Part-II**

# **Developing And Implementing AI & ML**

# Machines Learn in Many Different Ways

*This article gives the reader a bird's eye view of machine learning models, and solves a use case through Sframes and Python.*

'Data is the new oil'—and this is not an empty expression doing the rounds within the tech industry. Nowadays, the strength of a company is also measured by the amount of data it has. Facebook and Google offer their services free in lieu of the vast amount of data they get from their users.

These companies analyse the data to extract useful information. For instance, Amazon keeps on suggesting products based on your buying trends, and Facebook always suggests friends and posts in which you might be interested. Data in the raw form is like crude oil—you need to refine crude oil to make petrol and diesel. Similarly, you need to process data to get useful insights and this is where machine learning comes handy.

Machine learning has different models such as regression, classification, clustering and similarity, matrix factorisation, deep learning, etc. In this article, I will briefly describe these models and also solve a use case using Python.

**Linear regression:** Linear regression is studied as a model to understand the relationship between input and output numerical values. The representation is a linear equation that combines a specific set of input values ( $x$ ), the solution to which is the predicted output for that set of input values. It helps in estimating the values of the coefficients used in the representation with the data that we have available. For example, in a simple regression problem (a single  $x$  and a single  $y$ ), the form of the model is:

$$y = B_0 + B_1 \cdot x$$

Using this model, the price of a house can be predicted based on the data available on nearby homes.

**Classification model:** The classification model helps identify the sentiments of a particular post. For example, a user review can be classified as positive or negative based on the words used in the comments. Given one or more inputs, a classification model will try to predict the value of one or more outcomes. Outcomes are labels that can be applied to a data set. Emails can be categorised as spam or not, based on these models.

**Clustering and similarity:** This model helps when we are trying to find similar objects. For example, if I am interested in reading articles about football, this model will search for documents with certain high-priority words and suggest articles about football. It will also find articles on Messi or Ronaldo as they are involved with football. TF-IDF (term frequency - inverse term frequency) is used to evaluate this model.

**Deep learning:** This is also known as deep structured learning or hierarchical learning. It is used for product recommendations and image comparison based on pixels.

Now, let's explore the concept of clustering and similarity, and try to find out the documents of our interest. Let's assume that we want to read an article on soccer. We like an article and would like to retrieve another article that we may be interested in reading. The question is how do we do this? In the market, there are lots and lots of articles that we may or may not be interested in. We have to think of a mechanism that suggests articles that interest us. One of the ways is to have a word count of the article, and suggest articles that have the highest number of similar words. But there is a problem with this model as the document length can be excessive, and other unrelated documents can also be fetched as they might have many similar words. For example, articles on football players' lives may also get suggested, which we are not interested in. To solve this, the TF-IDF model

	URI	name	text
	<http://dbpedia.org/resource/Digby_Morrell> ...	Digby Morrell	digby morrell born 10 october 1979 is a former ...
	<http://dbpedia.org/resource/Alfred_J._Lewy> ...	Alfred J. Lewy	alfred j lewy aka sandy lewy graduated from ...
	<http://dbpedia.org/resource/Harpdog_Brown> ...	Harpdog Brown	harpdog brown is a singer and harmonica player who ...
	<http://dbpedia.org/resource/Franz_Rottensteiner> ...	Franz Rottensteiner	franz rottensteiner born in waidmannsfeld lower ...
	<http://dbpedia.org/resource/G-Enka> ...	G-Enka	henry krvits born 30 december 1974 in tallinn ...
	<http://dbpedia.org/resource/Sam_Henderson>	Sam Henderson	sam henderson born october 18 1969 is an

Figure 1: The people data loaded in Sframes

	URI	name	text
	<http://dbpedia.org/resource/Barack_Obama> ...	Barack Obama	barack hussein obama ii brk husen bm born august ...

[? rows x 3 columns]

Note: Only the head of the SFrame is printed. This SFrame is lazily evaluated.  
You can use sf.materialize() to force materialization.

Figure 2: Data generated for the Obama article

comes in. In this model, the words are prioritised to find the related articles. Let's get hands-on for the document retrieval. The first thing you need to do is to install GraphLab Create, on which Python commands can be run. GraphLab Create can be downloaded from <https://turi.com/> by filling in a simple form, which asks for a few details such as your name, email id, etc. GraphLab Create has the IPython notebook, which is used to write the Python commands. The IPython notebook is similar to any other notebook with the advantage that it can display the graphs on its console.

```

localhost:8888/notebooks/Document%20Retrieval.ipynb
File Edit View Insert Cell Kernel Help

Sort the word counts for the Obama article

Turning dictionary of word counts into a table

In [17]: obama_word_count_table = obama[['word_count']].stack('word_count', new_column_name=

Sorting the word counts to show most common words at the top

In [18]: obama_word_count_table.head()

Out[18]:
  word    count
  normalize   1
  sought      1

```

Figure 3: Sorting the word count

Open the IPython notebook which runs in the browser at <http://localhost:8888/>. Import GraphLab using the Python command:

```
import graphlab
```

Next, import the data in Sframe using the following command:

```
peoples = graphlab.SFrame('people_wiki.gl/')
```

**Compute TF-IDF for the corpus**

To give more weight to informative words, we weigh them by their TF-IDF scores.

```
In [20]: people['word_count'] = graphlab.text_analytics.count_words(people['text'])
people.head()
```

URI	name	text	word_count
<http://dbpedia.org/resource/Digby_Morrell> ...	Digby Morrell	digby morrell born 10 october 1979 is a former ...	{'since': 1L, 'carl': 1L, 'being': 1L, '20': 1L}
<http://dbpedia.org/resource/Alfred_J._Lewy> ...	Alfred J. Lewy	alfred j lewy aka sandy lewy graduated from ...	{'precise': 1L, 'the': 1L, 'closely': 1L}
<http://dbpedia.org/resource/Harpdog_Brown> ...	Harpdog Brown	harpdog brown is a singer and harmonica player who ...	{'just': 1L, 'issu': 1L, 'mainly': 1L}
<http://dbpedia.org/resource/Franz_Rottensteiner> ...	Franz Rottensteiner	franz rottensteiner born in waidmannsfeld lower ...	{'all': 1L, 'bauforschung': 1L}
<http://dbpedia.org/resource/G-Enka> ...	G-Enka	henry krvits born 30 december 1974 in tallinn ...	{'legendary': 1L, 'gangstergenka': 1L}
<http://dbpedia.org/resource/Sam_Henderson> ...	Sam Henderson	sam henderson born october 18 1969 is an ...	{'now': 1L, 'curre': 1L, 'less': 1L, 'be': 1L}
<http://dbpedia.org/resource/Aaron_LaCrate> ...	Aaron LaCrate	aaron lacrate is an american music producer	{'exclusive': 2L, 'produced': 1L, 'tr': 1L}

Figure 4: Compute TF-IDF for the corpus

**Examine the TF-IDF for the Obama article**

```
In [23]: obama = people[people['name'] == 'Barack Obama']
```

```
In [24]: obama[['tfidf']].stack('tfidf',new_column_name=['word','tfidf']).sort('tfidf',ascending=False)
```

word	tfidf
obama	43.2956530721
act	27.878222823
iraq	17.747378588
control	14.8870608452
law	14.7229357618
ordered	14.5333739509
military	13.1159327785

Figure 5: TF-IDF for the Obama article

To view the data, use the command:

```
peoples.head()
```

This displays the top few rows in the console.

The details of the data are the URL, the name of the people and the text from Wikipedia.

I will now list some of the Python commands that can be used to search for related articles on US ex-President Barack Obama.

1. To explore the entry for Obama, use the command:

```
obama = people[people['name'] == 'Barack Obama']
```

2. Now, sort the word counts for the Obama article. To turn the dictionary of word counts into a table, give the following command:

```
obama_word_count_table = obama[['word_count']].stack('word_count', new_column_name = ['word', 'count'])
```

3. To sort the word counts to show the most common words at the top, type:

```
obama_word_count_table.head()
```

4. Next, compute the TF-IDF for the corpus. To give more weight to informative words, we evaluate them based on their TF-IDF scores, as follows:

```
people['word_count'] = graphlab.text_analytics.count_words(people['text'])
people.head()
```

5. To examine the TF-IDF for the Obama article, give the following commands:

```
obama = people[people['name'] == 'Barack Obama']
obama[['tfidf']].stack('tfidf',new_column_name=['word','tfidf']).sort('tfidf',ascending=False)
```

Words with the highest TF-IDF are much more informative. The TF-IDF of the Obama article brings up similar articles that are related to it, like Iraq, Control, etc.

Machine learning is not a new technology. It's been around for years but is gaining popularity only now as many companies have started using it.

# Tools that Accelerate a Newbie's Understanding of Machine Learning

*The world of machine learning (ML) and deep learning (DL) is a fascinating one. Many newbies would like to dabble in this field but have some inhibitions. This article introduces readers to some of the best tools to kickstart their journey into the ML/ DL domain.*

Machine learning is a term that is applied to a broad range of topics which have one thing in common – the use of algorithms and other statistical models to improve the performance of a particular task. All machine learning (ML) models are used to build a mathematical model of the data provided, to help the user predict or make decisions with a very high level of accuracy; thereby considerably reducing the strain of manually sifting through the data. Currently, there are several fields in which ML is used extensively, from filtering out emails and computer vision, to choosing a smart assistant. Readers are urged to spend some time researching only the topics that they are interested in and not get overwhelmed or distracted by the countless other applications of ML.

Deep learning can be considered a subset of machine learning, though to call it that really does not do it justice. Currently, most applications have moved on from the somewhat old fashioned ML algorithms to employ deep learning (DL) algorithms. The latter provide much more support for the building of more complex mathematical models which cannot be provided by ML. Deep learning algorithms employ multiple ML algorithms in a manner vaguely inspired by how the human brain works, and hence the term 'neural networks'. Deep learning algorithms have been developed to the point where it has been proved that computers can indeed be smarter than humans.

This article explores five of the most user-friendly, highly scaleable ML libraries/tools available. These are:

- Scikit-learn
- OpenCV
- TensorFlow
- Keras
- Google Colaboratory

Some of you may be familiar with a couple of these libraries and even know how to use them. This is not meant to be an in-depth tutorial for any of these tools. The idea is that, hopefully, by the end of this article, readers will discover which of the five libraries featured, interests them enough to read further on the topic.

## **Scikit-learn**

The founders of Scikit-learn started off by trying to find answers to simple problems. I believe that any aspirant must follow the pioneer's footsteps. Scikit-learn is a free library written for Python which lets users do exactly that. It operates using Python's NumPy and SciPy libraries to achieve multiple aspects of machine learning with ease, such as various algorithms for classification, regression and clustering. It lets the user easily step into the world of artificial intelligence (AI) without making them feel that they've taken too big a leap. With its easy-to-use libraries, you can easily incorporate any of the above ML algorithms into your code. While Scikit-learn is essentially for older ML algorithms, beginners will have their hands full trying to implement various algorithms to see how they affect a particular use case.

This library is largely implemented in Python with some parts of it written in Cython, thus restricting its usage to how much Python you know. Python, incidentally, has a tremendous collection of libraries and third party APIs. The advantage of starting off with Scikit-learn is that most programmers with similar aspirations have also started off along the same path. There is huge support for Scikit-learn with the entire code written using the Scikit-learn library, making the learning process fairly simple. It also helps that the official documentation of Scikit-learn is one of the more refined and well written.

So, most of your doubts will probably get clarified by just going through the documentation.

To start off with Scikit Learn, just follow the instructions available at the official site, at <https://scikit-learn.org/stable/install.html>.

## OpenCV

OpenCV is a cross platform library that lets the user tackle any kind of real-time computer vision.

 **Note:** Computer vision is a field of computer science that deals with how computers can be made to acquire a high level knowledge from photos or videos. It includes the tasks of acquiring, processing, analysing and understanding the data.

OpenCV has a wide range of applications, starting with simple algorithms that deal with 2D images, all the way up to augmented reality, motion tracking, etc. OpenCV is not as user-friendly as Scikit-learn, yet, with a little time anyone can easily understand the flow of things – well enough to use OpenCV to its maximum extent. OpenCV, unlike Scikit-learn, is mostly written in C/C++ and is a cross platform library. This means that the users can truly optimise their code by using lower level and complied languages such as C/C++ or Java. OpenCV also includes a statistical ML library with which it can stand alone to some extent. OpenCV also has integrations with leading DL frameworks such as TensorFlow and Torch/PyTorch, making it a very versatile tool which can be used even if the provided ML library does not meet the user's demands.

One of the notable things about OpenCV is that it supports hardware acceleration in three different cases, namely:

- Intel's Integrated Performance Primitives
- A CUDA based GPU interface
- An OpenCL based GPU interface

The fact that OpenCV can use hardware acceleration means that the user can easily increase the performance of the code with simple modifications, thereby resulting in smoother real-time applications due to faster code execution.

 **Note:** It is advisable to use hardware accelerations that are mathematically intensive to perform, else there is a chance of a drop in performance due to the time it takes to shift resources to the GPU from the CPU.

Detailed installation instructions for OpenCV are available from the official documentation for the Python programming language at [https://docs.opencv.org/3.0-beta/doc/py\\_tutorials/py\\_setup/py\\_setup\\_in\\_windows/py\\_setup\\_in\\_windows.html](https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_setup/py_setup_in_windows/py_setup_in_windows.html). Other installations can also be found in the documentation.

## TensorFlow

TensorFlow is an open source library which tackles data flow programming across a range of applications, from a symbolical mathematical library to the core library in neural network programming. Developed by the Google Brain team for the search giant's internal use, today it is used within the firm for purposes ranging from research to production.

 **Note:** The TensorFlow library is so named because it operates on multi-dimensional data arrays called tensors, whilst performing calculations for neural networks.

Unlike the earlier libraries, TensorFlow has been streamlined to make neural networks and is optimised for DL applications. TensorFlow is best at data crunching at a scale where data can be realistically processed using any other older technique. You can do all this while keeping the usage so simple, that, any devoted Python user will feel right at home with the line-by-line implementation of the neural network, allowing the user to easily implement high level concepts that involve a huge amount of mathematical theory behind them, with the call of a single function. This allows the user to quickly create a neural model and focus on refining it rather than worry about the mathematics behind it, making for faster prototyping.

Like OpenCV, TensorFlow is not restricted to Python but supports other languages. Support ranges from having direct support to having third party APIs taking care of that. TensorFlow has a very flexible architecture system that allows it to be used in various devices, whether it is CPUs, GPUs or even TPUs (tensor processing units), as well as clustered servers or even handheld devices such as mobile phones. This allows TensorFlow to have a hardware acceleration capability that is unrivalled by most other libraries.

 **Note:** Keras is a pure Python library unlike TensorFlow

TensorFlow can take code to the browser with the use of TensorFlow.js, making DL projects lightweight and easily scaleable. TensorFlow Lite, which was built for using TensorFlow, specifically for Android development (beginning from Android Oreo) is even lighter. It has a wide reach. There are several applications of this library for different use cases, from being the foundation of some new technologies such as Deep Dream, an automated image captioning software, to being the core in many users' neural network projects. While it can be applied for a great many things, getting started off is not that hard. TensorFlow has huge support online with no lack of example code to go through before you tackle your first project.

You can begin your TensorFlow journey by getting the Python library at <https://www.Tensorflow.org/install/> and if you have a GPU that meets CUDA requirements, you can even optimise your execution using hardware acceleration.

## Keras

Keras is an open source Python library which runs on top of other Python libraries such as TensorFlow, Microsoft Cognitive Toolkit, or Theano. Keras focuses on being more user-friendly, modular and extensible compared to the libraries that it runs on top of.

Keras has official support from Google's TensorFlow team (for the TensorFlow

core library), allowing users to quickly build their code using the numerous implementations of commonly used neural networks methods already predefined in the Keras package. Like TensorFlow, Keras also supports hardware acceleration by the use of GPUs or TPUs, allowing the user far easier coding in almost the same execution time.

Unlike the other tools mentioned in our list, Keras is not a standalone library. Rather, it acts as a high-level API to other libraries that require comparatively more complex ways of coding. This allows users to achieve far more rapid prototyping with their code. As a trade-off for this level of user-friendly Python coding, you lose some level of control over your code. For example, in Keras, while you do have considerable control over your network, any of the lower level packages that implement Keras will control the finer things of the network.

This might lead you to wonder, why one should even use Keras? The sole reason is that most of the time, while trying out new implementations of neural networks, that high level of control over the hyper parameters isn't required. The trade-off obtained by dropping that functionality makes far more sense to any programmer who only wants to see how the network performs. You can think of Keras as a way of getting the feel of what you are trying to build before trying to optimise it at a much lower level so that your networks perform the best. Keras has one of the most user-friendly approaches to coding neural networks. For anyone who just wants to try out the latest advances in current technology, Keras will almost always meet such a user's demands. One small shortcoming is that Keras is a pure Python library unlike TensorFlow.

 **Note:** As of June 2019, Keras has officially been ported together with Tensorflow to become Tensorflow 2.0. This new version of Tensorflow cuts back on a lot of extras previously available, resulting in a more standardised approach of writing code using the Keras API. Tensorflow 2.0 is currently in beta stage, but it already implements a lot of the more commonly used features, so if you feel like foraging into the new and improved library, head over to: <https://www.tensorflow.org/beta>

It is possible for one to transform Keras models from Python to lower level languages such as C/C++.

As an example of how easy it is to understand and use Keras as compared to TensorFlow, let us take a look at both these libraries when used for the same application – the classification of handwritten numbers using the *Mnist* data set.

First let us look at an example of using pure TensorFlow to finish the task, as shown below:

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

n_nodes = 512
n_classes = 10
batch_size = 100

x = tf.placeholder('float', [None, 784])
y = tf.placeholder('float')

def neural_network_model(data):
    layer_1 = {'weights':tf.Variable(tf.random_normal([784, n_nodes])),
               'biases':tf.Variable(tf.random_normal([n_nodes]))}
    output_layer = {'weights':tf.Variable(tf.random_normal([n_nodes, n_classes])),
                   'biases':tf.Variable(tf.random_normal([n_classes]))}
    l1 = tf.add(tf.matmul(data,layer_1['weights']), layer_1['biases'])
    l1 = tf.nn.relu(l1)
    output = tf.matmul(l1,output_layer['weights']) + output_layer['biases']
    return output

def train_neural_network(x):
    prediction = neural_network_model(x)
```

```
cost = tf.reduce_mean( tf.nn.sparse_softmax_cross_entropy_with_logits(logit
s=prediction,labels=y) )

optimizer = tf.train.AdamOptimizer().minimize(cost)

hm_epochs = 5

with tf.Session() as sess:
    sess.run(tf.initialize_all_variables())
    for epoch in range(hm_epochs):
        epoch_loss = 0
        for _ in range(int(mnist.train.num_examples/batch_size)):
            epoch_x, epoch_y = mnist.train.next_batch(batch_size)
            _, c = sess.run([optimizer, cost], feed_dict={x: epoch_x, y:
epoch_y})
            epoch_loss += c
        print('Epoch', epoch, 'completed out of', hm_epochs, 'loss:', epoch_
loss)

    correct = tf.equal(tf.argmax(prediction, 1), tf.argmax(y, 1))
    accuracy = tf.reduce_mean(tf.cast(correct, 'float'))
    print('Accuracy:',accuracy.eval({x:mnist.test.images, y:mnist.test.
labels})))

train_neural_network(x)
```

Now that we have that part done, let us take a look at the same implementation using *tf.keras* :

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
```

```
tf.keras.layers.Flatten(),
tf.keras.layers.Dense(512, activation=tf.nn.relu),
tf.keras.layers.Dense(10, activation=tf.nn.softmax)

])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

print("running...")
model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

The Keras code is far simpler and the language is at a higher level than TensorFlow.

## Colaboratory by Google

Increased performance in the execution of code is possible through the use of hardware like GPUs or TPUs rather than by running it solely on the CPUs. This is because CPUs, unlike the other two, have a limited number of cores and cannot process the vast mathematical calculations required even for quite a simple neural network. However, what can be done if your GPU is not supported by CUDA or you simply do not have one?

That is where services such as Colaboratory come into play. Colaboratory is an online service that lets you use an online notebook system called Jupyter to write and execute code which is wholly based on the cloud. This avoids investing in expensive hardware such as a GPU to optimise code. Google Colaboratory gives you access to a Nvidia Tesla K80 GPU at no cost at all. The Colaboratory environment is exactly the same as any offline Python implementation. This means that there is absolutely no learning curve prior to starting off with Colaboratory. The fact that you run your code completely online means that you do not have to install any of the dependencies on your personal computer and this means that you are not restrained from implementing the latest ideas. Putting your projects completely on the cloud allows you far more freedom

and the access to Google TPUs which power the Colaboratory project. Unlike the Nvidia Tesla K80 GPU, the TPU does not come free, but you can use it for a basic fee calculated on TPU usage per second. This allows for the least possible execution time at the cheapest possible price. Like the TPU, Colaboratory also has several plans for other more powerful GPU prices, which are lower than the TPU rates, allowing you to choose what you feel is best for your particular use case.

As mentioned earlier, Colaboratory is the same as any offline Python development environment that you may already be used to, thus allowing you to use packages that you normally do, such as OpenCV, TensorFlow, Keras, etc. A simple implementation of TensorFlow using Google Colaboratory is given below, starting from the official welcome page, <https://colab.research.google.com/notebooks/welcome.ipynb>.

To start off, first create your Python3 notebook in Colaboratory as shown in Figure 1.

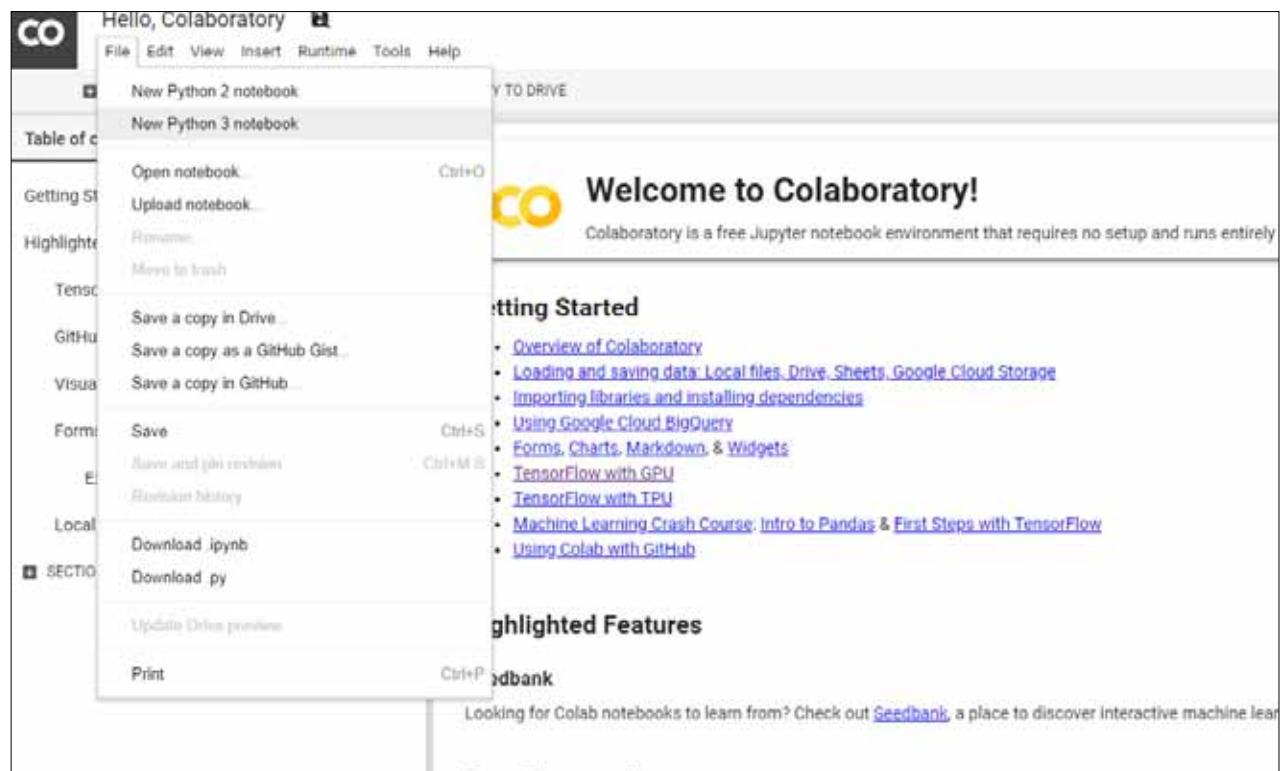


Figure 1: Creating a notebook

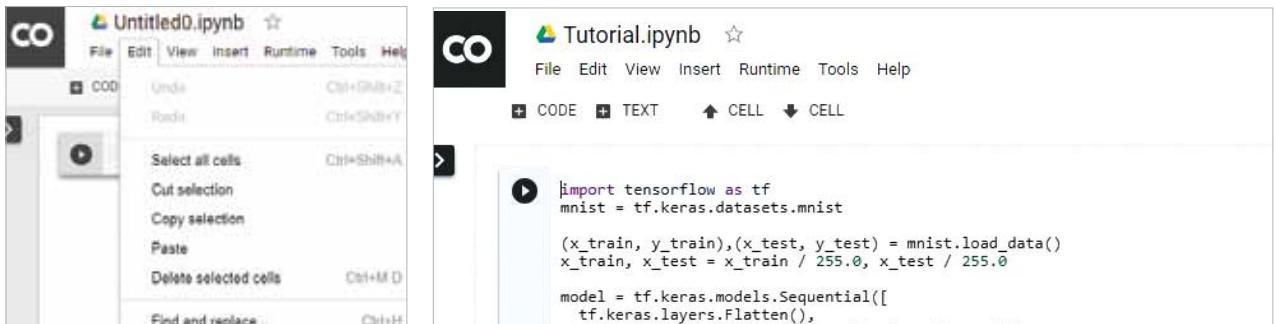


Figure 2: Accessing notebook settings

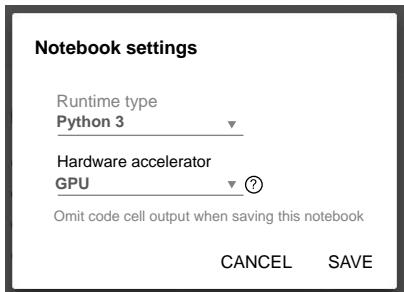


Figure 3: Hardware accelerator

```

UntitledD.ipynb ☆
File Edit View Insert Runtime Tools Help
CO Undo Ctrl+Shift+Z
Redo Ctrl+Shift+Y
Select all cells Ctrl+Shift+A
Cut selection
Copy selection
Paste
Delete selected cells Ctrl+M+D
Find and replace... Ctrl+H

Tutorial.ipynb ☆
File Edit View Insert Runtime Tools Help
CODE TEXT CELL CELL

▶ import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)

Download data from https://storage.googleapis.com/tensorflow/tf-keras-
11493376/11490434 [=====] - 0s 0us/step
Epoch 1/5
60000/60000 [=====] - 18s 295us/step - loss: 0.2
Epoch 2/5
60000/60000 [=====] - 17s 280us/step - loss: 0.0
Epoch 3/5

```

Figure 4: Mnist example

Once you have done that, navigate to the notebook settings (*Edit > Notebook Settings*) as shown in Figure 2.

Once you have selected the notebook settings, you can now change the runtime type to Python2 or Python3 and you can also change the hardware accelerator. It is here that you will decide on whether you want use the free Nvidia Tesla K80 GPU or the TPU provided by the service (Figure 3).

Now that the environment is set up, we can finally try out some program to test it. The purpose of this article is not to take you through any specific machine learning or deep learning example. For the sake of testing Colaboratory, let's use the classic Mnist example that has been taken directly from the TensorFlow website <https://www.tensorflow.org/tutorials/>

Once you have grabbed the code, simply execute it in the notebook and you should see an output as shown in Figure 4.

Using Colaboratory, you have executed a basic classification example within minutes, without the need for any extra installations of TensorFlow (or even

Python) on your computer. Thus, you can fully concentrate only on what you want to achieve and not bother about the dependencies needed for your project.

We have seen only five of the vast number libraries that are available in the open source world. These five tools will provide the reader with a strong-enough footing to push forward and explore more complex tools that are streamlined for a particular use case. I urge you to also try to learn the mathematics of any new technology that you implement as it gives you a finer understanding of the how and whys of each thing. A nice start to any theory is the Machine Learning Cheatsheet, <https://ml-cheatsheet.readthedocs.io/en/latest/index.html> which, while barely scratching the surface of the mathematics behind the implementations, gives a very clear idea about the concepts and code examples to lead you in the right direction.

# Why Python is Ideal for Machine Learning

*Many programming languages facilitate machine learning, a few prominent ones being Java, Python, R and C++. Python leads the field in its ease of use and the simplicity of programming code. Here are some very compelling reasons for why one should use Python for machine learning.*

Parents teach us behavioural skills right from childhood, teachers help us increase our knowledge of a range of subjects, and as adults we learn a few life skills based on our experiences. So learning is a part of our life, right from birth to death. Wouldn't it be amazing if our machines also start learning the same way we do? This question made Frank Rosenblatt jump into the arena of machine learning and neural networks in the 1950s, and he went on to develop the first neural network for computers, which is best known as the Perceptron.

We always wonder how machines can be made to learn by themselves—since they are non-living objects and do not have brains like us. So Rosenblatt tried giving brains to computers. He used the Perceptron to simulate different thought processes of the human brain, in computers. Our brains have a network of neurons and a central nervous system which helps us read situations, actions and conditions, and make decisions based on these. Similarly, the networks developed by Rosenblatt act like the network of neurons for machines or computers. The various algorithms used by neural networks act as the central nervous system, which helps machines make decisions on the basis of information or data collected.

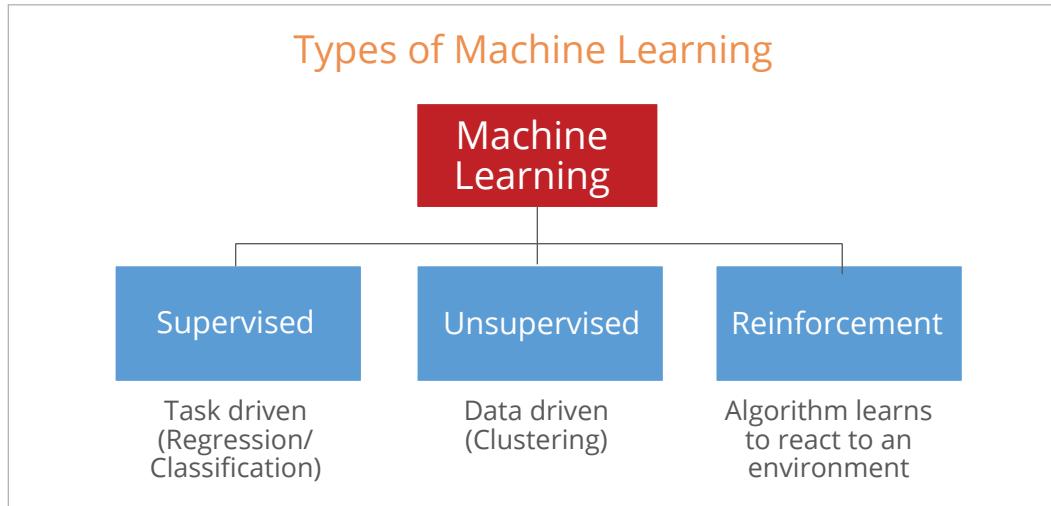


Figure 1: Different types of machine learning (Image source: [googleimages.com](http://googleimages.com))

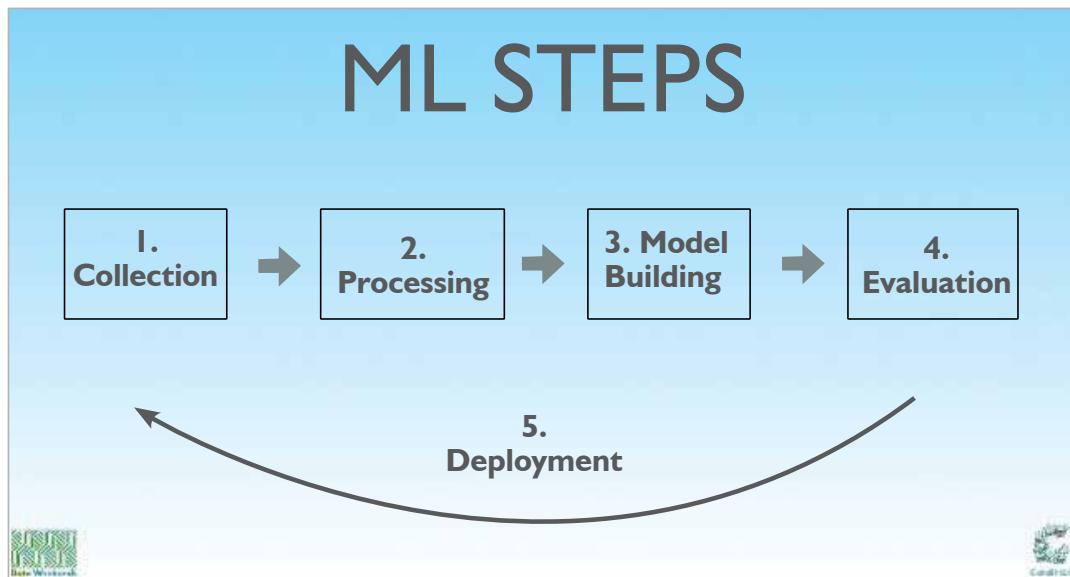


Figure 2: Different steps involved in developing a machine learning project (Image source: [googleimages.com](http://googleimages.com))

In 1957, one of the first algorithms named Nearest Neighbour was written and it allowed computers to use basic pattern recognition techniques. This algorithm could be used to map the route for travelling salesmen, starting at any city and ensuring they visited all the different cities they needed to, during their tour.

Today, machines are capable of learning by themselves, and this is called machine learning.

Machine learning (ML) is defined as a sub-set of artificial intelligence where different computer algorithms are used to autonomously learn from available data sets and information. In ML, machines or computers don't need to be explicitly programmed — they can change and continuously improve their algorithms by themselves. Nowadays, ML algorithms help computers to communicate with human beings on their own, write and publish sports match reports, autonomously drive cars, find terrorist suspects and do a whole lot more. ML gave birth to Sophia, a humanoid robot, which received citizenship from Saudi Arabia in 2017.

ML has basically evolved from the study of computational learning theory and pattern recognition in artificial intelligence. It explores the study and construction of different algorithms that can learn from and make predictions on the basis of the available data sets. All these algorithms follow the strictly static program instructions by applying data-driven decisions or predictions through the development of a model from given sample inputs. ML is implemented in a wide range of computing tasks where designing and developing explicit algorithms, that too, with good performance, is quite difficult and often not even feasible. For instance, e-mail filtering, the detection of network intruders or any malicious insiders working towards data breaches, learning to rank, optical character recognition, etc, are some well known applications of ML.

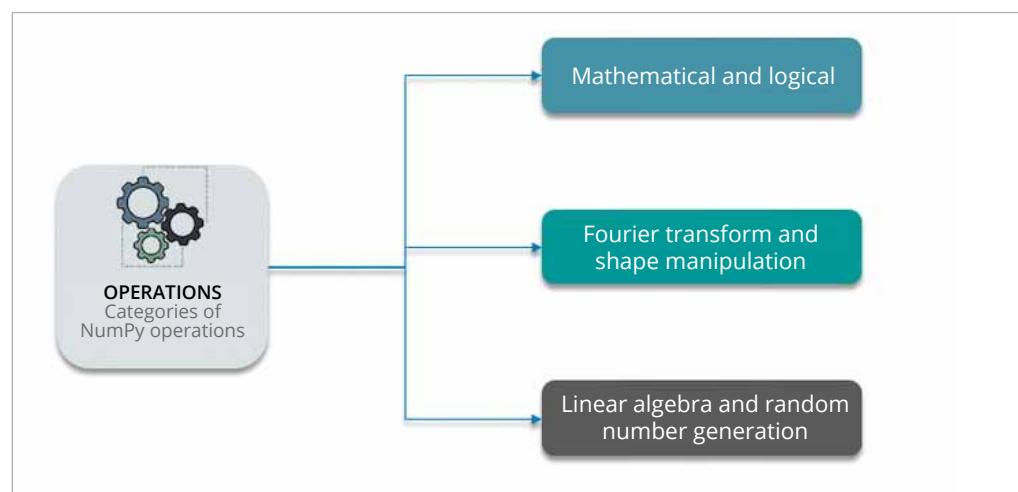


Figure 3: Operations performed by Python NumPy library (Image source: [googleimages.com](http://googleimages.com))

## Types of machine learning

Machine learning is broadly classified into three categories on the basis of the way machines learn to increase their knowledge and take decisions.

**Supervised machine learning:** In this case, we train and teach the machine or computer, using available data sets that are well labelled. It means that some of the data present in the data sets are already tagged with correct answers. Thereafter, the same computer or machine is provided with a new set of data that applies the supervised learning algorithm, analyses the data and then produces a correct outcome from the labelled data.

For example, if we are given a basket filled with different varieties of fruits, when implementing supervised learning, the first step will be to train the machine with respect to the different fruits, one by one. This involves teaching the machine to recognise the different categories so that this knowledge can be applied to the rest of the fruits.

Supervised learning is categorised into two types of algorithms.

**Classification:** A classification algorithm is used when the output variable expected is in terms of the name of a category, like red, blue, etc.

**Regression:** A regression algorithm is applied when the output variable expected is in terms of any real value, such as dollars, weight, etc.

**Unsupervised machine learning:** This is the training of machines using the available set of information that is neither categorised nor labelled. It directly allows the algorithm to act on the information set without any guidance. In unsupervised ML, it will be the task of the machine to group unsorted data according to patterns, similarities and differences, without any prior training. Unlike in the case of supervised learning, no such training is provided to the machine. Instead, the machine needs to find the hidden structure in the given set of unlabelled data on its own.

For example, let's consider an image that has both dogs and cats, but no information or label is provided with respect to these animals. In this case, unsupervised learning needs to be applied to help machines categorise them.

Unsupervised learning is categorised into the following two types of algorithms.

**Clustering:** A clustering algorithm is one that is applied wherever we want to discover the inherent groupings in the available data sets, like grouping customers on the basis of their purchasing behaviour.

**Association:** An association rule algorithm is applied wherever we need to discover a set of rules that describes a large portion of the data, like identifying that the people who buy X are likely to buy Y.

**Reinforcement machine learning:** This third category focuses on behavioural psychology. It looks at how software agents should take specific action in any environment to maximise some notion, in order to achieve the cumulative reward. The rewards can be winning a game or earning more money.

## **Python — an open source programming language for machine learning**

Python is simply the Swiss army knife of ML. It is one of the open source programming languages widely used to perform complex operations and has a full suite of tools for increasing the productivity of ML.

Python is popular among its users because it is easy to learn and pretty simple when it comes to programming. It is considered to be one of the most consistent math-like programming languages. Python has its own set of built-in functions and utilities, which help it perform a plethora of complex operations with just a few lines of code. It is also way ahead in terms of its easy syntactical character when compared to other programming languages.

Python has a large set of libraries that can be easily used for machine learning, such as SciPy, NumPy, ScikitLearn, PyBrain, etc. It can be used to develop code in the Map-Reduce model even while working in the Hadoop ecosystem. Spark, one of the modern technologies used for scalable Big Data analysis, has also got its machine learning libraries written in Python. Simplicity and wide applicability make Python a popular ML language.

At times, the code developed in Python almost seems to be written in the English language. The way Python mirrors human language or its mathematical counterparts makes ML a bit easier. As per the TIOBE programming index, Python is ranked No. 5, which is way above other programming languages that are used for ML and data analysis, like R. Python has outperformed R in the fields of data science and ML for the last five years.

Python's capabilities are given a big boost by the different available packages in ML and data analytics. Pandas, one of the best known data analysis packages, gives Python high-performance structures and several data analysis tools as well. Python excels in offering a playground for playing with data, not just numerically, but also for the following other functions:

- Downloading varied contents from websites and APIs
- Interfacing with different databases and spreadsheets
- Manipulating audio, text and images
- Availability of sophisticated tools for the exploration and presentation of results, like Pandas, Jupyter, etc.

## **Why is Python suitable for machine learning?**

Here are some important reasons why Python is considered ideal for ML.

1. Python is an easy to learn and developer-friendly programming language that is pretty simple to start with. Its syntax is simple and the code can be written in fewer lines as compared to other programming languages.

2. It has numerous built-in packages for ML and other computations — for example, Numpy, Keras, Pandas, etc. All these packages are well-documented, and hence are quite helpful in starting with any project or solution. It accelerates the process of fixing bugs.
3. All the available libraries of Python are quite powerful. They comprise many features that are helpful in performing complex computations. These help in fast, efficient and stable development. Python also uses a wide range of computation speed improvements continuously to improve the performance of libraries.
4. Python has got considerable support from the community, so developers can easily find a large number of tutorials and valuable tips during the development process. This makes it easier to use any new technology from scratch.
5. Python is like the new FORTRAN of the scientific world. It's very popular in the non-computer scientist's world, equipping users with an enormously large toolbox for different kinds of applied programming problems.
6. Python makes it very easy to quickly implement or experiment with any new ideas and prototypes. Hence, different scientific and research communities love to use it. That is why it is being widely used in ML and data science.

## Starting a Python machine learning project

The following steps act as a good guide for how to begin work on a machine learning project.

1. **Define the problem:** The problem or the scenario for which we want to develop an ML solution must be defined and understood. At times, there is a deviation from the pre-defined problem set while the ML solution is being developed. Hence, this is a significant step to start with.

2. **Prepare the data:** The data sets for a ML solution that have to be processed and analysed to give a concluding result must be prepared. These available data sets are the inputs used by algorithms and several analysis operations are performed on them before coming to a final result. These data sets have a great impact on the final result.
3. **Evaluate algorithms:** Different algorithms must be analysed before choosing and deciding to go ahead with one. The algorithm chosen decides the performance of the ML project. The problem set, the behaviour and the volume of the data set, as well as what is expected of the ML project must be considered before choosing an algorithm. There is every possibility that more than two algorithms may perform operations on the data set and give the desired result but the performance factor associated with each algorithm must be evaluated. This plays quite a significant role, especially when it comes to a large ML project.
4. **Improve the results:** As any ML project has the basic quality of continuously improving itself in terms of the technique used by it or the algorithm implemented, it always obtains better results, every time. Also, machines learn from the diverse set of data which they analyse before giving the result. Hence, even the quality of data sets helps an ML solution produce improved results.
5. **Present the results:** Once the machine fetches the desired result, it is very important that it is presented in an effective manner so that it can be used further for different applications. The presentation of the desired result is one of the very basic qualities of any ML project. The results can be presented with the help of graphical plots, numerically in the form of some table or even just as the concluding result. But, ultimately, the project should present the result in the desired manner.

Now let us consider some other important factors to ensure an effective and efficient ML project.

**Model building, training and evaluation:** Developing a machine learning model is a very responsible task as it is almost similar to building a product. The starting point is always ideation, wherein the problem for which the solution is required is considered in its various aspects with some potential approaches. Once a clear direction is established, the solution is prototyped and further tested to check if it meets the needs. A continuous loop between the process of ideation, prototyping and validation takes place until the ML solution is mature enough to be brought into the market, when it is productised for a broader launch.

**Data pipeline:** A machine learning algorithm generally takes a cleaned data set and learns some patterns in it so that it can decide and make some predictions on the new data set. However, when ML is used for real-life applications, the raw data obtained from the real world is not ready to be directly fed into the ML algorithm. This raw data needs to be preprocessed to generate input data for the ML algorithm. Hence, this entire process of converting the raw data to usable data by the ML algorithm, training an ML algorithm, and finally using this output to perform different actions in the real world is called a pipeline.

**Data wrangling:** To avoid the ‘garbage-in-garbage-out’ (GIGO) condition, mapping and the transformation of raw data into ML-ready data must be done properly. Data wrangling is one of the key processes that enables an ML project to make a big impact by avoiding GIGO situations. Filtering, cleansing, joining and stacking are the necessary steps to get data ready for ML. Pyspark is one of the widely used engines for data wrangling. This can also be seamlessly integrated with other Big Data engines. Python has got Dask as its indigenous Big Data engine.

**Feature extraction and engineering:** In an ML project, feature extraction has got its own role to play. During different phases like image processing and pattern recognition in a machine learning project, feature extraction starts from a very initial set of measured data and builds derived values (also called features) that are intended to be non-redundant and informative, hence

facilitating the subsequent learning as well as generalisation of steps (and in some conditions leading to better human interpretations). Feature extraction is basically related to dimensionality reduction. Python has a couple of image or video libraries that can assist in feature extraction.

## **Different open source scientific Python libraries used for machine learning**

There are many open source libraries used to implement ML for different applications. They are widely referred to as scientific Python libraries as they are put to use while performing the elementary machine learning tasks.

1. **Numpy:** This is a Python library that is widely used for N-dimensional array objects.
2. **Pandas:** This library file is used for Python data analysis, including different structures such as data frames.
3. **Matplotlib:** This is a 2D plotting library that produces publication-quality figures.
4. **Scikit-learn:** This library contains a couple of the ML learning algorithms that are used for data mining and data analysis tasks.
5. **Seaborn:** Sometimes it is difficult to get accurate plots with the help of Matplotlib as it focuses on line plots. In such cases, one can go with a more specific library, called Seaborn. It focuses on the visual aspects of different statistical models including heat maps, and also depicts the overall distribution of the data.

# Machine Learning Libraries that can Make Web Applications Smarter

*Web applications have come a long way. From the simple data capturing applications of the past they are now complex and dynamic. To enrich their capabilities further, it would be great to harness their machine learning potential. This article explores four Web browser based machine learning libraries and their features.*

Web applications enable organisations to cater to the needs of a large number of users, distributed across various geographical locations, without requiring any local installation process. Upgrading Web applications is also comparatively simple, as the code needs to be updated only in the server. It is very tedious to update a desktop application in each installed location. Due to the advantages of scale and ease of maintenance, Web applications are now preferred to their desktop counterparts.

In the past, Web applications were comparatively simpler in nature. They mostly functioned as data collection platforms with simple interfaces. With the prolific growth in Web technologies, these apps have evolved into complex and dynamic entities.

Machine learning (ML) is evolving rapidly and is being applied to various domains. Web apps, too, can be enriched with ML capabilities and become more powerful. Machine learning can be incorporated into Web applications in two ways:

- Machine learning as a server side component
- Machine learning as a client side component

There are pros and cons for both these approaches. The server side applications have an advantage of better processing capabilities with a

bigger memory. At the same time, one of the bottlenecks in server side ML is the delay due to network traffic. Each request with ML functionality needs to be communicated to the server, where it has to be processed, and the results should be returned to the client. These steps introduce latency in the application, which is best avoided.

To overcome this problem, it would be better to run the ML components in the browser itself. Thereby, the latency due to the network round-trip can be avoided. As the client devices these days have better processing capabilities, the ML components can be executed with an acceptable performance. This article explores four browser based ML libraries (Figure 1).



Figure 1: Web browser based machine learning libraries

## TensorFlow.js

TensorFlow is a popular machine learning library from Google. It makes the ML implementation effective. The TensorFlow.js is a JavaScript library, which can be used to implement ML models in Web browsers. Both training and deploying of the models can be done through TensorFlow.js, which has the following capabilities:

- The ML models can be built and trained from scratch with powerful APIs.
- It can be used to run the existing TensorFlow models inside the browser.
- The existing models can be retrained with TensorFlow.js.

The official documentation has provided many working examples (real-time human estimation, etc) of TensorFlow.js implementations that will give you an idea about the capabilities of this powerful ML library (<https://js.tensorflow.org/>).

If you are familiar with the working of Tensor, layers, optimisation and loss functions, then using the TensorFlow.js library is very simple.

TensorFlow.js can be included in Web applications in many ways. The simplest approach is to use a script tag, as follows:

```
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@0.14.2/dist/tf.min.js"></script>
```

Another approach is to use NPM, as follows:

```
npm install @tensorflow/tfjs
```

If you want to convert an existing model, the *tensorflowjs\_converter* can be used. The following code is used to convert the *SavedModel*:

```
tensorflowjs_converter \ --input_format=tf_saved_model \ --output_node_names='MobilenetV1/Predictions/Reshape_1' \ --saved_model_tags=serve \ /mobilenet/saved_model \ /mobilenet/web_model
```

In the case of a *FrozenModel*, the following command can be used:

```
tensorflowjs_converter \ --input_format=tf_frozen_model \ --output_node_names='MobilenetV1/Predictions/Reshape_1' \ /mobilenet/frozen_model.pb \ /mobilenet/web_model
```

To run this conversion, you require a Python environment. The converter can

be installed with Pip:

```
pip install tensorflowjs
```

The conversion script generates three files:

- The dataflow graph (*web\_model.pb*)
- The weight manifest file (*weights\_manifest.json*)
- A collection of binary weight files (*group1-shard\\*of\\**)

The loading and running of the model in the browser can be done using the following code snippet:

```
import * as tf from '@tensorflow/tfjs';
import {loadFrozenModel} from '@tensorflow/tfjs-converter';
const MODEL_URL = 'https://.../mobilenet/web_model.pb';
const WEIGHTS_URL = 'https://.../mobilenet/weights_manifest.json'; const model
= await loadFrozenModel(MODEL_URL, WEIGHTS_URL);
const cat = document.getElementById('cat');
model.execute({input: tf.fromPixels(cat)});
```

(Source: <https://js.tensorflow.org/tutorials/import-saved-model.html>)

The complete source code and execution procedure can be gathered from <https://github.com/tensorflow/tfjs-converter/tree/master/demo/mobilenet>. A detailed tutorial on TensorFlow.js is available at <https://js.tensorflow.org/>.

## ml5.js

An understanding of ML concepts is necessary to use libraries such as TensorFlow.js effectively. If you are not interested in the internals of how things work but you only want ML applications, then give ml5.js a try. The official documentation claims that this library is made to enable a broader audience get involved in machine learning. ml5.js is built on top of TensorFlow.js and is hugely inspired by 'processing' and p5.js.

Setting up your environment to use ml5.js is very simple. Just use a `<script>` tag to link the external JavaScript file.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Machine Learning with ml5.js</title>
    <script src="https://unpkg.com/ml5@0.1.3/dist/ml5.min.js">
    </script>
  </head>
</html>
```

A simple image classification example is shown below:

```
<body>
<h1>Image classification using MobileNet</h1>
<p>The MobileNet model labeled this as
<span id="result">...</span> with a confidence of
<span id="probability">...</span></p>


<script>
// The image we want to classify
const image = document.getElementById('image');
// The result tag in the HTML
const result = document.getElementById('result');
// The probability tag in the HTML
const probability = document.getElementById('probability');

// Initialize the Image Classifier method with MobileNet
const classifier = ml5.imageClassifier('MobileNet', function() {
  console.log('Model Loaded!');
```

```
});

// Make a prediction with the selected image
// This will return an array with a default of 10 options with their
probabilities

classifier.predict(image, function(err, results) {
  result.innerText = results[0].className;
  probability.innerText = results[0].probability.toFixed(4);
});

</script>
</body>
</html>
```

(Source: <https://ml5js.org/docs/getting-started.html>)

Just save this file and open it in a browser. Your working demo is ready. There are no dependencies to configure. It can be inferred from the above code that ml5.js has direct functions available to perform the classification task. We just need to specify the model and the results are ready.

The official documentation has listed various code examples such as a style transfer, a Pong game in the browser, and a text editor with ML based suggestions (<https://ml5js.org/experiments>).

## Brain.js

This is a useful JavaScript ML library. It is used to build neural networks in the browser. The 'node' based configuration of Brain.js can be done as follows:

```
npm install brain.js
```

...or by using the following command:

```
yarn add brain.js
```

The types of neural networks supported by Brain.js are given below:

- Feed forward neural network: *brain.NeuralNetwork*
- FF neural networks with back-propagation: *brain.NeuralNetworkGPU*
- Time step recurrent neural network (NN) or RNN: *brain.recurrent.RNNTimeStep*
- Time step long/short term memory (LSTM) NN: *brain.recurrent.LSTMTimeStep*
- Time step gated recurrent unit (GRU): *brain.recurrent.GRUTimeStep*
- RNN: *brain.recurrent.RNN*
- LSTM neural network: *brain.recurrent.LSTM*
- GRU: *brain.recurrent.GRU*

Further details on Brain.js can be obtained from <https://github.com/BrainJS/brain.js>.

## ConvNetJS

This is an implementation of neural networks in JavaScript. ConvNetJS supports the following:

- Common NN modules (full connected layers and non-linearities)
- Classification and regression
- Can train convolutional neural networks for processing images
- A reinforcement learning module based on Deep Q learning

ConvNetJS has no external dependencies and hence is very simple to handle. The official documentation lists many working demos. A CIFAR-10 demo is provided at <https://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html>.

Importing ConvNetJS is done with the following simple script tag:

```
<!-- import convnetjs library -->
```

```
<script src="convnet-min.js"></script>
```

Multi-layer networks can be built with the following code sequence:

```
var layer_defs = [];
// input layer of size 1x1x2 (all volumes are 3D)
layer_defs.push({type:'input', out_sx:1, out_sy:1, out_depth:2});
// some fully connected layers
layer_defs.push({type:'fc', num_neurons:20, activation:'relu'});
layer_defs.push({type:'fc', num_neurons:20, activation:'relu'});
// a softmax classifier predicting probabilities for two classes: 0,1
layer_defs.push({type:'softmax', num_classes:2});
// create a net out of it
var net = new convnetjs.Net();
net.makeLayers(layer_defs);
(Source: https://cs.stanford.edu/people/karpathy/convnetjs/started.html)
```

The libraries mentioned in this article enable the developer to incorporate ML into Web applications without many dependencies. Browser based ML can harness the untapped hardware potential on the client side instead of loading the server heavily with a huge number of requests. As these libraries advance further, future Web applications will be able to provide various smart features based on ML algorithms that will benefit users greatly.

# What's Good About TensorFlow 2.0?

*Version 2.0 of TensorFlow is focused on simplicity and ease of use. It has been strengthened with updates like eager execution and intuitive higher level APIs accompanied by flexible model building. It is platform agnostic, and makes APIs more consistent, while removing those that are redundant.*

Machine learning and artificial intelligence are experiencing a revolution these days, primarily due to three major factors. The first is the increased computing power available within small form factors such as GPUs, NPUs and TPUs. The second is the breakthrough in machine learning algorithms. State-of-art algorithms and hence models are available to infer faster. Finally, huge amounts of labelled data is essential for deep learning models to perform better, and this is now available.

TensorFlow is an open source AI framework from Google which arms researchers and developers with the right tools to build novel models. It was made open source in 2015 and, in the past few years, has evolved with various enhancements covering operator support, programming languages, hardware support, data sets, official models, and distributed training and deployment strategies.

TensorFlow 2.0 was released recently at the TensorFlow Developer Summit. It has major changes across the stack, some of which will be discussed from the developers' point of view.

TensorFlow 2.0 is primarily focused on the ease-of-use, power and scalability aspects. Ease is ensured in terms of simplified APIs, Keras being the main high level API interface; eager execution is available by default. Version 2.0 is powerful in the sense of being flexible and running much faster than earlier,

with more optimisation. Finally, it is more scalable since it can be deployed on high-end distributed environments as well as on small edge devices.

This new release streamlines the various components involved, from data preparation all the way up to deployment on various targets. High speed data processing pipelines are offered by *tf.data*, high level APIs are offered by *tf.keras*, and there are simplified APIs to access various distribution strategies on targets like the CPU, GPU and TPU. TensorFlow 2.0 offers a unique packaging format called SavedModel that can be deployed over the cloud through a TensorFlow Serving. Edge devices can be deployed through TensorFlow Lite, and Web applications through the newly introduced TensorFlow.js and various other language bindings that are also available. TensorFlow.js was announced at the developer summit with off-the-shelf pretrained models for the browser, node, desktop and mobile native applications. The inclusion of Swift was also announced. Looking at some of the performance improvements since last year, the latest release claims a training speedup of 1.8x on NVIDIA Tesla V100, a 1.6x training speedup on Google Cloud TPUs and a 3.3.x inference speedup on Intel Skylake.

## Upgrade to 2.0

The new release offers a utility *tf\_upgrade\_v2* to convert a 1.x Python application script to a 2.0 compatible script. It does most of the job in converting the 1.x deprecated API to a newer compatibility API. An example of the same can be seen below:

```
test-pc:~$cat test-infer-v1.py

# Tensorflow imports
import tensorflow as tf

save_path = 'checkpoints/dev'
with tf.gfile.FastGFile("./trained-graph.pb", 'rb') as f:
    graph_def = tf.GraphDef()
```

```
graph_def.ParseFromString(f.read())
tf.import_graph_def(graph_def, name='')

with tf.Session(graph=tf.get_default_graph()) as sess:
    input_data = sess.graph.get_tensor_by_name("DecodeJPGInput:0")
    output_data = sess.graph.get_tensor_by_name("final_result:0")

    image = 'elephant-299.jpg'
    if not tf.gfile.Exists(image):
        tf.logging.fatal('File does not exist %s', image)
    image_data = tf.gfile.FastGFile(image, 'rb').read()

    result = sess.run(output_data, {'DecodeJPGInput:0': image_data})
    print(result)
```

test-pc:~\$ tf\_upgrade\_v2 --infile test-infer-v1.py --outfile test-infer-v2.py

```
INFO line 5:5: Renamed 'tf.gfile.FastGFile' to 'tf.compat.v1.gfile.FastGFile'
INFO line 6:16: Renamed 'tf.GraphDef' to 'tf.compat.v1.GraphDef'
INFO line 10:9: Renamed 'tf.Session' to 'tf.compat.v1.Session'
INFO line 10:26: Renamed 'tf.get_default_graph' to 'tf.compat.v1.get_default_
graph'
INFO line 15:15: Renamed 'tf.gfile.Exists' to 'tf.io.gfile.exists'
INFO line 16:12: Renamed 'tf.logging.fatal' to 'tf.compat.v1.logging.fatal'
INFO line 17:21: Renamed 'tf.gfile.FastGFile' to 'tf.compat.v1.gfile.FastGFile'
TensorFlow 2.0 Upgrade Script
```

---

Converted 1 files

Detected 0 issues that require attention

---

Make sure to read the detailed log 'report.txt'

test-pc:~\$ cat test-infer-v2.py

```
# Tensorflow imports
import tensorflow as tf

save_path = 'checkpoints/dev'
with tf.compat.v1.gfile.FastGFile("./trained-graph.pb", 'rb') as f:
    graph_def = tf.compat.v1.GraphDef()
    graph_def.ParseFromString(f.read())
    tf.import_graph_def(graph_def, name='')

with tf.compat.v1.Session(graph=tf.compat.v1.get_default_graph()) as sess:
    input_data = sess.graph.get_tensor_by_name("DecodeJPGInput:0")
    output_data = sess.graph.get_tensor_by_name("final_result:0")

    image = 'elephant-299.jpg'
    if not tf.io.gfile.exists(image):
        tf.compat.v1.logging.fatal('File does not exist %s', image)
    image_data = tf.compat.v1.gfile.FastGFile(image, 'rb').read()

    result = sess.run(output_data, {'DecodeJPGInput:0': image_data})
    print(result)
```

As we can see here, the *tf\_upgrade\_v2* utility converts all the deprecated APIs to compatible v1 APIs, to make them work with 2.0.

**Eager execution:** Eager execution allows real-time evaluation of Tensors without calling *session.run*. A major advantage with eager execution is that we can print the Tensor values any time for debugging.

With TensorFlow 1.x, the code is:

```
test-pc:~$python3
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
```

Type “help”, “copyright”, “credits” or “license” for more information.

```
>>> import tensorflow as tf  
>>> print(tf.__version__)  
1.14.0  
>>> tf.add(2,3)  
<tf.Tensor 'Add:0' shape=() dtype=int32>
```

TensorFlow 2.0, on the other hand, evaluates the result that we call the API:

```
test-pc:~$python3  
Python 3.6.7 (default, Oct 22 2018, 11:32:17)  
[GCC 8.2.0] on linux  
Type “help”, “copyright”, “credits” or “license” for more information.  
>>> import tensorflow as tf  
>>> print(tf.__version__)  
2.0.0-beta1  
>>> tf.add(2,3)  
<tf.Tensor: id=2, shape=(), dtype=int32, numpy=5>
```

In v1.x, the resulting Tensor doesn't display the value and we need to execute the graph under a session to get the value, but in v2.0 the values are implicitly computed and available for debugging.

## Keras

Keras (*tf.keras*) is now the official high level API. It has been enhanced with many compatible low level APIs. The redundancy across Keras and TensorFlow is removed, and most of the APIs are now available with Keras. The low level operators are still accessible through *tf.raw\_ops*.

We can now save the Keras model directly as a Tensorflow SavedModel, as shown below:

```
# Save Model to SavedModel
```

```
saved_model_path = tf.keras.experimental.export_saved_model(model, '/path/to/  
model')  
  
# Load the SavedModel  
new_model = tf.keras.experimental.load_from_saved_model(saved_model_path)  
  
# new_model is now keras Model object.  
new_model.summary()
```

Earlier, APIs related to various layers, optimisers, metrics and loss functions were distributed across Keras and native TensorFlow. Latest enhancements unify them as *tf.keras.optimizer.\**, *tf.keras.metrics.\**, *tf.keras.losses.\** and *tf.keras.layers.\**.

The RNN layers are now much more simplified compared to v 1.x.  
With TensorFlow 1.x, the commands given are:

```
if tf.test.is_gpu_available():  
    model.add(tf.keras.layers.CudnnLSTM(32))  
else  
    model.add(tf.keras.layers.LSTM(32))
```

With TensorFlow 2.0, the commands given are:

```
# This will use Cudnn kernel when the GPU is available.  
model.add(tf.keras.layer.LSTM(32))
```

TensorBoard integration is now a simple call back, as shown below:

```
tb_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir)  
  
model.fit(  
    x_train, y_train, epochs=5,
```

```
validation_data = [x_test, y_test],
Callbacks = [tb_callbacks])
```

With this simple call back addition, TensorBoard is up on the browser to look for all the statistics in real-time.

Keras offers unified distribution strategies, and a few lines of code can enable the required strategy as shown below:

```
strategy = tf.distribute.MirroredStrategy()

with strategy.scope():
    model = tf.keras.models.Sequential([
        tf.keras.layers.Dense(64, input_shape=[10]),
        tf.keras.layers.Dense(64, activation='relu'),
        tf.keras.layers.Dense(10, activation='softmax')])

    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
```

As shown above, the model definition under the desired scope is all we need to apply the desired strategy. Very soon, there will be support for multi-node synchronous and TPU strategy, and later, for parameter server strategy.

## TensorFlow function

Function is a major upgrade that impacts the way we write TensorFlow applications. The new version introduces *tf.function*, which simplifies the applications and makes it very close to writing a normal Python application. A sample *tf.function* definition looks like what's shown in the code snippet below. Here the *tf.function* declaration makes the user define a function as a TensorFlow operator, and all optimisation is applied automatically. Also, the function is faster than eager execution. APIs like *tf.control\_dependencies*,

*tf.global\_variable\_initializer*, and *tf.cond*, *tf.while\_loop* are no longer needed with *tf.function*. The user defined functions are polymorphic by default, i.e., we may pass mixed type tensors.

```
test-pc:~$ cat tf-test.py
import tensorflow as tf

print(tf.__version__)

@tf.function
def add(a, b):
    return (a+b)

print(add(tf.ones([2,2]), tf.ones([2,2])))
```

```
test-pc:~$ python3 tf-test.py
2.0.0-beta1
tf.Tensor(
[[2. 2.]
 [2. 2.]], shape=(2, 2), dtype=float32)
```

Here is another example to demonstrate automatic control flows and Autograph in action. Autograph automatically converts the conditions, while it loops Python to TensorFlow operators.

```
test-pc:~$ cat tf-test-control.py
import tensorflow as tf

print(tf.__version__)

@tf.function
def f(x):
    while tf.reduce_sum(x) > 1:
```

```

x = tf.tanh(x)
return x

print(f(tf.random.uniform([10])))

test-pc:~$ python3 tf-test-control.py

2.0.0-beta1
tf.Tensor(
[0.10785562 0.11102211 0.11347286 0.11239681 0.03989326 0.10335539
 0.11030331 0.1135259 0.11357211 0.07324989], shape=(10,), dtype=float32)

```

We can see Autograph in action with the following API over the function.

```
print(tf.autograph.to_code(f)) # f is the function name
```

## TensorFlow Lite

The latest advancements in edge devices add neural network accelerators. Google has released EdgeTPU, Intel has the edge inference platform Movidius, Huawei mobile devices have the Kirin based NPU, Qualcomm has come up with NPE SDK to accelerate on the Snapdragon chipsets using Hexagon power and, recently, Samsung released Exynos 9 with NPU. An edge device optimised framework is necessary to support these hardware ecosystems.

Unlike TensorFlow, which is widely used in high power-consuming server infrastructure, edge devices are challenging in terms of reduced computing power, limited memory and battery constraints. TensorFlow Lite is aimed at bringing in TensorFlow models directly onto the edge with minimal effort. The TF Lite model format is different from TensorFlow. A TF Lite converter is available to convert a TensorFlow SavedBundle to a TF Lite model.

Though TensorFlow Lite is evolving, there are limitations too, such as in the number of operations supported, and the unsupported semantics like

control-flows and RNNs. In its early days, TF Lite used a TOCO converter and there were a few challenges for the developer community. A brand new 2.0 converter is planned to be released soon. There are claims that using TF Lite results in huge improvements across the CPU, GPU and TPU.

TF Lite introduces delegates to accelerate parts of the graph on an accelerator. We may choose a specific delegate for a specific sub-graph, if needed.

```
#import "tensorflow/lite/delegates/gpu/metal_delegate.h"

// Initialize interpreter with GPU delegate
std::unique_ptr<Interpreter> interpreter;
InterpreterBuilder(*model, resolver)(&interpreter);
auto* delegate = NewGpuDelegate(nullptr); // default config
if (interpreter->ModifyGraphWithDelegate(delegate) != kTfLiteOk) return false;

// Run inference
while (true) {
    WriteToInputTensor(interpreter->typed_input_tensor<float>(0));
    if (interpreter->Invoke() != kTfLiteOk) return false;
    ReadFromOutputTensor(interpreter->typed_output_tensor<float>(0));
}

// Clean up
interpreter = nullptr;
DeleteGpuDelegate(delegate);
```

As shown above, we can choose GPUDelegate, and modify the graph with the respective kernel's runtime. TF Lite is going to support the Android NNAPI delegate, in order to support all the hardware that is supported by NNAPI.

For edge devices, CPU optimisation is also important, as not all edge devices are equipped with accelerators; hence, there is a plan to support further

optimisations for ARM and x86.

Optimisations based on quantisation and pruning are evolving to reduce the size and processing demands of models. Quantisation generally can reduce model size by 4x (i.e., 32-bit to 8-bit). Models with more convolution layers may get faster by 10 to 50 per cent on the CPU. Fully connected and RNN layers may speed up operation by 3x.

TF Lite now supports post-training quantisation, which reduces the size along with compute demands greatly. TensorFlow 2.0 offers simplified APIs to build models with quantisation and by pruning optimisations.

A normal dense layer without quantisation looks like what follows:

```
tf.keras.layers.Dense(512, activation='relu')
```

Whereas a quality dense layer looks like what's shown below:

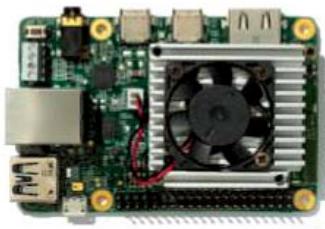
```
quantize.Quantize(tf.keras.layers.Dense(512, activation='relu'))
```

Pruning is a technique used to drop connections that are ineffective. In general, 'dense' layers contain lots of connections which don't influence the output. Such connections can be dropped by making the weight zero. Tensors with lots of zeros may be represented as 'sparse' and can be compressed. Also, the number of operations in a sparse tensor is less.

Building a layer with *prune* is as simple as using the following command:

```
prune.Pruned(tf.keras.layers.Dense(512, activation='relu'))
```

In a pipeline, there is Keras based quantised training and Keras based connection pruning. These optimisations may push TF Lite further ahead of the competition, with regard to other frameworks.



#### Dev Board

A single-board computer with a removable system-on-module (SoM) featuring the Edge TPU.

- **Supported OS:** Mendel Linux (derivative of Debian)
- **Supported Framework:** TensorFlow Lite, AutoML Edge
- **Languages:** Python and C++
- **CPU:** NXP i.MX 8M SoC (quad Cortex-A53, Cortex-M4F)
- **GPU:** Integrated GC7000 Lite Graphics
- **ML accelerator:** Google Edge TPU coprocessor
- **RAM:** 1 GB LPDDR4
- **Flash memory:** 8 GB eMMC
- **Wireless:** Wi-Fi 2x2 MIMO (802.11b/g/n/ac 2.4/5GHz) and Bluetooth 4.2
- **Dimensions:** 48mm x 40mm x 5mm

#### USB Accelerator

A USB accessory featuring the Edge TPU that brings ML inferencing to existing systems.

- **Supported host OS:** Debian Linux
- Compatible with Raspberry Pi boards
- **Supported Framework:** TensorFlow Lite, AutoML Vision Edge.
- **ML accelerator:** Google Edge TPU coprocessor
- **Connector:** USB 3.0 Type-C\* (data/power)
- **Dimensions:** 65 mm x 30 mm

Figure 2: Coral products with edge TPU (Image source: <http://coral.withgoogle.com>)

## Coral

Coral is a new platform for creating products with on-device ML acceleration. The first product here features Google's Edge TPU in SBC and USB form factors. TensorFlow Lite is officially supported on this platform, with the salient features being very fast inference speed, privacy and no reliance on network connection.

More details related to hardware specifications, pricing, and a getting started guide can be found at <https://coral.withgoogle.com>.

With these advances as well as a wider ecosystem, it's very evident that TensorFlow may become the leading framework for artificial intelligence and machine learning, similar to how Android evolved in the mobile world.

# How TensorFlow Makes Machines Learn

*This article covers TensorFlow, the machine learning tool developed by the Google Brain Team and open sourced in November 2015. Read on to learn what makes this tool suitable for machine learning applications and how to use it. You will understand why everyone at Google holds it in such high regard.*

A few years ago, the Google Brain team developed a software library to perform machine learning across a range of tasks. The aim was to cater to the needs of the team's machine learning systems, those that were capable of building and training neural networks. The software was meant to help such systems detect and decipher patterns and correlations, just like the way human beings learn and reason.

In November 2015, Google released this library under the Apache 2.0 licence, making it open for use, providing everyone the opportunity to work on their own artificial intelligence (AI) based projects. By June 2016, 1500 repositories on GitHub mentioned the software, of which only five were from Google.

## **Working with TensorFlow**

When you import TensorFlow into the Python environment, you get complete access over its classes, methods and symbols. You can take TensorFlow operations and arrange these into a graph of nodes called the computational graph. Typically, each node takes tensors as inputs and produces a corresponding output tensor. Values for the nodes get evaluated as and when a session is run. You can combine nodes with operations, which are also nodes in a certain form, to build more complicated computations.

## **Customise and improvise**

To tune TensorFlow for your machine learning application, you need to

construct the model such that it can take arbitrary inputs and deliver outputs accordingly. The way to do this with TensorFlow is to add variables, reflecting trainable parameters. Each of these has a type and an initial value, letting you tune your system to the required behaviour.

How do you know if your system is functioning exactly the way you intended it to? Simple... just introduce a loss function. TensorFlow provides optimisers that slowly change each variable so that loss functions can be minimised. There are also higher abstractions for common patterns, structures and functionality.

## **Multiple APIs for easier control**

As a new user to any software, it is important to enjoy the experience. TensorFlow is built with that mindset, with the highest-level application program interface (API) tuned for easy learning and usage. With experience, you will learn how to handle the tool, and know which modification will result in changing the entire functionality and in what way this will happen. It is then obvious to want to be able to work around the model and have fine levels of control over these aspects. TensorFlow's core API, which is the lowest-level, helps you achieve this fine control. Other higher-level APIs are built on top of this very core. The higher the level of the API, the easier it is to perform repetitive tasks and to keep the flow consistent between multiple users.

## **MNIST is ‘Hello World’ to machine learning**

The Mixed National Institute of Standards and Technology (MNIST) database is the computer vision data set that is used to train the machine learning system. It is basically a set of handwritten digits that the system has to learn and identify by the corresponding label. The accuracy of your model will depend on the intensity of your training. The broader the training data set, the better the accuracy of your model.

One example is the Softmax Regression model, which exploits the concept of probability to decipher a given image. As every image in MNIST is a

handwritten digit between zero and nine, the image you are analysing can be only one of the ten digits. Based on this understanding, the principle of Softmax Regression allots a certain probability of being a particular number, to every image under test.

#### About TensorFlow

**Developed by:** Google Brain Team  
**Last stable release:** 1.0  
**Repository:** GitHub  
**Written in:** Python and C++  
**Supports:** Linux, Mac and Windows  
**Licence:** Apache 2.0  
**Website:** [www.tensorflow.org](http://www.tensorflow.org)

## Smart handling of resources

As this process might involve a good bit of heavy lifting, just like other compute-heavy operations, TensorFlow offloads the heavy lifting outside the Python environment. As the developers describe it, instead of running a single expensive operation independently from Python, TensorFlow lets you describe a graph of interacting operations that run entirely outside Python.

## A few noteworthy features

Using TensorFlow to train your system comes with a few added benefits.

**Visualising learning:** No matter what you hear or read, it is only when you visually see something that the concept stays in your mind. The easiest way to understand the computational graph is, of course, to understand it pictorially. A utility called TensorBoard can display this very picture. The representation is very similar to a flow or a block diagram.

**Graph visualisation:** Computational graphs are complicated and not easy to view or comprehend. The graph visualisation feature of TensorBoard helps you understand and debug the graphs easily. You can zoom in or out, click on blocks to check their internals, check how data is flowing from one block to another, and so on. Name your scopes as clearly as possible in order to visualise better.

# The Capabilities of Tensor Virtual Machine, an Open Deep Learning Compiler Stack

*The Tensor Virtual Machine stack began as a research project at the SAMPL (System, Architecture, Machine learning and Programming Language) group of the Paul G. Allen School of Computer Science & Engineering, at the University of Washington in the US. This project is now driven by an open source community, and involves multiple industry and academic institutions.*

Tensor Virtual Machine or TVM is an open deep learning compiler stack to compile various deep learning models from different frameworks to the CPU, GPU or specialised accelerators. TVM supports model compilation from a wide range of frontends like TensorFlow, Onnx, Keras, Mxnet, Darknet, CoreML and Caffe2. TVM-compiled modules can be deployed on backends like LLVM (JavaScript or WASM, AMD GPU, ARM or X86), NVIDIA GPU (CUDA), OpenCL and Metal. TVM also supports runtime bindings for programming languages like JavaScript, Java, Python, C++ and Golang. With a wide range of frontend, backend and runtime bindings, this deep learning compiler enables developers to integrate and deploy deep learning models from any framework to any hardware, via any programming language.

## The TVM architecture

TVM provides a two-level optimisation mechanism, as shown in Figure 1. The first level of optimisation happens at the graph level after a model is imported. This optimisation provides graph level fusion, layout transformation and memory management. Later optimisation happens at the tensor level – at the code-generation layer and is based on the paper <https://arxiv.org/abs/1802.04799>.

The TVM stack comprises multiple layers, as shown in Figure 1. The top and user-facing layer is the framework layer. This is written in Python and contains various import modules for each framework. This layer converts the models from any framework (Tensorflow, Caffe2, etc) to a graph representation of TVM. At the computation graph optimisation layer, the graph representation is optimised by various passes like *precompute prune* which prunes the graph nodes that can be computed at compilation time, the *layout conversion* pass which adds the necessary layout conversion operations (or nodes) across layers if there is a layout mismatch between layers, and a *fusion* pass which joins the computation of multiple nodes into one based on certain rules. These optimisations reduce overall computation costs considerably.

The next layer in the stack is the Tensor compute description which basically generates a computation definition of each node in the graph, based on the inputs. The TOPI sub-module in the stack implements the computes for all the operators. The next layer in the stack is the schedule space and optimisations. This layer is very important in terms of low level as well as hardware-specific optimisations.

## **Build, integration and deployment**

As shown in Figure 2, TVM provides frontends to import trained deep learning models from various frameworks. The TVM compiler outputs a library which contains the operator (layer) compute definitions, the graph represented in JSON format, and a param blob which contains all the parameters of the model. TVM also provides various programming language binding libraries or packages, which can be imported or linked against, to load the compiled artifacts. Depending on the target hardware configured, the TVM runtime can be as small as approximately 300kB.

To demonstrate the end-to-end capabilities of the TVM compiler stack here, let us look at an example of importing, compiling and running a TensorFlow vision model, MobilenetV1, on x86 and NVIDIA (CUDA) targets.

## Step 1: The setup

TVM can be setup by building from source or using Docker. The simple steps given below can set up the environment for us. Please refer to <https://docs.tvm.ai/install/index.html> for other ways of setting up.

Download the source as follows:

```
test@test-pc:~$git clone --recursive https://github.com/dmlc/tvm
```

Build and run Docker using the following code:

```
test@test-pc:~$ cd tvm  
test@test-pc:~$ ./docker/bash.sh tvmai/demo-cpu
```

You may choose to launch Jupyter, if needed, inside Docker, as follows:

```
test@test-pc:~$jupyter notebook
```

One can verify the setup by just importing TVM as shown below, in Notebook or in a Python shell.

```
import tvm
```

## Step 2: Downloading the TensorFlow MobilenetV1 model

Various TensorFlow official models can be found at <https://github.com/tensorflow/models/tree/master/research/slim>. Use the following link to download the MobilenetV1 official model from TensorFlow: [http://download.tensorflow.org/models/mobilenet\\_v1\\_2018\\_02\\_22/mobilenet\\_v1\\_1.0\\_224.tgz](http://download.tensorflow.org/models/mobilenet_v1_2018_02_22/mobilenet_v1_1.0_224.tgz). Extract the downloaded archive and look for *mobilenet\_v2\_1.0\_224\_frozen.pb*.

This is the Protobuf format for a frozen model after training. Latest version of TVM also supports importing Tensorflow saved bundle. Model-specific information like input shapes, input and output node names is needed to

compile a model from TensorFlow. For TensorFlow, this information is available in the file below, from Mobilenet. The input shape for Mobilenet is (1, 244, 244, 3), which is the image resolution

```
cat mobilenet_v1_1.0_224_info.txt
Model: mobilenet_v1_1.0_224
Input: input

Output: MobilenetV1/Predictions/Reshape_1
```

### Step 3: Importing and compiling

The latest version of TVM provides a sub-module relay (*tvm.relay*) which contains all the frontend import utilities. Please refer to the code snippet given below to import and build the TensorFlow model on TVM.

```
# import tvm and tensorflow
import tvm
import tensorflow as tf

# We want to build TVM for llvm target (x86)
target = 'llvm'

# Import the tensorflow model.
with tf.gfile.FastGFile(os.path.join("mobilenet_v1_1.0_224_frozen.pb"), 'rb') as f:
    graph_def = tf.GraphDef()
    graph_def.ParseFromString(f.read())
    graph = tf.import_graph_def(graph_def, name='')

# Call the utility to import the graph definition into default graph.
graph_def = tf_testing.ProcessGraphDefParam(graph_def)

# Add shapes to the graph.
```

```
# Alternatively can use “add_shapes=True” while exporting graph from
Tensorflow.

with tf.Session() as sess:
    graph_def = tvm.relay.testing.tf.AddShapesToGraphDef(sess,
'MobilenetV1/Predictions/Reshape_1')

# Set input shape for graph input.
shape_dict = {'input': (1, 244, 244, 3)}

# Import graph through frontend
sym, params = relay.frontend.from_tensorflow(graph_def, shape=shape_dict)

# sym : represents tvm symbol graph constructed from imported model.

# params : graph parameters imported from model.

# Compile the model on TVM.
# The target here indicates the compiler to build the output for ‘llvm’
graph, lib, params = relay.build(sym, target=target, params=params)
```

## Step 4: Saving the compiler output

In the above step, the build process has resulted in a graph, *lib* and *params*. The graph is an object which holds the compiler graph. *lib* represents the library for the ‘llvm’ target and *params* represents the parameters for the model. Please refer to the code snippet below to save the compilation output to the disk.

```
# nnvm is part of TVM which is old version of compiler.
# we use the save_param_dict from here.
import nnvm

# Save the model as a library.
lib.export_library("libmobilenet.so")
```

```
# Save the graph definition as a JSON.  
with open("mobilenet.json", "w") as fo:  
    fo.write(graph.json())  
  
# Save the params.  
with open("mobilenet.params", "wb") as fo:  
    fo.write(nnvm.compiler.save_param_dict(params))
```

## Step 5: Loading and executing

The saved outputs from the compilation process are a library, a JSON and a *params* binary file. We may take these executables as targets for deploying data files. Along with these, we need a target-specific application to load the compiled model on the target. In our case, the target is x86 and for the deployment explanation I am choosing Python. The code snippet which follows with inlined documentation shows a Python application to deploy and infer.

```
# Load the module  
loaded_lib = tvm.module.load("libmobilenet.so")  
# Read graph and params.  
  
loaded_json = open("mobilenet.json").read()  
loaded_params = bytearray(open("mobilenet.params", "rb").read())  
  
# graph runtime initializes runtime from loaded module,  
# graph and on given context. In this case the context is CPU.  
from tvm.contrib import graph_runtime  
module = graph_runtime.create(loaded_json, loaded_lib, tvm.cpu(0))  
  
# Initialize the parameters.  
params = nnvm.compiler.load_param_dict(loaded_params)  
module.load_params(loaded_params)  
  
# Initialize some random data for model input.
```

```
input_data = np.random.uniform(size=(1, 244, 244, 3)).astype('float32')
# Set model input
module.set_input(x=input_data)

# Execute the Model
module.run()

# Get the first output
out = module.get_output(0)

# out is an NDArray type and out.asnumpy() can
# return an numpy array for the same.
```

The example given above explains the import, compilation and execution on target for a TensorFlow model on a x86 target using a Python runtime interface. Given below are a few choices for the usage of other options for frontends, hardware targets and programming languages.

We can compile various different target devices by changing *target='llvm'* in Step 3 above. TVM supports a wide range of targets like 'llvm', 'cuda', 'opencl', 'metal', 'rocm', 'vulkan', 'nvptx', 'llvm-device=arm\_cpu', 'opencl-device=mali' and 'aocl\_sw\_emu'. We may also need to specify *target\_host* while using accelerators from Linux hosts like CUDA, OpenCL, etc. We may need to pass *target\_host='llvm'*. Cross-compilation is also possible, bypassing additional options to LLVM. For example, we may pass *target\_host='llvm -target=aarch64-linux-gnu'* for an Android platform.

Apart from Python, TVM supports various programming languages while deploying a compiled module. Please refer to <https://docs.tvm.ai/deploy/index.html> for other programming languages.

## Cross-compilation and RPC

To be developer friendly, TVM supports RPC to cross-compile the model, and to



Figure 1: The TVM stack

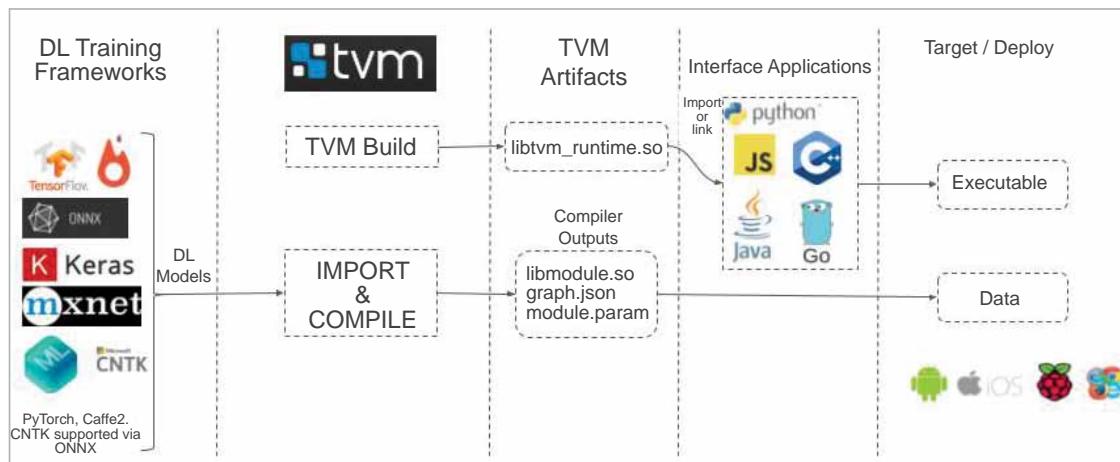


Figure 2: End-to-end flow diagram

deploy and test it. RPC enables faster development by remote-loading compiler output on target, and by setting input and getting the output from a target to the host seamlessly. If you are interested, refer to [https://docs.tvm.ai/tutorials/cross\\_compilation\\_and\\_rpc.html#](https://docs.tvm.ai/tutorials/cross_compilation_and_rpc.html#).

For Android developers, TVM provides an RPC application as a quick-start to compile a model, deploy it remotely via RPC and test it ([https://github.com/dmlc/tvm/blob/master/apps/android\\_deploy/README.md#build-and-installation](https://github.com/dmlc/tvm/blob/master/apps/android_deploy/README.md#build-and-installation)).

## AutoTVM

Apart from the standard compilation process, TVM also provides a framework to infer the best hardware parameters customised for the real underlying hardware. The `tvm.autotvm` sub-module provides various APIs for this. Refer to <https://docs.tvm.ai/api/python/autotvm.html?highlight=autotvm#module-tvm.autotvm> to know more about AutoTVM.

## Versatile Tensor Accelerator (VTA)

VTA is an open, generic and customisable deep learning accelerator with a complete TVM based compiler stack. It was designed to expose the most salient and common characteristics of mainstream deep learning accelerators. Together, TVM and VTA form an end-to-end hardware-software deep learning system stack that includes hardware design, drivers, a JIT runtime, and an optimising compiler stack based on TVM. Do refer to <https://docs.tvm.ai/vta/index.html> to know more about VTA.

## Benchmark

Benchmark information for various models like densenet, mobilenet, resnet, squeezeNet, etc, on different platforms like ARM CPU, ARM GPU, NVIDIA GPU and AMD GPU is available at <https://github.com/dmlc/tvm/wiki/Benchmark>.

# An Introduction to TensorFlow Programming in Python

*Deep learning is now widely used for the development of intelligent systems and has become a powerful tool for Big Data analysis. TensorFlow is the leading open source software for deep learning and is used for computer based natural language processing (NLP), computer vision, speech recognition, fault diagnosis, predictive maintenance, mineral exploration and much more.*

This article will acquaint readers with the basic environment of TensorFlow, its computational library, and with dataflow graphs, which will help them with its advanced applications. Since TensorFlow's computational environment is graph based processing, it is of utmost necessity to understand it at the outset. Here we shall learn how graph based computing in TensorFlow is performed within a Python Anaconda environment. TensorFlow building blocks are constants, placeholders and variables, all of which form the graph based machine learning computation environment where computing operations interact with each other.

A higher level TensorFlow API assists in building prototype models, but the knowledge of lower level TensorFlow core is valuable for experimentation and debugging code. It gives an inner view of the operation model of the code, which helps us to understand the code using higher level APIs.

## **Graphs and sessions**

TensorFlow uses a dataflow graph to represent all computations in terms of the dependencies between individual operations. At the outset, programming requires a dataflow graph to define all operations, after which a TensorFlow

session is created to run parts of the graph across a set of local and remote devices. High level APIs such as *tf.estimator.Estimator* and Keras hide the details of graphs and sessions from the end user. Low level programming is useful to understand how the graph model works under the process session.

In a dataflow graph, the nodes represent units of computation and the edges represent the data consumed or produced by a computation. For example, in a matrix multiplication operation, *tf.matmul* is a node in a TensorFlow graph and the multiplicand, multiplier and the result of multiplication are the three edges. This dataflow graph processing environment is helpful for distributed and parallel computing. It also speeds up the compilation process to convert the graph to code. Since the dataflow graph is a language-independent representation of the entire process environment, it is portable amongst different programming platforms. The dataflow representation of a process environment can be saved in one programming environment and transported to another programming environment.

The TensorFlow token *tf.Graph* contains dual information — the graph structure and graph collection. Using nodes and edges, the graph structure represents the composition of each individual operation, but it does not prescribe the methods of their use. Graph collections, on the other hand, provide a general mechanism for storing a collection of metadata within the graph. The graph collection mechanism provides ways to add and view relevant objects to and from key fields.

## **Creating a graph**

In general, a TensorFlow program starts with a graph building phase. During this process, the API adds a new node and edge to a default graph instance. For example, a *ts.constant(x)* creates a single operation to produce x, then adds the value to the default graph and returns a tensor that represents the constant value. In the case of a variable, *tf.Variable(0)* adds an operation to store a writeable tensor value. The variable survives in between two session runs (*tf.Session.run*). If we consider a matrix multiplication operation

`tf.matmul(a,b)`, then it will create and add a matrix multiplication operation to the default graph to multiply matrices a and b. It will return a tensor `tf.Tensor` to represent the multiplication.

In most cases a default graph is sufficient to run a program, but in the case of multiple graphs, API `tf.estimator` is used to manage the default graph, and it uses different graphs for training and evaluation.

## Session

An interactive session TensorFlow class is used in interactive contexts, such as a shell. This is convenient in interactive shells and IPython notebooks, as it is not required to pass an explicit session object to run an operation. The following example shows how a tensor constant c can be evaluated without an explicit call to a ‘session run’ operation. ‘Method close’ terminates an open interactive session.

```
sess = tf.InteractiveSession()
a = tf.constant(5.0)
b = tf.constant(6.0)
c = a * b
# We can just use 'c.eval()' without passing 'sess'
print(c.eval())
sess.close()
```

In the case of a regular session activation using a `with` statement, in non-interactive programs the tensor object is executed as follows:

```
a = tf.constant(5.0)
b = tf.constant(6.0)
c = a * b
with tf.Session():
    # We can also use 'c.eval()' here.
    print(c.eval())
```

Session method *as\_default* returns a context manager that makes the session object a default session. When used by the *with* keyword *tf.Operation.run* or *tf.Tensor.eval* should be executed in this session.

```
c = tf.constant(599)
sess = tf.Session()
with sess.as_default():
    #assert tf.get_default_session() is sess

    print(c.eval())
```

Since the *as\_default* context manager does not close the session upon exit from the context, it is mandatory to close the session explicitly on exiting the context. This can be avoided by invoking the session with *tf.Session()* directly within the ‘with statement’. This class closes the session upon exiting from the context.

The default session is a property of the current thread. To attach the default session to a new thread, it is necessary to explicitly use *with sess.as\_default()* in that thread. In the case of multiple graphs, if *sess.graph()* is different from the default graph (*tf\_get\_default\_graph*), it is necessary to set *sess.graph.as\_default()* for that graph.

## ***list\_devices***

Session method *list\_devices()* lists available devices in an activated session.

```
devices = sess.list_devices()
for d in devices:
    print(d.name)
```

This will display the full name of the device along with its type and the maximum amount of memory available. For example, a typical display of the above script may be */job:localhost/replica:0/task:0/device:CPU:0*.

## Slice

This operation is performed with `tf.slice()`, and it extracts a slice from a tensor object starting at the marked template location. The slice size is represented as a tensor shape, where `size[i]` is the number of elements of the *i*th dimension of the input that are to be sliced. The starting location `begin` of a slice is an offset in each dimension of the input. This starting location is zero-based, whereas the size is one-based. A value -1 of `size[i]` indicates that all remaining elements in dimension *i* are included in the slice and can be written as:

```
size[i] = input.dim_size(i) - begin[i]
```

For example:

```
t = tf.constant([[1, 1, 1], [2, 2, 2]],  
                [[3, 3, 3], [4, 4, 4]],  
                [[5, 5, 5], [6, 6, 6]])  
  
init_op = tf.global_variables_initializer()  
  
with tf.Session() as sess:  
    sess.run(init_op)  
  
    tf.slice(t, [1, 0, 0], [1, 1, 3])  
    tf.slice(t, [1, 0, 0], [1, 2, 3])  
    tf.slice(t, [1, 0, 0], [2, 1, 3])  
  
    init_op = tf.global_variables_initializer()  
  
    with tf.Session() as sess:  
        sess.run(init_op)  
        print("Slice 1:", sess.run(tf.slice(t, [1, 0, 0], [1, 1, 3])), "\n")  
  
    with tf.Session() as sess:  
        sess.run(init_op)  
        print("Slice 2:", sess.run(tf.slice(t, [1, 0, 0], [1, 2, 3])), "\n")
```

```
withtf.Session() as sess:  
    sess.run(init_op)  
    print("Slice 2:", sess.run(tf.slice(t, [1, 0, 0], [2, 1, 3])), "\n")  
  
Output:  
Slice 1: [[[3 3 3]]]  
Slice 2: [[[3 3 3]  
          [4 4 4]]]  
Slice 2: [[[3 3 3]]]  
[[[5 5 5]]]
```

## Placeholder

A placeholder is simply a variable that we will assign data at a later state. It allows us to create our operations and build our computation graphs, without needing the data. In TensorFlow terminology, we then feed data into the graph through these placeholders.

```
import tensorflow as tf  
x = tf.placeholder("float", None)  
y = x + 2.5  
withtf.Session() as session:  
    result = session.run(y, feed_dict={x: [1, 2, 3]})  
    print(result)
```

Output:

```
[3.5 4.5 5.5]
```

The first line creates a placeholder called x. It reserves a place in memory to store values at a later stage. This is used only to define the structure in the memory and no initial value is assigned into that location. Then a tensor call is used for adding x with 2.5. At a later stage, one can use this storage space and the defined operation to perform a vector addition. For this it is necessary to initiate a session and execute the defined operation within the session for a set of values of x.

Running `y` requires values of `x`. This can be assigned to run inside the `feed_dict` argument. Here, values of `x` are [1, 2, 3]. The run of the session of `y` will produce [3.5 4.5 5.5].

A larger graph consisting of numerous computations can be divided into small segments of atomic operations, and each of these operations can be performed individually. Such partial executions of a larger graph operation are an exclusive feature of TensorFlow and are not available in many other libraries that do similar jobs. During operation, placeholders do not require any static shape and can hold values of any length.

```
import tensorflow as tf
x = tf.placeholder("float", None)
y = x + 2.5
with tf.Session() as session:
    result = session.run(y, feed_dict={x: [1, 2, 3, 4]})
print(result)
```

This script will run for four values of vector `x` and will produce: [3.5 4.5 5.5 6.5]. To perform the same operation of adding 2.5 with an array of three columns with no upper boundary for the number of rows, one can modify the above script as follows:

```
x = tf.placeholder("float", [None, 3])
y = x + 5
with tf.Session() as session:
    x_data = [[2, 3, 1],
              [4, 7, 6],]
    result = session.run(y, feed_dict={x: x_data})
print(result)
Output:
[[ 7.  8.  6.]
 [ 9. 12. 11.]]
```

The placeholder can be extended to an arbitrary number of *None* dimensions. For example, to load an RGB image, a placeholder needs to have three slices to store all the three RGB image planes. Therefore, the placeholder needs two *None* for the first two dimensions and three for the last dimension. Then TensorFlow's slice method can be used to take a sub-segment out of the image.

```
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
import os

# First, load the image again
dir_path = os.path.dirname(os.path.realpath(__file__))

filename = dir_path + "\MarshOrchid.jpg"

print(dir_path)

raw_image_data = mpimg.imread(filename)
print(raw_image_data.shape)
plt.imshow(raw_image_data)
plt.show()

image = tf.placeholder("int32", [None, None, 3])
slice = tf.slice(image, [0, 0, 0], [300, -1, 1])

withtf.Session() as session:
    result = session.run(slice, feed_dict={image: raw_image_data})
    print(result.shape)

    plt.imshow(result)
    plt.show()
```

## Variables

In TensorFlow, variables are used to manage data. A TensorFlow variable

is the best way to represent shared and persistent tensor states to be manipulated by a program. Specific operations are required to read and modify the values of this tensor. Since a *tf.Variable* exists outside the context of a single *session.run* call, modified values are visible across multiple *tf.Session*, and multiple users can view the same values. These variables can be accessed via the *tf.Variable* class objects.

Many parts of the TensorFlow library use this facility. For example, when a variable is created, it is added by default to collections representing global variables and trainable variables. When in later stages *tf.train.Saver* or *tf.train.Optimizer* are created, the variables in these collections are used as the default arguments.

## Variable initialisers

Global tensor variables can be initialised using the following TensorFlow methods:

```
tf.global_variables_initializer  
tf.initializers.global_variables  
tf.initializers.global_variables()
```

There is an alternative shortcut to implicitly initialise a variable list. The shortcut is *tf.variables\_initializer(var\_list, name='init')*. If there is no definition of the variables before calling *tf.global\_variables\_initializer*, the variable list is essentially empty.

The code below illustrates this:

```
import tensorflow as tf  
  
with tf.Graph().as_default():  
    # Nothing is printed  
    for v in tf.global_variables():  
        print("List Global variable 1", v, "\n")
```

```
init_op = tf.global_variables_initializer()

a = tf.Variable(0)
b = tf.Variable(0)
c = tf.Variable(0)

init_op = tf.global_variables_initializer()

# 3 Variables are printed here
for v in tf.global_variables():
    print("List Global Variable 2", v, "\n")

with tf.Session() as sess:
    sess.run(init_op)
    print("List session", sess.run(a), "\n")
```

Execution of this script will not generate the variable list of the first loop, whereas the second for-loop will iterate four times over the global variable list and will display the variable graph in four lines.

```
init_op = tf.global_variables_initializer()
```

After this, the following lines:

```
a = tf.Variable(0)
b = tf.Variable(0)
c = tf.Variable(0)
d = tf.Variable(0)
```

...will initialise all the global variables lists for further processing of the graph.

## A simple mathematical operation

Here is an example of a simple TensorFlow mathematical operation using

variables a and b:

```
a = tf.Variable([4])
b = tf.Variable([7])
with tf.Session() as session:
    session.run(tf.global_variables_initializer())
    b = a + b
    result = session.run(a)
    print(a)
    result = session.run(b)
    print(b)
    print(session.run(a))
    print(session.run(b))
```

TensorFlow software is a new programming concept designed on graph based computing. This article is an introductory tutorial to understand this open source programming environment. The building blocks of this computing environment have been discussed with simple examples. The reader will hopefully get an idea of TensorFlow programming from this brief introduction.

# **Building Deep Learning Models with TensorFlow**

*Deep learning is impacting and revolutionising the tech industry. Many applications used on a day-to-day basis have been built incorporating deep learning. This article explains how the popular TensorFlow framework can be used to build a deep learning model.*

Let's assume the reader has the requisite knowledge of deep learning models and algorithms. There are various frameworks that are used to build these deep learning (neural networks) models, with TensorFlow and Keras being the most popular. Figures 1 and 2 show the adoption levels and the support community for all these frameworks.

## **TensorFlow**

TensorFlow code *depicts* 'computations' and doesn't actually perform them. To run or execute TensorFlow code, we need to create 'tf.Session' objects in Python.

**Tensors:** Tensors are mathematical constructs used especially in fields like physics and engineering. Traditionally, tensors have made less inroads into computer science. All these years computer science was more related to discrete mathematics and logical reasoning. But recent developments in machine learning have changed all that, and introduced the vector based mathematics and calculus of tensors. Here are a few basics:

- 1) Scalars are rank-0 tensors.
- 2) A Rank-1 tensor is a simple vector —all row vectors (1,2) and column vectors (2,1) are of these shapes.
- 3) A Rank-2 or 2D tensor is a simple 2X2 matrix. An example is: co-ordinates

of a plane. A black and white image can be a 2D tensor.

- 4) Similarly, we have Rank-3 or 3D tensors and Rank-4 or 4D tensors, etc. A colour image of size (255,255, 3) is a 3D tensor (Note: RGB is a three-colour channel). A video of a few minutes duration is an example of a 4D tensor, assuming that we have 60 frames per second, and RGB channels the size of (255, 255, 3, 3600).

 **Note:** The respective version of TensorFlow can be installed on your select choice of environment like Anaconda and Jupyter notebooks, Spyder, or any select framework that supports TensorFlow. The latest supported and stable version of TensorFlow is r1.13 or r1.12. TensorFlow 2.0 Alpha is also available now.

 **Note:** Make sure the respective Python version and matching TensorFlow version are both installed in your environment.

TensorFlow is loaded with computational as well as basic features like:

- Constants
- Variables
- Placeholders

**Constants:** The value of constants does not change. For example, in the 'Hello World' code we have initialised two constants and the value is fixed.

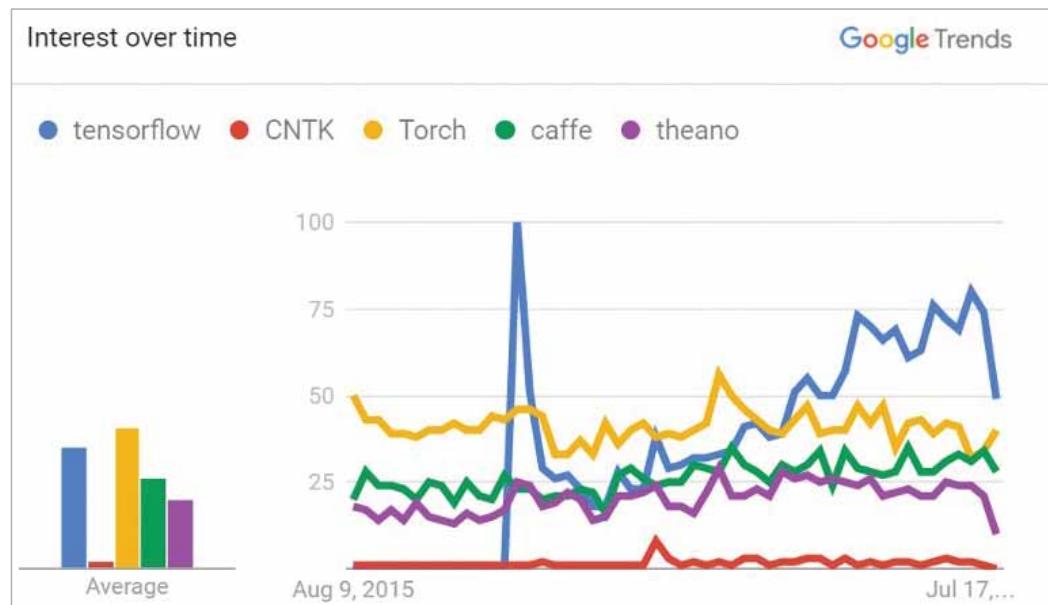


Figure 1: Interest in TensorFlow usage (Source: <https://towardsdatascience.com/what-happened-at-the-tensorflow-dev-summit-2017-part-1-3-community-applications-77fb5ce03c52>)

Table 1: A comparison of the various deep learning frameworks

Index	TensorFlow	Caffe2	Theano	Keras	Torch	CNTK
1	Google created TensorFlow some time back primarily to replace Theano. Both these libraries are quite similar. It was said that Ian Goodfellow, who created Theano, also created TensorFlow. Google made it open source in 2015.	Caffe2 is the latest version of this model. This was created by Yangqing Jia who works at Facebook. This is the second most backed DL framework from Facebook after Torch/PyTorch.	Theano is a deep learning library framework developed by the Université de Montréal in 2007.	Keras is a Python based framework for deep learning and was developed by François Fleuret, who works for Google.	PyTorch is an open source machine learning library for Python, based on Torch. It is developed by FB's AI research group-- Adam, Sam Gross, Soumith, and Gregory.	This computational toolkit was developed by Microsoft's research team during 2015-16.
2	The most used framework in the DL fraternity.	Generally preferred for RNN based models.	Cross-platform support.	Second most popular tool for DL.	Old ML based framework.	Most of the MS Windows based applications support this.
3	Developed using C/C++ backend engines. Performance is very good.	Developed using C++.	Performance is not that good for complex models.	Performance is good for most of the neural network layers.		Supports C++ and Python.
4	Supports both CPUs and GPUs across platforms.	Supports both CPUs and GPUs.	Supports both CPUs and GPUs.	Supports both CPUs and GPUs.		
5	Very good community support, and it provides good computation based features.	Backed by Facebook.		Many features are built in and it has good community support.		MIT licence and small community.
6	C++/ Python support.	Python/Java support.	Python support.	Python support.	Python support.	C++/Python support.

**Variables:** These are objects in TensorFlow, of which the value will be 're-filled' every time they are called or run in a session. These variables will maintain a fixed state in a graph. The `tf.Variable()` is used to declare or call variables in TensorFlow.

**Placeholders:** These are built-in structures for feeding input data. Sometimes these are thought of as empty variables that will be filled with data in the future. Placeholders have an optional 'shape' argument. The default option is `None`; otherwise, it will be assumed to be of any size.

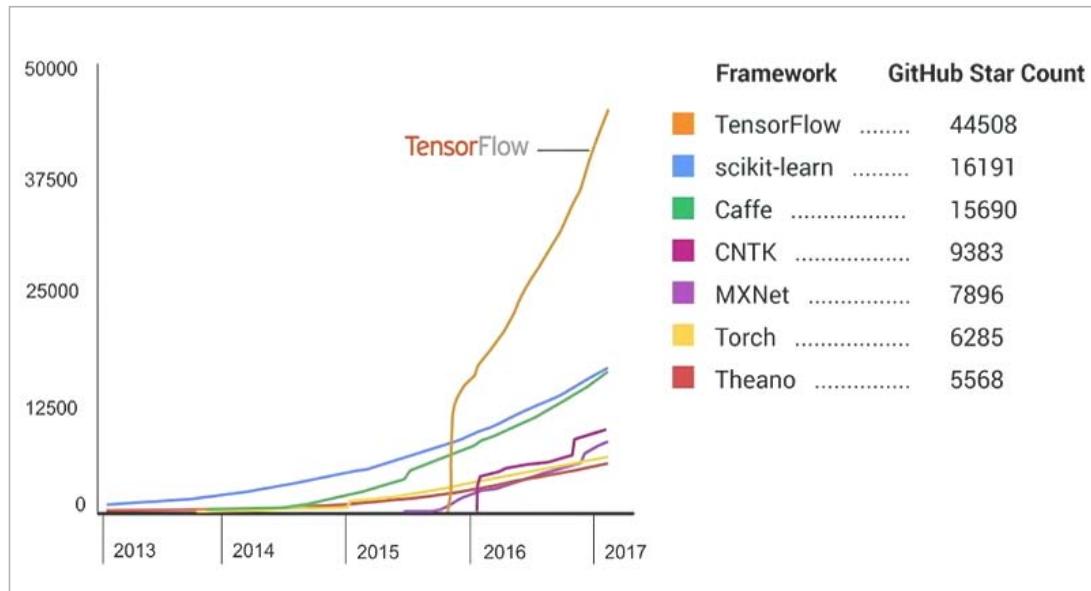


Figure 2: TensorFlow adoption (Source: <https://medium.com/@jrodrthoughts/five-deep-learning-frameworks-that-you-should-know-about-and-some-data-to-back-it-up-4480d31c86a8>)

```
In [3]: import tensorflow as tf
tf.InteractiveSession()

Out[3]: <tensorflow.python.client.session.InteractiveSession at 0x1e4752552b0>

Constant Tensors

In [4]: tf.zeros(5)
Out[4]: <tf.Tensor 'zeros_1:0' shape=(5,) dtype=float32>

In [6]: a_const = tf.zeros(5)
a_const.eval()
Out[6]: array([0., 0., 0., 0., 0.], dtype=float32)

In [7]: b_const = tf.zeros((2,4))
b_const.eval()
Out[7]: array([[0., 0., 0., 0.],
              [0., 0., 0., 0.]], dtype=float32)

In [9]: c_const = tf.ones((2,3))
c_const.eval()
Out[9]: array([[1., 1., 1.],
              [1., 1., 1.]], dtype=float32)
```

Figure 3: TensorFlow constants

```
Out[10]: 3

In [11]: import tensorflow as tf

In [12]: hello = tf.constant("Hello")
world = tf.constant("World")
helloworld = hello + world

In [13]: helloworld.eval()
Out[13]: b'HelloWorld'
```

Figure 4: TensorFlow *HelloWorld*

**Sessions in TensorFlow:** Any tensor must reside in the computer memory in order to be useful to computer programmers. Once this interactive session is loaded, we are good to program. `tf.InteractiveSession()` is used to start the session and until we close or come out of this window, this session will be live. Figures 3, 4, 5 and 6 show the usage of constant, *Hello World* of TensorFlow, variables and placeholders with examples in the Jupyter notebook.

## TensorFlow graphs

TensorFlow has inbuilt features that help us to build algorithms, and computing operations that assist us to interact with one another. These interactions are nothing but graphs, also called computational graphs. TensorFlow optimises the computations with the help of the graphs' connectivity. Each of these graphs has its own set of nodes and dependencies. Each of the types in TensorFlow — like constants, variables and placeholders — creates graphs for connectivity and computation.

```
In [14]: val = tf.random_normal((1,5), 0, 1)

In [15]: my_var = tf.Variable(val, name='my_var')

In [19]: init=tf.global_variables_initializer()

In [25]: print("Before run: \n{}".format(my_var))

Before run:
<tf.Variable 'var:0' shape=(1, 5) dtype=float32_ref>

In [24]: with tf.Session() as sess:
    sess.run(init)
    after_var = sess.run(my_var)

print("\nAfter run:\n{}".format(after_var))

After run:
[[ 0.23448779  0.57947916  1.1972933   1.4584718  -0.08119686]]
```

Figure 5: TensorFlow variables

```
In [28]: import numpy as np
x_data = np.random.randn(5,10)
y_data = np.random.randn(10,1)

In [30]: with tf.Graph().as_default():
    x = tf.placeholder(tf.float32, shape=(5,10))
    w = tf.placeholder(tf.float32, shape=(10,1))
    b = tf.fill((5,1), -1.)
    xw = tf.matmul(x, w)

    xb = xw + b
    s = tf.reduce_max(xb)

    with tf.Session() as sess:
        outs = sess.run(s, feed_dict = {x: x_data, w:y_data})

    print("outs = {}".format(outs))
    outs = 1.6235082149505615
```

Figure 6: TensorFlow placeholders

Here is a coding example of linear regression using random variable generation:

```
import tensorflow as tf
print(tf.__version__)
import numpy as np
import matplotlib.pyplot as plt

%config InlineBackend.figure_format = 'svg'

C:\Users\Sony\Anaconda3\lib\site-packages\h5py\__init__.py:34: FutureWarning:
Conversion of the second argument of issubdtype from `float` to `np.floating`
is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).
type`.

from ._conv import register_converters as _register_converters

1.10.0

# Below are hyper parameters like learning rate, number of epochs, number of
samples
learning_rate = 0.01
epochs=100
n_samples = 30

train_x = np.linspace(0, 20, n_samples)
train_y = 3 * train_x + 4 * np.random.randn(n_samples)
## Plot before starting the training
plt.plot(train_x, train_y, 'x')
plt.plot(train_x, 3 * train_x)
plt.show()

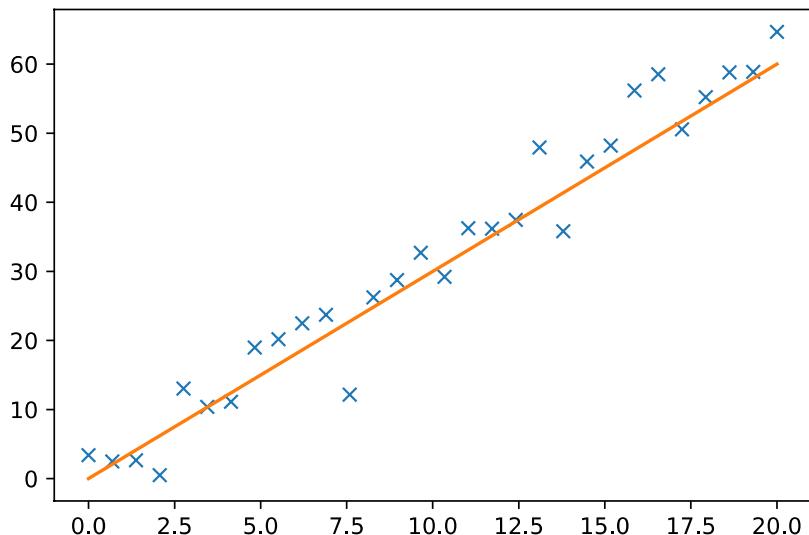
# Initialize training weights in placeholder
```

```
X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)

# Now initialize training biases
W = tf.Variable(np.random.randn(), name ='weights')
B = tf.Variable(np.random.randn(), name='bias')
# The prediction can be based input X, Weight W and bias B
prediction = tf.add(tf.multiply(X, W), B)

# Cost function(To minimize the loss as much as possible)
# Use mean square error

cost = tf.reduce_sum((prediction - Y) ** 2) /(2 * n_samples)
optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)
#Global initializer for Tensorflow variables
```



```
init = tf.global_variables_initializer()
# tf.session for all the computation
# here the no.of epochs are run

with tf.Session() as sess:
    sess.run(init)
```

```

for epoch in range(epochs):
    for x, y in zip(train_x, train_y):
        sess.run(optimizer, feed_dict = {X: x, Y:y})
    if not epoch % 20:
        c = sess.run(cost, feed_dict={X:x, Y:y})
        w = sess.run(W)
        b = sess.run(B)
        print(f'epoch: {epoch:04d} cc={c:.4f} w={w:.4f} b={b:.4f}')
weight = sess.run(W)
bias = sess.run(B)
plt.plot(train_x, train_y, 'o')
plt.plot(train_x, weight * train_x + bias)
plt.show()

```

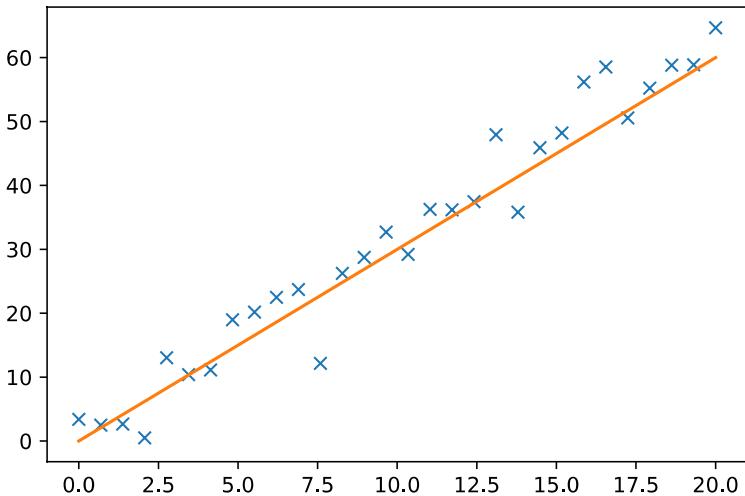
epoch: 0000 cc=5.4618 w=2.3091 b=0.3990

epoch: 0020 cc=0.0384 w=3.1360 b=0.4447

epoch: 0040 cc=0.0381 w=3.1375 b=0.4217

epoch: 0060 cc=0.0378 w=3.1389 b=0.3999

epoch: 0080 cc=0.0375 w=3.1402 b=0.3792



To get familiar with the basics of machine learning and deep learning, go to <https://opensourceforu.com/2017/12/deep-learning-using-algorithms-to-make-machines-think/>.

# **TensorFlow Hub: A Machine Learning Ecosystem**

*Machine learning (ML) is the new ‘in’ thing in the world of computer science. For a newbie to get into it is not very easy, however. TensorFlow Hub is one way in which newbies can work with tried and tested models, and hone their skills in this rapidly evolving field.*

Starting from scratch, machine learning requires four key components — algorithms, data, compute, and expertise. If you don’t have access to any one of these, then good luck to getting a great result! One viable alternative for small-scale projects is to reproduce pre-existing experiments relevant to their use cases, tweak them, and improve on the results. However, that is easier said than done.

Reproducibility in machine learning is a crisis that has been highlighted by top researchers and publications over the years. Recent times have seen a shift towards a more open paradigm. The field’s largest conference, NeurIPS, received 8,000 submissions last year, with organisers now encouraging code-sharing for papers submitted to them. This has been done in full consideration of existing proprietary libraries and dependencies that cannot accompany the code-release, i.e., code submission is encouraged even if it is ‘non-executable’ in external environments. The move has been widely perceived as a start to a more open culture in solving problems, working with data sets, and in general, promoting transparency and reproducibility.

“Machine learning is really hard to do, so we built TensorFlow Hub (TFHub) to allow people to reuse machine learning. We are letting people pack up good machine learning and share it with the world!” Jeremiah Harmsen from Google Zurich excitedly explains. He adds that TFHub gives people ‘smaller pieces’ of machine learning models, making it more likely for developers to reuse code.

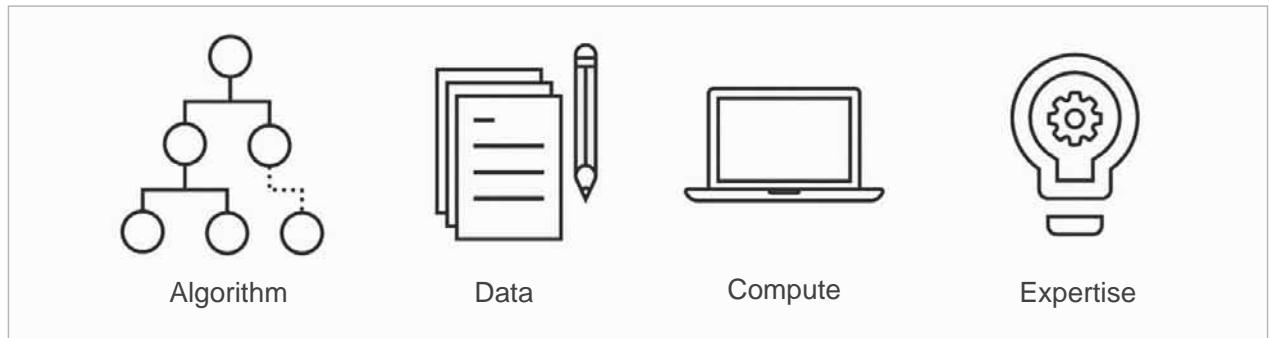


Figure 1: Prerequisites for machine learning [Source: Google Inc.]



Figure 2: The most overused developer excuse for reproducibility! [Source: teepublic.com]

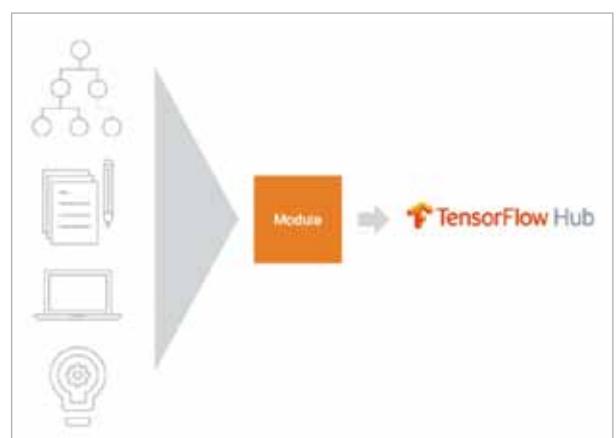


Figure 3: An overview of TFHub's functionality [Source: Google Inc.]

It is true that models require a specific type and format of input to produce a specific output. The minute you try to modify a model, you're likely to run into issues. With TFHub, models and modules are treated as analogues to binaries and libraries. With this, it becomes well-defined in terms of importing modules and using functionality from within it. Modules are then composable, reusable, and retrainable.

In a talk at the TensorFlow Developer Summit, researchers at Google Zurich explained how you can customise a TFHub module for your own application, by using the example of an image classification model. Most image classifiers today have already been trained for hundreds of hours on millions of images. However, if given a custom task of making predictions related to a few types of rabbits, it is not possible to find a model that directly handles this for you. It becomes a complex problem to start to solve from scratch, but with TFHub

and three lines of code, you find yourself retraining a model that can give you much better results than you would probably have obtained had you started from scratch. This approach could also incorporate the weights and architectures of models that have taken researchers over 60,000 hours of effort to develop!

TFHub aims to foster engagement within the community and promote more transparency in machine learning. At the same time, it provides a one-stop shop for all models of TensorFlow; just add a few lines of code and you have a model running inference on new input data or better still, re-training on input

The screenshot shows the TensorFlow Hub website interface. On the left, there's a sidebar with categories: Text, Embedding, Image, Classification, Feature Vector, Generator, Other, Video, Classification, and Publishers (Google, DeepMind). The main content area has two sections: "Text embedding" and "Image feature vectors".

**Text embedding:**

- universal-sentence-encoder** By Google
  - text-embedding DAN English
  - Encoder of greater-than-word length text trained on a variety of data.
- universal-sentence-encoder-large** By Google
  - text-embedding Transformer English
  - Encoder of greater-than-word length text trained on a variety of data.
- elmo** By Google
  - text-embedding 1 Billion Word Benchmark ELMo English
  - Embeddings from a language model trained on the 1 Billion Word Benchmark.

[View more text embeddings](#)

**Image feature vectors:**

- imagenet/inception\_v3/feature\_vector** By Google
  - image-feature-vector ImageNet (ILSVRC-2012-CLS) Inception V3
  - Feature vectors of images with Inception V3 trained on ImageNet (ILSVRC-2012-CLS).
- imagenet/mobilenet\_v2\_140\_224/feature\_vector** By Google
  - image-feature-vector ImageNet (ILSVRC-2012-CLS) MobileNet V2
  - Feature vectors of images with MobileNet V2 (depth multiplier 1.40) trained on ImageNet (ILSVRC-2012-CLS).
- imagenet/resnet\_v2\_50/feature\_vector** By Google
  - image-feature-vector ImageNet (ILSVRC-2012-CLS) ResNet V2 50
  - Feature vectors of images with ResNet V2 50 trained on ImageNet (ILSVRC-2012-CLS).

[View more image feature vector modules](#)

[Video classification](#)

Figure 4: Take a look at the TFHub Web interface!

data to provide an even better, more fine-tuned performance. As expected, Google remains committed to pushing out more models and enabling a better Web experience on TFHub, as evidenced by the focus on providing resources akin to the ‘Universal Sentence Encoder’ module –“a successful example of speeding up the translation from fundamental machine learning science to application in the broader developer community.”

For many teams undertaking cutting-edge research, it is not feasible to publish their TFHub modules publicly. In such cases, TFHub allows developers to publish modules to, and consume from, private storage. Instead of referring to modules by their *tfhub.dev* URL, the User Guide explains how you can use a file system path to retrieve and use them in your code.

On the whole, with the introduction of TensorFlow 2.0 and the infrastructure offered by TensorFlow Hub, Google is doubling down on its commitment to provide developers with a complete package, along with bells and whistles, to perform cutting-edge machine learning. Other projects in this area include TensorFlow data sets that capitalise on the *tf.data* API in order to allow for easy pipelining and data manipulation, Tensorflow.js which offers in-browser training of neural networks among other awesome functionality, and Tensorboard which is a visualisation tool for machine learning.

TFHub has been surrounded by a lot of hype ever since its introduction, but it remains to be seen if the product is used as widely as it was expected to. The Google Zurich team is a fantastic team of engineers and researchers, who are working in tandem on TFHub, in what Dr Harmsen states is its first major infrastructure project. There are big things coming out of this lab, so it is going to be very exciting to see what is next!

# **How TensorFlow.js Defines, Trains Machine Learning Models**

*With artificial intelligence (AI) taking centre stage in the deployment of data, machine learning (ML), which is an important application of AI, gains importance. ML is much helped by the use of TensorFlow.js, an open source javaScript library this is used to define, train and run ML models.*

Machine learning is an application of artificial intelligence (AI) that provides systems with the ability to automatically learn and improve from experience, without being explicitly programmed. Machine learning (ML) focuses on the development of computer programs that can access data and use it to learn for themselves.

Basically, ML systems process the data and the output to come up with rules that match the input data to the output.

## **TensorFlow**

TensorFlow is an open source software library for numerical computation and it uses data flow graphs. The graph nodes represent mathematical operations, while the graph edges represent the multi-dimensional data arrays (tensors) that flow between them.

TensorFlow was originally developed by researchers and engineers working at Google. The system is general enough to be applicable to problems related to a wide variety of domains.

## TensorFlow.js

Tensorflow.js is an open source library that makes use of JavaScript and a high-level layer API to define, train as well as run machine learning models entirely in the browser. This open source library is powered by WebGL, which provides a high-level layer API for defining models, and offers a low-level API for linear algebra and automatic differentiation.

### What can be done with TensorFlow.js

Listed below are some of the capabilities of TensorFlow.js.

- *The ability to import an existing, pre-trained model or inference:* If the user has an existing TensorFlow or Keras model that has been previously trained offline, it can be converted into the TensorFlow.js format and then loaded into the browser for inference.
- *The ability to retrain an existing model:* By making use of a small amount of data collected in the browser using a technique called image retraining, you can make use of transfer learning in order to augment an existing model that has previously been trained offline. This is one way to train an accurate model quickly, using only a small amount of data.
- *The ability to author models directly in the browser:* You can also use TensorFlow.js with JavaScript and a high-level layer API to define, train as well as run models entirely in the browser.

### Benefits of TensorFlow.js

Listed below are some of the benefits of TensorFlow.js:

- It supports WebGL, out-of-the-box.
- It supports GPU acceleration and therefore, behind the scenes, will accelerate your code when a GPU is available.
- All data stays on the client, which makes TensorFlow.js useful not only for low-latency inference but for privacy-preserving applications as well.
- Even from a mobile device, you can open your Web page, in which case your model can take advantage of sensor data.

## Core concepts of TensorFlow.js

TensorFlow.js provides low-level building blocks for machine learning as well as a high-level, Keras-inspired API for constructing neural networks. Let's take a look at some of the core components of the library.

**Tensors:** The central unit of data in TensorFlow.js is the tensor—a set of numerical values represented as an array of one or more dimensions. A tensor also contains the `shape` attribute, which defines the array's shape and essentially gives the count of elements per dimension.

The most common way to create a tensor is by using the `tf.tensor` function, as shown in the code snippet below:

```
// 2x3 Tensor
const shape = [2, 3]; // 2 rows, 3 columns
const a = tf.tensor([2.0, 3.0, 4.0, 20.0, 30.0, 40.0], shape);
a.print(); // print Tensor values
// Output: [[2,3,4 ],
//           [20, 30, 40]]
```

However, for constructing low-rank tensors, Google advises using the `tf.scalar`, `tf.tensor1d`, `tf.tensor2d`, `tf.tensor3d` and `tf.tensor4d` functions.

The following code example creates an identical tensor to the one shown in the previous code snippet using `tf.tensor2d`:

```
const c = tf.tensor2d([[2.0, 3.0, 4.0], [20.0, 30.0, 40.0]]);
c.print();
// Output: [[2,3,4 ],
//           [20, 30, 40]]
```

TensorFlow.js also provides functions for creating tensors with all values set to 0 (`tf.zeros`) or all values set to 1 (`tf.ones`).

```
// 3x5 Tensor with all values set to 0
const zeros = tf.zeros([3, 5]);
// Output: [[0, 0, 0, 0, 0],
//           [0, 0, 0, 0, 0],
//           [0, 0, 0, 0, 0]]
```

**Variables:** Variables are initialised with a tensor of values. Unlike tensors, however, variables are mutable. You can assign a new tensor to an existing variable using the `assign` method, as shown below:

```
const initialValues = tf.zeros([6]);
const biases = tf.variable(initialValues); // initialize biases
biases.print(); // output: [0, 0, 0, 0, 0, 0]

const updatedValues = tf.tensor1d([0, 1, 0, 1, 0, 1]);
biases.assign(updatedValues); // update values of biases
biases.print(); // output: [0, 1, 0, 1, 0, 1]
```

**Operations:** While tensors allow you to store data, operations (ops) allow you to manipulate that data. TensorFlow.js provides a wide array of operations suitable for general computations and ML-specific operations on tensors. Because tensors are immutable, operations return new tensors after computations are done—for example, a unary operator such as a square.

```
const data = tf.tensor2d([[2.0, 3.0], [4.0, 5.0]]);
const data_squared = data.square();
data_squared.print();
// Output: [[4, 9],
//           [16, 25]]
```

So is the case with binary operations such as `add`, `sub` and `mul`, as shown below:

```
const e = tf.tensor2d([[1.0, 2.0], [3.0, 4.0]]);
```

```
const f = tf.tensor2d([[5.0, 6.0], [7.0, 8.0]]);

const e_plus_f = e.add(f);
e_plus_f.print();
// Output: [[6 , 8 ],
//           [10, 12]]
```

TensorFlow.js has a chainable API; you can call operations on the result of operations.

```
const sq_sum = e.add(f).square();
sq_sum.print();
// Output: [[36 , 64 ],
//           [100, 144]]
```

All operations are exposed as functions in the main name space; so you could also do the following:

```
const sq_sum = tf.square(tf.add(e, f));
```

**Models and layers:** Conceptually, a model is a function that takes some input and produces some output. In TensorFlow.js, there are two ways to create models. You can use operations directly to represent the work of the model. For example:

```
// Define function
function predict(input) {
  // y = a * x ^ 2 + b * x + c
  // More on tf.tidy in the next section
  return tf.tidy(() => {
    const x = tf.scalar(input);

    const ax2 = a.mul(x.square());
```

```
const bx = b.mul(x);
const y = ax2.add(bx).add(c);

return y;
});

}

// Define constants: y = 2x^2 + 4x + 8

const a = tf.scalar(2);
const b = tf.scalar(4);
const c = tf.scalar(8);

// Predict output for input of 2

const result = predict(2);
result.print() // Output: 24
```

Or you can use APIs like *tf.model* and *tf.sequential* to construct a model. The following code constructs a *tf.sequential* model:

```
const model = tf.sequential();
model.add(
  tf.layers.simpleRNN({
    units: 20,
    recurrentInitializer: 'GlorotNormal',
    inputShape: [80, 4]
  })
);

const optimizer = tf.train.sgd(LEARNING_RATE);
model.compile({optimizer, loss: 'categoricalCrossentropy'});
model.fit({x: data, y: labels});
```

There are many kinds of layers available in TensorFlow.js. A few examples include *tf.layers.simpleRNN*, *tf.layers.gru* and *tf.layers.lstm*.

## Memory management

TensorFlow.js uses the GPU to accelerate math operations, so it's necessary to manage GPU memory while working with tensors and variables.

TensorFlow.js provides two functions for this.

**Dispose:** You can call *dispose* on a tensor or variable to purge it and free up its GPU memory:

```
const x = tf.tensor2d([[0.0, 2.0], [4.0, 6.0]]);  
const x_squared = x.square();  
  
x.dispose();  
x_squared.dispose();
```

**Tf.tidy:** Using *dispose* can be cumbersome when doing a lot of tensor operations. TensorFlow.js provides another function, *tf.tidy*, which plays a similar role to regular scopes in JavaScript, but for GPU-backed tensors. *tf.tidy* executes a function and purges any intermediate tensors created, freeing up their GPU memory. It does not purge the return value of the inner function.

*tf.tidy* takes a function to tidy up after:

```
const average = tf.tidy(() => {
```

*tf.tidy* will clean up all the GPU memory used by tensors inside this function, other than the tensor that is returned.

Even in a short sequence of operations like the one below, a number of intermediate tensors get created. So it is a good practice to put your math ops in a *tidy*!

```
const y = tf.tensor1d([1.0, 2.0, 3.0, 4.0]);
const z = tf.ones([4]);

return y.sub(z).square().mean();
});

average.print() // Output: 3.5
```

Using *tf.tidy* will help prevent memory leaks in your application. It can also be used to more carefully control when memory is reclaimed. The function passed to *tf.tidy* should be synchronous and also not return a *promise*.

*tf.tidy* will not clean up variables. As variables typically last through the entire life cycle of a machine learning model, TensorFlow.js doesn't clean them up even if they are created in a *tidy*; however, you can call *dispose* on them, manually.

# **How TensorFlow Can Be Used for Video Analytics**

*Object detection now plays a very important role in our lives, right from face detection and unlocking a smartphone to detecting bombs in places where people congregate, like airports, bus terminals, railway stations, etc. Such advanced features are a result of the application of machine learning and artificial intelligence. This article discovers how TensorFlow, a machine learning library, can be used to identify objects.*

Security and surveillance cameras are very widely used across organisations, so it is important to enhance their effectiveness while saving manpower costs and preventing human errors. In such scenarios, image/video analytics plays a very important role in performing real-time event detection, post-event analysis, and the extraction of statistical and operational data from the videos. Video analytics (VA) is the general analysis of video images to recognise unusual or potentially dangerous behaviour and events in real-time. It can perform three major tasks — provide information, offer assistance, and generate alerts.

Video analytics is widely used for suspicious object detection in schools, in the banking and financial sector, and in critical infrastructure protection. It is also used at heritage sites, commercial spaces, offices, government buildings, factories, airports, railway/metro stations, busy traffic intersections, stadiums, etc. Video analytics is used for monitoring people and vehicle flows. It can spot and prevent the crossing of the speed threshold and vehicular movement in the wrong direction. Other capabilities include crowd counting, incident

detection, licence plate recognition, safety alerts, people/object recognition, post-event analysis, lost object detection, etc.

Wikipedia defines many functional applications of video analytics, which are listed in Table 1.

Function	Description
Dynamic masking	Blocking a part of the video signal based on the signal itself, for example, because of privacy concerns.
Flame and smoke detection	IP cameras with intelligent video surveillance technology are being used to detect flames and smoke within 15 to 20 seconds or even less because of the built-in DSP chip.
Egomotion estimation	This is used to determine the location of a camera by analysing its output signal.
Motion detection	This determines the presence of relevant motion in the observed scene.
Shape recognition	This recognises shapes in the input video, for example, circles or squares. This feature is one aspect of more advanced functionalities such as object detection.
Object detection	Object detection determines the presence of a type of object or entity, for example a person or car.
Recognition	Face recognition and automatic number plate recognition make it possible to identify individuals or cars.
Style detection	Style detection is used in settings where a video signal has been produced, for example, for television broadcast.
Tamper detection	Tamper detection tells us whether the camera or output signal has been tampered with.
Video tracking	This feature is used to determine the location of persons or objects in the video signal, possibly with regard to an external reference grid.
Object co-segmentation	This involves the joint object discovery, classification and segmentation of targets in one or multiple related video sequences.

One of the major applications of video analytics is object identification, which is the process of identifying objects within images or videos. It is based on the concept that each object has unique features. These features help to classify the objects into different classes. Identification can be done either by using the machine learning (ML) approach or the deep learning (DL) method.

In the machine learning method, first the features are identified and then support vector techniques (SVM) are used for classification. Deep learning methods are based on convolution neural networks (CNN), which is a kind of neural network that has an input layer, output layer and multiple hidden layers. It uses mathematical model convolution to pass on the result to various successive layers. In this article, I will explain the process of object identification using TensorFlow, with a focus on deep learning methods.

TensorFlow is an open source software library for numerical computation using data flow graphs. The graph nodes represent mathematical operations, while the graph edges represent the multi-dimensional data arrays (tensors) that flow between them. This flexible architecture enables users to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device without rewriting code. TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the limits of the state-of-art in ML, and helps developers easily build and deploy ML-powered applications.

Deep learning is a part of machine learning in artificial intelligence. It works in a manner that is similar to how the human brain works to process data and to create patterns, which are used for decision making. Deep learning allows computational models to learn representation of data with multiple levels of abstraction. These models are composed of multiple processing layers. Deep learning methods have proved to be very effective in speech recognition, visual object recognition, object detection and many other domains.

```
File Edit View Search Terminal Help
surabhi@surabhi-seng:~$ pip install tensorflow
Collecting tensorflow
  Downloading https://files.pythonhosted.org/packages/d3/59/d88fe8c58ffb66aca21d03c0e290cd68327cc133591130c674985e98a482/tensorflow-1.14.0-cp27-cp27mu-manylinux1_x86_64.whl (109.2MB)
    100% |████████████████████████████████| 109.2MB 15kB/s
Collecting grpcio>=1.8.6 (from tensorflow)
  Downloading https://files.pythonhosted.org/packages/a5/46/5d08b6e26748ed6f3b5e93d980ea5daa63c3a8200b2ad270645b0e2f9566/grpcio-1.22.0-cp27-cp27mu-manylinux1_x86_64.whl (2.2MB)
    100% |████████████████████████████████| 2.2MB 724kB/s
Collecting mock>=2.0.0 (from tensorflow)
  Downloading https://files.pythonhosted.org/packages/85/d2/f94e68be6b17f46d2c353564da56e6fb89ef09faeefff3313a046cb810ca9/mock-3.0.5-py2.py3-none-any.whl
Collecting keras-applications>=1.0.6 (from tensorflow)
  Downloading https://files.pythonhosted.org/packages/21/56/4bcec5a8d9503a87e58e814c4e32ac2b32c37c685672c30bc8c54c6e478a/Keras_Applications-1.0.8.tar.gz (289kB)
    100% |████████████████████████████████| 296kB 1.2MB/s
Collecting wrapt>=1.11.1 (from tensorflow)
  Downloading https://files.pythonhosted.org/packages/23/84/323c2415280bc4fc880ac5050dddfb3c8062c2552b34c2e512eb4aa60f79/wrapt-1.11.2.tar.gz
Collecting protobuf>=3.6.1 (from tensorflow)
  Downloading https://files.pythonhosted.org/packages/30/bf/d7f8bc958f9eb4661498cde9f7e630da863e43278222d46105712dde4e13/protobuf-3.9.0-cp27-cp27mu-manylinux1_x86_64.whl (1.2MB)
    100% |████████████████████████████████| 1.2MB 929kB/s
Collecting keras-preprocessing>=1.0.5 (from tensorflow)
  Downloading https://files.pythonhosted.org/packages/28/6a/8c1f62c37212d9fc441a7e26736df51ce6f8e38455816445471f10da4f0a/Keras_Preprocessing-1.1.0-py2.py3-none-any.whl (41kB)
    100% |████████████████████████████████| 51kB 6.7MB/s
Collecting gast>=0.2.0 (from tensorflow)
  Downloading https://files.pythonhosted.org/packages/4e/35/11749bf99b2d4e3ccebd55ca22590b0d7c2c62b9de38ac4a4a7f4687421/gast-0.2.2.tar.gz
Collecting wheel (from tensorflow)
  Downloading https://files.pythonhosted.org/packages/bb/10/44230dd6bf3563b8f227dbf344c908d412ad2ff48066476672f3a72e174e/wheel-0.33.4-py2.py3-none-any.whl
Collecting tensorboard<1.15.0,>=1.14.0 (from tensorflow)
  Downloading https://files.pythonhosted.org/packages/f4/37/e6a7af1c92c5b68fb427f853b06164b56ea92126bcfd87784334ec5e4d42/tensorboard-1.14.0-py2-none-any.whl (3.1MB)
```

Figure 1: TensorFlow installation

```

File Edit View Search Terminal Tabs Help
surabhi@surabhi-seng:~/home/surabhi/Documents/Documents/Software/TensorFlow$ python -m tensorflow.test
[ OK ] ModelBuilderTest.test_create_faster_rcnn_models_from_config_faster_rcnn_with_matmul
[ OK ] ModelBuilderTest.test_create_faster_rcnn_models_from_config_faster_rcnn_without_matmul
[ OK ] ModelBuilderTest.test_create_faster_rcnn_models_from_config_faster_rcnn_without_matmul
[ OK ] ModelBuilderTest.test_create_faster_rcnn_models_from_config_mask_rcnn_with_matmul
[ OK ] ModelBuilderTest.test_create_faster_rcnn_models_from_config_mask_rcnn_with_matmul
[ OK ] ModelBuilderTest.test_create_faster_rcnn_models_from_config_mask_rcnn_without_matmul
[ OK ] ModelBuilderTest.test_create_faster_rcnn_models_from_config_no_matmul
[ OK ] ModelBuilderTest.test_create_rfcn_model_from_config
[ OK ] ModelBuilderTest.test_create_rfcn_model_from_config
[ OK ] ModelBuilderTest.test_create_ssdfpn_model_from_config
[ OK ] ModelBuilderTest.test_create_ssdfpn_model_from_config
[ OK ] ModelBuilderTest.test_create_ssdfpn_models_from_config
[ OK ] ModelBuilderTest.test_create_ssdfpn_models_from_config
[ OK ] ModelBuilderTest.test_invalid_faster_rcnn_batchnorm_update
[ OK ] ModelBuilderTest.test_invalid_faster_rcnn_batchnorm_update
[ OK ] ModelBuilderTest.test_invalid_first_stage_nms_iou_threshold
[ OK ] ModelBuilderTest.test_invalid_first_stage_nms_iou_threshold
[ OK ] ModelBuilderTest.test_invalid_model_config_proto
[ OK ] ModelBuilderTest.test_invalid_second_stage_batch_size
[ OK ] ModelBuilderTest.test_invalid_second_stage_batch_size
[ SKIPPED ] ModelBuilderTest.test_session
[ RUN ] ModelBuilderTest.test_session
[ OK ] ModelBuilderTest.test_unknown_faster_rcnn_feature_extractor
[ OK ] ModelBuilderTest.test_unknown_meta_architecture
[ OK ] ModelBuilderTest.test_unknown_meta_architecture
[ OK ] ModelBuilderTest.test_unknown_ssdfpn_feature_extractor
[ OK ] ModelBuilderTest.test_unknown_ssdfpn_feature_extractor

Ran 16 tests in 0.162s
OK (skipped=1)
surabhi@surabhi-seng:~/home/surabhi/Documents/Documents/Software/TensorFlow/tensorflow/models/research$ 

```

Figure 2: Testing the installation

In the subsequent section, I will discuss a very basic method of object identification using TensorFlow. This experiment has been carried on Ubuntu 18.04.3 with Python, TensorFlow and Protobuf 3.9.

The following steps can be used for object detection using TensorFlow. This will identify objects kept in the *test\_images* folder of the TensorFlow directory.

- Set up the environment; install TensorFlow and the Tensor GPU using the *pip* command. Pip is a tool for installing and managing Python packages. Pip 19 or later is required for TensorFlow 2.0.

```

python -m pip install tensorflow
python -m pip install tensorflow-gpu

```

Install the required object detection libraries, as follows:

```

# static compiler
pip install --user Cython

```

```
# backport of the contextlib module to earlier Python versions.
```

```
pip install --user contextlib2
#python imaging library
pip install --user pillow
# For processing HTML and xml in python
pip install --user lxml
# open-source web application
pip install --user jupyter
#Plotting library for python
pip install --user matplotlib
```

Download Protocol Buffers (Protobuf). Run the following command from the *tensorflow/models/research/* folder:

```
protoc object_detection/protos/*.proto --python_out=.
```

Or you may compile each file in the *protos* folder, individually; for example:

```
protoc object_detection/protos/anchor_generator.proto --python_out=.
```

Add libraries to *PYTHONPATH*. Run the following command from *tensorflow/models/research/*:

```
export PYTHONPATH=$PYTHONPATH:`pwd`:`pwd`/slim
```

You can test the installation of the TensorFlow Object Detection API by running the following command:

```
python object_detection/builders/model_builder_test.py
```

Run the object detection tutorial using the following command from the *tensorflow/models/research/object\_detection* directory:

```
jupyter-notebook object_detection_tutorial.ipynb
```

It will run in the browser and will look like what's shown in Figure 3.

```
In [23]: 1 import numpy as np
2 import os
3 import six.moves.urllib as urllib
4 import sys
5 import tarfile
6 import tensorflow as tf
7 import zipfile
8
9 from distutils.version import StrictVersion
10 from collections import defaultdict
11 from io import StringIO
12 from matplotlib import pyplot as plt
13 from PIL import Image
14
15 # This is needed since the notebook is stored in the object_detection folder.
16 sys.path.append('.')
17 from object_detection.utils import ops as utils_ops
18
19 if StrictVersion(tf.__version__) < StrictVersion('1.12.0'):
20     raise ImportError('Please upgrade your TensorFlow installation to v1.12.*.')
21
```

Figure 3: Code of object detection tutorial

From the cell, select *Run all*, which will run and generate the output at the end of the page.

You will see the detection of all the images from the *test\_images* directory (*tensorflow/models/research/object\_detection/test\_images*). If you want to test the code with your images, just add the path to the images, to the line *TEST\_IMAGE\_PATHS*.

```
for image_path in TEST_IMAGE_PATHS:
    image = Image.open(image_path)
```

If you want to add more images, you can change the number of images in the range, as follows:

```
PATH_TO_TEST_IMAGES_DIR = 'test_images'
```

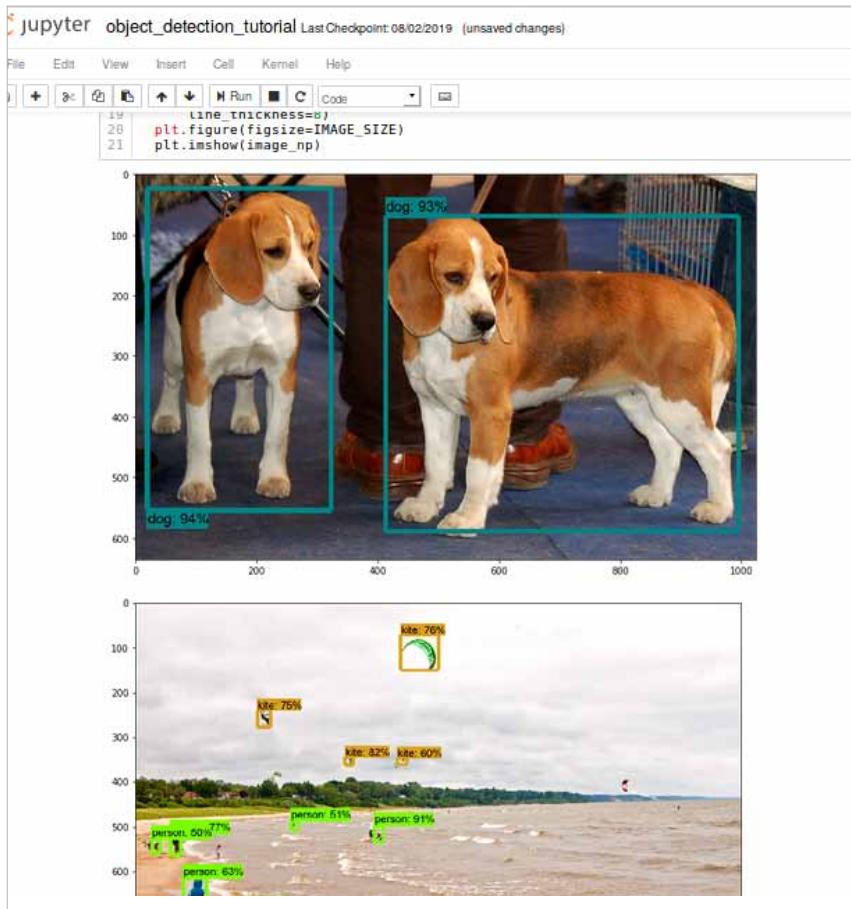


Figure 4: Object detection from an image

```
TEST_IMAGE_PATHS = [ os.path.join(PATH_TO_TEST_IMAGES_DIR, 'image{}.jpg'.format(i)) for i in range(1, 5) ]
```

```
# Size, in inches, of the output images.  
IMAGE_SIZE = (12, 8)
```

The final object identification will be shown as an output of images. The objects will be detected with labels, and the percentage(score of that object will be similar to the training set.

This very basic example can be followed for simple object identification. Moving objects/videos can also be tagged and identified in a similar manner, with a slight modification of code and addition of libraries like OpenCV, etc.

# Why the Apache Mahout Framework is So Popular

*The world we are living in today is one of information and data overload. The processing of this data can only be handled efficiently by a distributed framework like Mahout which, in conjunction with Hadoop, can be competently used for data mining. That it is used by large corporates like Facebook, Foursquare, Twitter, LinkedIn and Yahoo! is testimony to its effectiveness.*

Apache Mahout is an open source project that is used to construct scalable libraries of machine learning algorithms. Initially, it started as a child project of Apache Lucene. In 2010, it became the top-level project for Apache. There are many open source machine learning libraries available in the market but Apache Mahout stands out for various reasons:

- Many other open source libraries are only research oriented.
- There is a scarcity of resources, i.e., documentation and examples for all open source ML libraries, whereas there are ample resources available for Mahout.
- The community surpasses others.
- Apache Mahout provides greater scalability than others.

All these reasons make Apache Mahout the perfect choice for creating ML algorithms that scale.

Mahout is supported by its three pillars.

- **Recommender engines (collaborative filtering):** Recommenders can be classified as being user based or item based. Amazon and Facebook use this feature to attract users and suggest products by mining user behaviour. An example of how this feature is used is shown in Figure 1.

- **Clustering:** Clustering tries to make a cluster of things that share similarities. In simple words, it groups objects of a similar nature in one place. There are many algorithms available for clustering, like K-Means, Fuzzy K-Means, Mean Shift, Canopy, Dirichlet Classification, etc.



Figure 1: Mining user behaviour on e-commerce websites

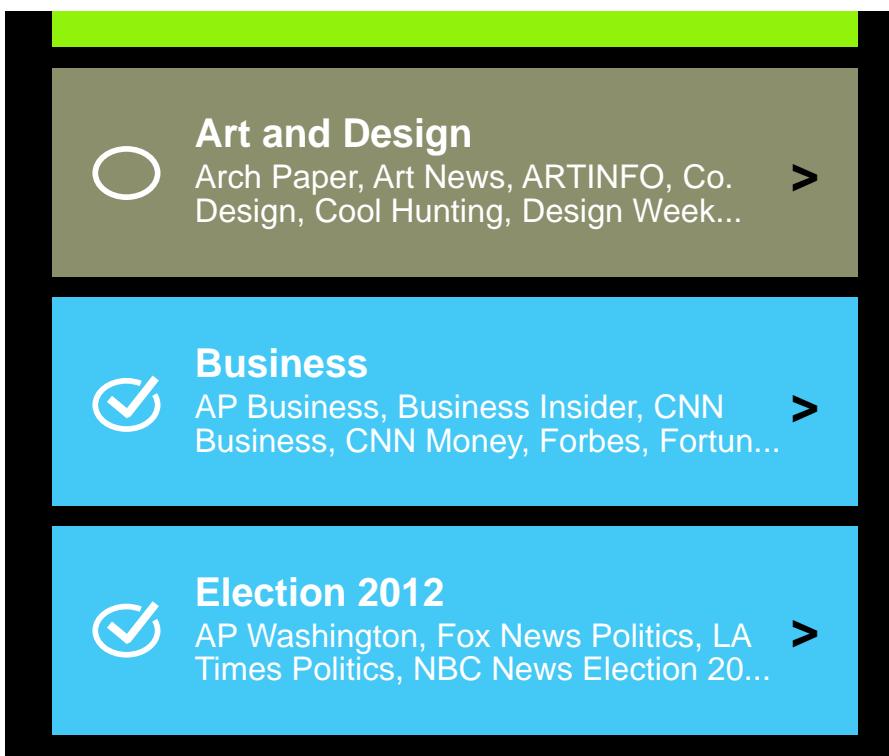


Figure 2: News in brief courtesy the Google app called Summly

A Google app called Summly shows the news from different news sites, in brief (Figure 2).

- **Classification:** Classification techniques decide whether a thing deserves to be a part of some type or not. Here, types are predetermined. Features of items in the same group are compared. Classification can predict the type of any new object based on its features. Facebook's face detection and spam checker use this technique.

## Features of the Mahout framework

- The Mahout framework is tightly coupled with Hadoop. So, it is very useful for distributed environments where Mahout uses the Apache Hadoop library to scale in the cloud.
- Developers can use Mahout for mining large volumes of data as it is a ready-to-use framework.
- Through Mahout, applications can analyse data faster and more effectively.
- MapReduce enabled clustering implementations are supported by Mahout—for example, clustering algorithms like K-Means, Fuzzy K-Means, Canopy, Dirichlet and Mean-Shift.
- It also supports distributed and complementary Naive Bayes classification implementations.
- Distributed fitness function capabilities are an inbuilt part of Apache Mahout for evolutionary programming.
- It includes vector and matrix libraries.
- It also has examples of all the above-mentioned algorithms.
- Mahout has great community support, which cannot be found for any other open source ML library.

## Applications of the Mahout framework

- IT giants like Facebook, LinkedIn, Adobe, Twitter and Yahoo! use Mahout.
- Foursquare uses the Mahout recommender engine to serve you by finding the places, entertainment options and food to your liking, in a specific area.
- User interest can be modelled, and Twitter uses Mahout for that.
- The pattern mining of the Mahout framework is used by Yahoo!

# Apache Mahout The Recommender System for Big Data

*In social networking sites and in e-commerce, recommendations play a key role. E-commerce, in particular, depends on website traffic which is driven to it by various means. Fortunately, machine learning algorithms like Recommender can be used effectively in both these cases. This article focuses on Mahout's recommender systems.*

Mahout is an open source machine learning library from the Apache Software Foundation. It implements many data mining algorithms like *recommender engines()*, *clustering ()*, *classification ()*, and is scalable to very large data sets—up to terabytes and petabytes, which is in the Big Data realm.

In this article, I will focus on recommender systems in Mahout.

Recommender engines are very popular machine learning algorithms that are used to recommend books, movies or articles based on the users' past actions and interests.

Recommender engine algorithms come in two categories – user based and item based recommendations. User based recommendations predict what the users will like, based on their similarity with other users and do not require the properties or description of the items; for example, Facebook's friend recommendations. In item based recommendations, the preferences of users for some items based on their preference of other similar items can be suggested, and this requires prior knowledge of the particular item's properties. For example, if a user likes one kind of comedy movie, he may like other similar comedy movies.

The following are the main components of the recommendation system in Mahout.

- *Data model interface*: This converts the raw data to a Mahout compliant data format. Data sources like MySQL, PostgreSQL, MongoDB, Cassandra, flat files, etc, are supported.
- *User similarity interface*: This contains methods to compute the similarity among users.
- *Item similarity interface*: This contains methods to compute the similarity between items. Many similarity metrics are available in Mahout like Pearson Correlation, Euclidean distance, etc.
- *User neighbourhood interface*: This is a method to construct a neighbourhood around a given user, which satisfies the similarity or 'nearest neighbour' threshold criteria.
- *Recommender interface*: This is a method that finally makes the actual recommendation.

In order to process Big Data, Mahout can use HDFS. So a prerequisite to installing Mahout is JDK/JAVA 1.7 or later, Maven 3.0 or later and the Hadoop cluster. If you are using Ubuntu, you can use the following commands.

Install Maven from the repository *sudo apt-get install maven*. Download the latest distribution of Mahout from the site <http://www.apache.org/dyn/closer.cgi/lucene/mahout/>. Unzip and copy the following command to the desired location:

```
cp -R SOURCE DESTINATION
```

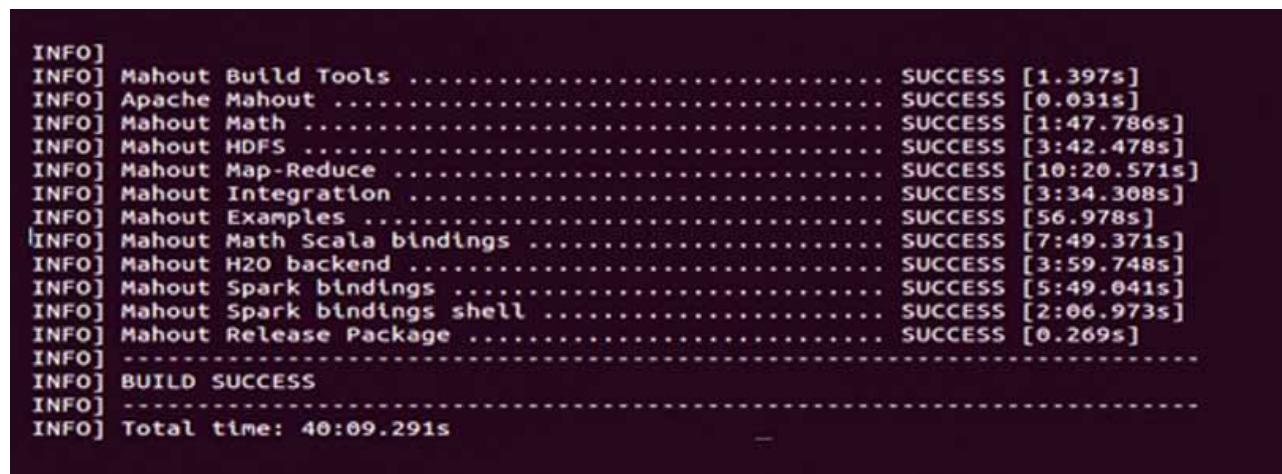
I copied it in */usr/local*.

Now, *cd /usr/local/mahout/distribution* or wherever you have copied Mahout. Then run the following command:

```
sudo mvn install
```

(Install Maven 3.0.1 or above for Mahout .20 distribution; else, it will throw

some error.) You will get the screen shown in Figure 1 after successful installation.

A terminal window displaying the build log for Apache Mahout. The log shows various build steps for different components like Build Tools, Apache Mahout, Mahout Math, Mahout HDFS, Mahout Map-Reduce, Mahout Integration, Mahout Examples, Mahout Math Scala bindings, Mahout H2O backend, Mahout Spark bindings, Mahout Spark bindings shell, and Mahout Release Package. Each step is followed by a SUCCESS status and a duration in brackets. The log concludes with a BUILD SUCCESS message and a total time taken of 40:09.291s.

```
INFO] INFO] Mahout Build Tools ..... SUCCESS [1.397s]
INFO] INFO] Apache Mahout ..... SUCCESS [0.031s]
INFO] INFO] Mahout Math ..... SUCCESS [1:47.786s]
INFO] INFO] Mahout HDFS ..... SUCCESS [3:42.478s]
INFO] INFO] Mahout Map-Reduce ..... SUCCESS [10:20.571s]
INFO] INFO] Mahout Integration ..... SUCCESS [3:34.308s]
INFO] INFO] Mahout Examples ..... SUCCESS [56.978s]
INFO] INFO] Mahout Math Scala bindings ..... SUCCESS [7:49.371s]
INFO] INFO] Mahout H2O backend ..... SUCCESS [3:59.748s]
INFO] INFO] Mahout Spark bindings ..... SUCCESS [5:49.041s]
INFO] INFO] Mahout Spark bindings shell ..... SUCCESS [2:06.973s]
INFO] INFO] Mahout Release Package ..... SUCCESS [0.269s]
INFO] -----
INFO] BUILD SUCCESS
INFO] -----
INFO] Total time: 40:09.291s
```

Figure 1: Building Apache Mahout on Ubuntu

Mahout's recommenders expect interactions between users and items as input. I tested a sample 568.3MB data, which contained the fields *userID*, *movieID* and *value*. Here, *userID* and *movie ID (itemID)* refer to a particular user and a particular item, and *value* denotes the strength of the interaction (e.g., the rating given to a movie).

The following steps can be used to run the Recommendation algorithm. Create a directory in the Hadoop file system to store the ratings file using the following command:

```
hadoop fs -mkdir /mahout_data/
```

Copy the downloaded file to HDFS using the following command:

```
hadoop fs -put /home/hduser/mydata/ml-latest/ratings.csv /mahout_data/
```

Go to the Mahout directory, `cd /usr/local/mahout/bin/` and issue the following command (the output file should be unique and `JAVA_HOME` should be properly set):

```
./mahout recommenditembased -s SIMILARITY_LOGLIKELIHOOD -i hdfs://localhost:9000/mahout_data/ratings.csv -o hdfs://localhost:9000/ratings_test/ --numRecommendations 25
-i hdfs://localhost:9000/mahout_data/ratings.csv - Denotes the input file
-o hdfs://localhost:9000/ratings_test/ -denotes the output file .
```

The command will run for a couple of minutes and you can see your output from the Web interface as well, as shown in Figure 2.



**Note:** In the above snippet, recommenditembased means we are creating an item based recommendation and not a user based recommendation. The difference between the two is that a user based recommendation finds similar users based on what they like, and item based recommendation figures out what the user likes and finds items to match those preferences. Mahout's item-based recommendation algorithm takes customer preferences by item as input, and generates an output recommending similar items with a score indicating whether a customer will like the recommended item.

You can check the output file and it will contain two columns: the *userID* and an array of *itemIDs* and scores. This could, for instance, recommend a user's preference for a particular movie, which he may be interested in watching.

Browse Directory						
	/					
Permission	Owner	Group	Size	Replication	Block Size	Name
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	clustered_data
-rW-r--r--	hduser	supergroup	6.12 KB	1	128 MB	donut.csv
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	mahout_data
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	mahout_seq
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	ratings_test
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	teragen_output
drwx-----	hduser	supergroup	0 B	0	0 B	tmp
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	user

Figure 2: Browsing directory

"We are leaving the age of information and entering the age of recommendation," says Chris Anderson, British entrepreneur and the curator of TED. The increasing adoption of the Web as a vehicle for business has changed the way in which businesses interact with the customer. Marketing teams and businesses are now using intelligent algorithms and social technologies to form meaningful, ongoing relationships with customers. It is here that technologies like Mahout Recommender will play a key role.

# Implementing Scalable and High Performance Machine Learning Algorithms Using Apache Mahout

*Apache Mahout aims at building an environment for quickly creating scalable and performant machine learning applications.*

Machine learning refers to the intelligent and dynamic response of software or embedded hardware programs to input data. Machine learning is the specialised domain that operates in association with artificial intelligence to make strong predictions and analyses. Using this approach, there is no need to explicitly program computers for specific applications; rather, the computing modules evaluate the data set with their inherent reactions so that real-time fuzzy based analysis can be done. The programs developed with machine learning paradigms focus on the dynamic input and data set, so that the custom and related output can be presented to the end user.

There are a number of applications for which machine learning approaches are widely used. These include fingerprint analysis, multi-dimensional biometric evaluation, image forensics, pattern recognition, criminal investigation, bioinformatics, biomedical informatics, computer vision, customer relationship management, data mining, email filtering, natural language processing, automatic summarisation, and automatic taxonomy construction. Machine learning also applies to robotics, dialogue systems, grammar checkers, language recognition, handwriting recognition, optical character recognition, speech recognition, machine translation, question answering, speech synthesis, text simplification, pattern recognition, facial recognition systems, image

recognition, search engine analytics, recommendation systems, etc. There are a number of approaches to machine learning, though traditionally, supervised and unsupervised learning are the models widely used. In supervised learning, the program is trained with a specific type of data set with the target value. After learning and deep evaluation of the input data and the corresponding target, the machine starts making predictions. The common examples of supervised learning algorithms include artificial neural networks, support vector machines and classifiers. In the case of unsupervised learning, the target is not assigned with the input data. In this approach, dynamic evaluation of data is done with high performance algorithms, including k-means, self-organising maps (SOM) and clustering techniques. Other prominent approaches and algorithms associated with machine learning include dimensionality reduction, the decision tree algorithm, ensemble learning, the regularisation algorithm, supervised learning, artificial neural networks, and deep learning. Besides these, there are also the instance-based algorithms, regression analyses, classifiers, Bayesian statistics, linear classifiers, unsupervised learning, association rule learning, hierarchical clustering, deep cluster evaluation, anomaly detection, semi-supervised learning, reinforcement learning and many others.

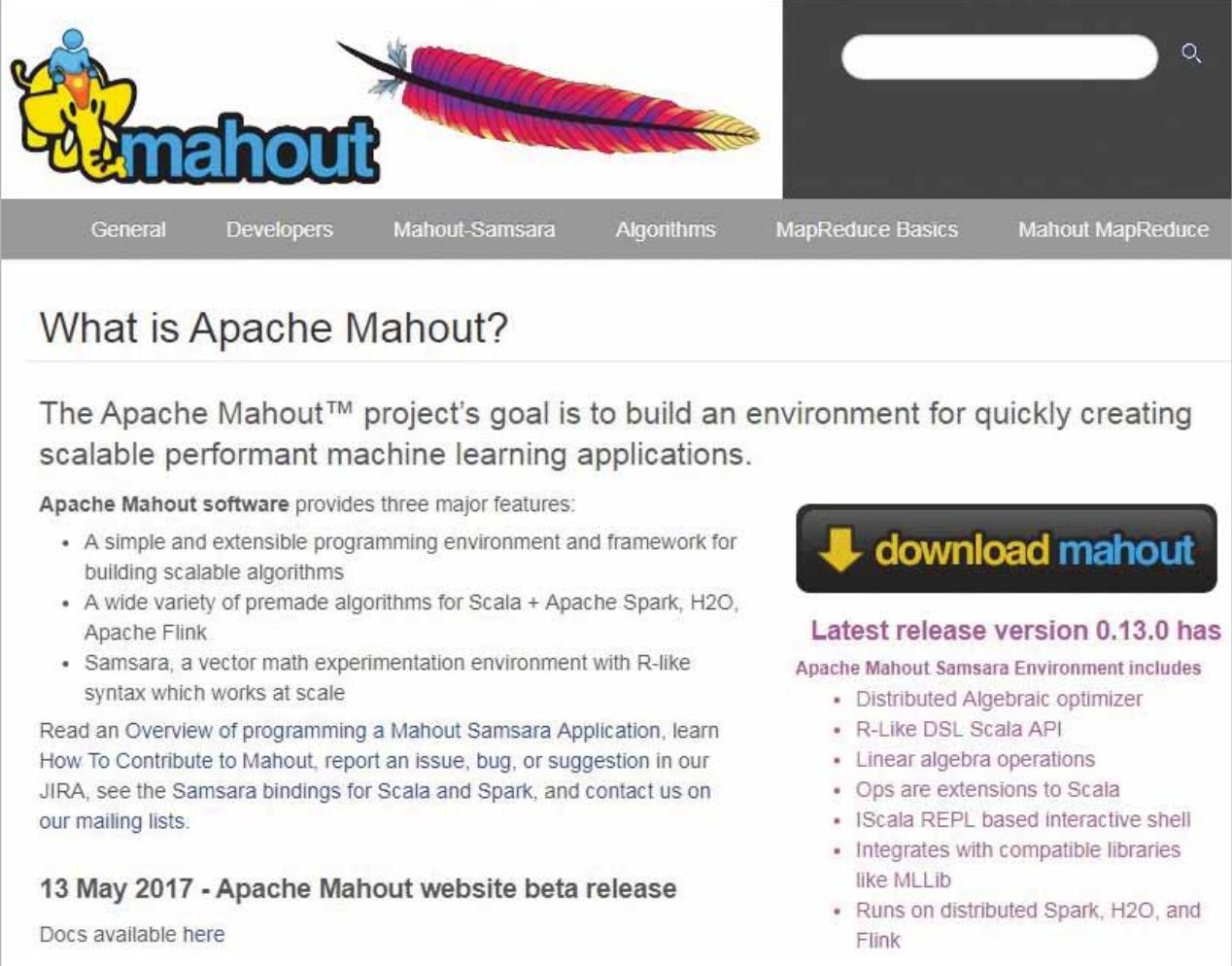
Free and open source tools for machine learning are Apache Mahout, Scikit-Learn, OpenAI, TensorFlow, Char-RNN, PaddlePaddle, CNTX, Apache Singa, DeepLearning4J, H2O, etc.

## **Apache Mahout, a scalable high performance machine learning framework**

Apache Mahout (*mahout.apache.org*) is a powerful and high performance machine learning framework for the implementation of machine learning algorithms. It is traditionally used to integrate supervised machine learning algorithms with the target value assigned to each input data set. Apache Mahout can be used for assorted research based applications including social media extraction and sentiment mining, user belief analytics, YouTube analytics and many related real-time applications.

In Apache Mahout, a 'mahout' refers to whatever drives or operates the elephant. The mahout acts as the master of the elephant in association with Apache Hadoop and is represented in the logo of the elephant. Apache Mahout runs with the base installation of Apache Hadoop, and then the machine learning algorithms are implemented with the features to develop and deploy scalable machine learning algorithms. The prime approaches, like recommender engines, classification problems and clustering, can be effectively solved using Mahout.

Corporate users of Mahout include Adobe, Facebook, LinkedIn, FourSquare, Twitter and Yahoo.



The screenshot shows the Apache Mahout website. At the top, there is a header with the Apache logo, the word "mahout" in blue, and a magnifying glass icon for search. Below the header is a navigation bar with links: General, Developers, Mahout-Samsara, Algorithms, MapReduce Basics, and Mahout MapReduce. The main content area has a title "What is Apache Mahout?". Below the title, a paragraph states: "The Apache Mahout™ project's goal is to build an environment for quickly creating scalable performant machine learning applications." To the right of this text is a "download mahout" button with a yellow arrow pointing down. Further down, a section titled "Latest release version 0.13.0 has" lists "Apache Mahout Samsara Environment includes" with a bulleted list of features: Distributed Algebraic optimizer, R-Like DSL Scala API, Linear algebra operations, Ops are extensions to Scala, IScala REPL based interactive shell, Integrates with compatible libraries like MLLib, and Runs on distributed Spark, H2O, and Flink. At the bottom left, there is a link "13 May 2017 - Apache Mahout website beta release" and a link "Docs available here".

Figure 1: The official portal of Apache Mahout

## Installing Apache Mahout

To start with the Mahout installation, Apache Hadoop has to be set up on a Linux distribution. To get ready with Hadoop, the installation is required to be updated as follows, in Ubuntu Linux:

```
$ sudo apt-get update  
$ sudo addgroup hadoop  
$ sudo adduser --ingroup hadoop hadoopuser1  
$ sudo adduser hadoopuser1 sudo  
$ sudo apt-get install ssh  
$ su hadoopuser1  
$ ssh-keygen -t rsa  
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys  
$ chmod 0600 ~/.ssh/authorized_keys  
$ ssh localhost
```

## Installing the latest version of Hadoop

Use the following code to install the latest version of Hadoop:

```
$ wget http://www-us.apache.org/dist/hadoop/common/hadoop-HadoopVersion/hadoop-  
HadoopVersion.tar.gz  
$ tar xvzf hadoop-HadoopVersion.tar.gz  
$ sudo mkdir -p /usr/local/hadoop  
$ cd hadoop-HadoopVersion/  
$ sudo mv * /usr/local/hadoop  
$ sudo chown -R hadoopuser1:hadoop /usr/local/hadoop  
$ hadoop namenode -format  
$ cd /usr/local/hadoop/sbin  
$ start-all.sh
```

The following files are required to be updated next:

- *~/.bashrc*
- *core-site.xml*

- *hadoop-env.sh*
- *hdfs-site.xml*
- *mapred-site.xml*
- *yarn-site.xml*

## Web interfaces of Hadoop

Listed below are some of the Web interfaces of Hadoop.

MapReduce: *http://localhost:8042/*

NameNode daemon: *http://localhost:50070/*

Resource Manager: *http://localhost:8088/*

SecondaryNameNode: *http://localhost:50090/status.html*

The default port to access Hadoop is 50070 and *http://localhost:50070/* is used on a Web browser.

After installing Hadoop, the setting up of Mahout requires the following code:

```
$ wget http://mirror.nexcess.net/apache/mahout/0.9/mahout-Distribution.tar.gz
$ tar zxvf mahout-Distribution.tar.gz
```



**Simple Logging Facade for Java (SLF4J)**

The Simple Logging Facade for Java (SLF4J) serves as a simple facade or abstraction for various logging frameworks (e.g. `java.util.logging`, logback, log4j) allowing the end user to plug in the desired logging framework at deployment time.

Before you start using SLF4J, we highly recommend that you read the two-page [SLF4J user manual](#).

Note that SLF4J-enabling your library implies the addition of only a single mandatory dependency, namely `slf4j-api.jar`. If no binding is found on the class path, then SLF4J will default to a no-operation implementation.

In case you wish to migrate your Java source files to SLF4J, consider our [migrator tool](#) which can migrate your project to use the SLF4J API in just a few minutes.

In case an externally-maintained component you depend on uses a logging API other than SLF4J, such as commons-logging, log4j or java.util.logging, have a look at SLF4J's binary-support for legacy APIs.

Copyright © 2004-2017 QOS.ch  
We are actively looking for volunteers to proofread the documentation. Please send your corrections or suggestions for improvement to "corrections@qos.ch". See also the instructions for contributors.

Figure 2: Simple Logging Facade for Java

## Implementing the recommender engine algorithm

Nowadays, when we shop at online platforms like Amazon, eBay, SnapDeal, FlipKart and many others, we notice that most of these online shopping platforms give us suggestions or recommendations about the products that we like or had purchased earlier. This type of implementation or suggestive modelling is known as a recommender engine or recommendation system. Even on YouTube, we get a number of suggestions related to videos that we viewed earlier. Such online platforms integrate the approaches of recommendation engines, as a result of which the related best fit or most viewed items are presented to the user as recommendations.



Figure 3: Stable JAR files from SLF4J portal

Apache Mahout provides the platform to program and implement recommender systems. For example, the Twitter hashtag popularity can be evaluated and ranked based on the visitor count, popularity or simply the hits by the users. In YouTube, the number of viewers is the key value that determines the actual popularity of that particular video. Such algorithms can be implemented using Apache Mahout, which are covered under high performance real-time machine learning.

For example, a data table that presents the popularity of products after online shopping by consumers is recorded by the companies, so that the overall analysis of the popularity of these products can be done. The user ratings from 0-5 are logged so that the overall preference for the product can be evaluated. This data set can be evaluated using Apache Mahout in Eclipse IDE.

To integrate Java Code with Apache Mahout Libraries on Eclipse IDE, there are specific JAR files that are required to be added from Simple Logging Facade for Java (SLF4J).

The following is the Java Code module, with methods that can be executed using Eclipse IDE with the JAR files of Mahout to implement the recommender algorithm:

```
DataModel dm = new FileDataModel(new File("inputdata"));
UserSimilarity us = new PearsonCorrelationSimilarity(dm);
UserNeighborhood un = new ThresholdUserNeighborhood(ThresholdValue), us, dm);
UserBasedRecommender r=new GenericUserBasedRecommender(dm, un, us);
List<RecommendedItem> rs=recommender.recommend(userID, Recommendations);
for (RecommendedItem rc : rs) {
System.out.println(rc);
```

## **Apache Mahout and R&D**

Research problems can be solved effectively using Apache Mahout with customised algorithms for multiple applications including malware predictive analytics, user sentiment mining, rainfall predictions, network forensics and network routing with deep analytics. Nowadays, the integration of deep learning approaches can be embedded in the existing algorithms so that a higher degree of accuracy and optimisation can be achieved in the results.

# **Apache SystemML: A Machine Learning Platform Suited for Big Data**

*Apache SystemML is an important machine learning platform that focuses on Big Data, with scalability and flexibility as its strong points. Its unique characteristics include algorithm customisation, multiple execution modes and automatic optimisation. This article introduces readers to the core features of Apache SystemML.*

Machine learning (ML) has applications across various domains, and has transformed the manner in which these are built. The traditional sequential algorithmic approaches are now getting replaced with learning based dynamic algorithms. ML's most important benefit is its ability to handle novel scenarios.

Machine learning research can be divided into two parts — one is the development of the underlying ML algorithms, which requires a detailed understanding of core mathematical concepts. The other part is the application of machine learning algorithms, which doesn't require the developer to know the underlying mathematics down to the smallest detail. The second part, i.e., the application of ML, involves people from various domains. For example, ML is now applied in bio-informatics, economics, earth sciences, etc.

Another positive change in the ML space is the creation of various frameworks and libraries by many leading IT majors. These frameworks have made both the development and the application of ML easier and more efficient. As of 2019, developers no longer need to burden themselves with the implementation of core components. Most of these components are available as off-the-shelf solutions.

This article explores an important machine learning platform from Apache called SystemML, which focuses on Big Data. The sheer volume and velocity of Big Data poses the challenge of scalability. One of the important advantages of

Apache SystemML is its ability to handle these scalability problems. The other distinguishing features of Apache SystemML (Figure 2) are:

- The ability to customise algorithms with the help of R-like and Python-like programming languages.
- The ability to work in multiple execution modes, which incorporate Spark MLContext, Spark Batch, etc.
- The ability to do optimisation automatically, which is based on the characteristics of both the data and cluster.

Apache SystemML has various components, all of which cannot be covered in this article. Here, we provide an introduction to the core features of Apache SystemML.

## Installation

The pre-requisite for the installation of Apache SystemML is Apache Spark. The variable *SPARK\_HOME* should be set to the location where Spark is installed. Installing Apache SystemML for the Python environment can be done with the Pip command as shown below:

```
pip install systemml
```

More information about this can be accessed at <http://systemml.apache.org/docs/1.2.0/index.html>.

If you want to work with the Jupyter Notebook, the configuration can be done as follows:

```
PYSPARK_DRIVER_PYTHON=jupyter PYSPARK_DRIVER_PYTHON_OPTS="notebook" pyspark  
-master local[*] -conf "spark.driver.memory=12g" -conf spark.driver.  
maxResultSize=0 -conf spark.default.parallelism=100
```

Instructions for installing SystemML with Scala can be got from the official documentation at <http://systemml.apache.org/install-systemml.html>.

## DML and PyDML

As stated earlier, flexibility is another advantage of SystemML, which it achieves through a high level declarative machine learning language. This language ships in two flavours—one is called DML and has syntax like R. The other is PyDML, which is like Python.

A code snippet of PyDML is shown below:

```
aFloat = 3.0
bInt = 2
print('aFloat = ' + aFloat)
print('bInt = ' + bInt)
print('aFloat + bInt = ' + (aFloat + bInt))
print('bInt ** 3 = ' + (bInt ** 3))
print('aFloat ** 2 = ' + (aFloat ** 2))

cBool = True
print('cBool = ' + cBool)
print('(2 < 1) = ' + (2 < 1))

dStr = 'Open Source'
eStr = dStr + ' For You'
print('dStr = ' + dStr)
print('eStr = ' + eStr)
```

A sample code snippet of DML is shown below:

```
aDouble = 3.0
bInteger = 2
print('aDouble = ' + aDouble)
print('bInteger = ' + bInteger)
print('aDouble + bInteger = ' + (aDouble + bInteger))
print('bInteger ^ 3 = ' + (bInteger ^ 3))
print('aDouble ^ 2 = ' + (aDouble ^ 2))
```

```
cBoolean = TRUE  
print('cBoolean = ' + cBoolean)  
print('(2 < 1) = ' + (2 < 1))
```

```
dString = 'Open Source'  
eString = dString + ' For You'  
print('dString = ' + dString)  
print('eString = ' + eString)
```

A basic matrix operation with PyDML is shown below:

```
A = full("1 2 3 4 5 6", rows=3, cols=2)  
print(toString(A))
```

```
B = A + 4  
B = transpose(B)  
print(toString(B))
```

```
C = dot(A, B)  
print(toString(C))
```

```
D = full(5, rows=nrow(C), cols=ncol(C))  
D = (C - D) / 2  
print(toString(D))
```

A detailed reference to PyDML and DML is available in the official documentation at <https://apache.github.io/systemml/dml-language-reference.html>.

For the benefit of Python users, SystemML has several language-level APIs, which enable you to use it without needing to know DML or PyDML.

```
import systemml as sml
```

```
import numpy as np
m1 = sml.matrix(np.ones((3,3)) + 2)
m2 = sml.matrix(np.ones((3,3)) + 3)
m2 = m1 * (m2 + m1)
m4 = 1.0 - m2
m4.sum(axis=1).toNumPy()
```

## Calling SystemML algorithms

SystemML has a sub-package called `mllearn`, which enables Python users to call SystemML algorithms. This is done with Scikit-learn or the MLPipeline API. A sample code snippet for linear regression is shown below:

```
import numpy as np
from sklearn import datasets
from systemml.mllearn import LinearRegression

#1 Load the diabetes dataset
diabetes = datasets.load_diabetes()

# 2 Use only one feature
diabetes_X = diabetes.data[:, np.newaxis, 2]

#3 Split the data into training/testing sets
X_train = diabetes_X[:-20]
X_test = diabetes_X[-20:]

#4 Split the targets into training/testing sets
y_train = diabetes.target[:-20]
y_test = diabetes.target[-20:]

#5 Create linear regression object
regr = LinearRegression(spark, fit_intercept=True, C=float("inf"),
solver='direct-solve')
```

```
#6 Train the model using the training sets  
regr.fit(X_train, y_train)  
y_predicted = regr.predict(X_test)  
print('Residual sum of squares: %.2f' % np.mean((y_predicted - y_test) ** 2))
```

The output of the above code is shown below:

```
Residual sum of squares: 6991.17
```

A sample code snippet with the MLPipeline interface and the logistic regression is shown below:

```
# ML Pipeline way  
from pyspark.ml import Pipeline  
from systemml.mllearn import LogisticRegression  
from pyspark.ml.feature import HashingTF, Tokenizer  
  
training = spark.createDataFrame([  
    (0, "a b c d e spark", 1.0),  
    (1, "b d", 2.0),  
    (2, "spark f g h", 1.0),  
    (3, "hadoop mapreduce", 2.0),  
    (4, "b spark who", 1.0),  
    (5, "g d a y", 2.0),  
    (6, "spark fly", 1.0),  
    (7, "was mapreduce", 2.0),  
    (8, "e spark program", 1.0),  
    (9, "a e c l", 2.0),  
    (10, "spark compile", 1.0),  
    (11, "hadoop software", 2.0)  
], ["id", "text", "label"])  
tokenizer = Tokenizer(inputCol="text", outputCol="words")  
hashingTF = HashingTF(inputCol="words", outputCol="features", numFeatures=20)
```

```
lr = LogisticRegression(sqlCtx)
pipeline = Pipeline(stages=[tokenizer, hashingTF, lr])
model = pipeline.fit(training)
test = spark.createDataFrame([
    (12, "spark i j k"),
    (13, "l m n"),
    (14, "mapreduce spark"),
    (15, "apache hadoop")], ["id", "text"])
prediction = model.transform(test)
prediction.show()
```

## Deep learning with SystemML

Deep learning has evolved into a specialised class of machine learning algorithms, which makes handling of features simple and efficient. SystemML also has support for deep learning. There are three methods with which deep learning can be carried out in SystemML (Figure 3):

- With the help of the DML-bodied NN library. This enables the utilisation of DML to implement neural networks.
- *Caffe2DML API*: This API enables the model to be represented in Caffe's proto format.
- *Keras2DML API*: This API enables the model to be represented in Keras.

A code snippet with Keras2DML, to implement ResNet50, is shown below:

```
import os
os.environ['CUDA_DEVICE_ORDER'] = 'PCI_BUS_ID'
os.environ['CUDA_VISIBLE_DEVICES'] = ''

# Set channel first layer
from keras import backend as K
K.set_image_data_format('channels_first')
from systemml.mllearn import Keras2DML
```

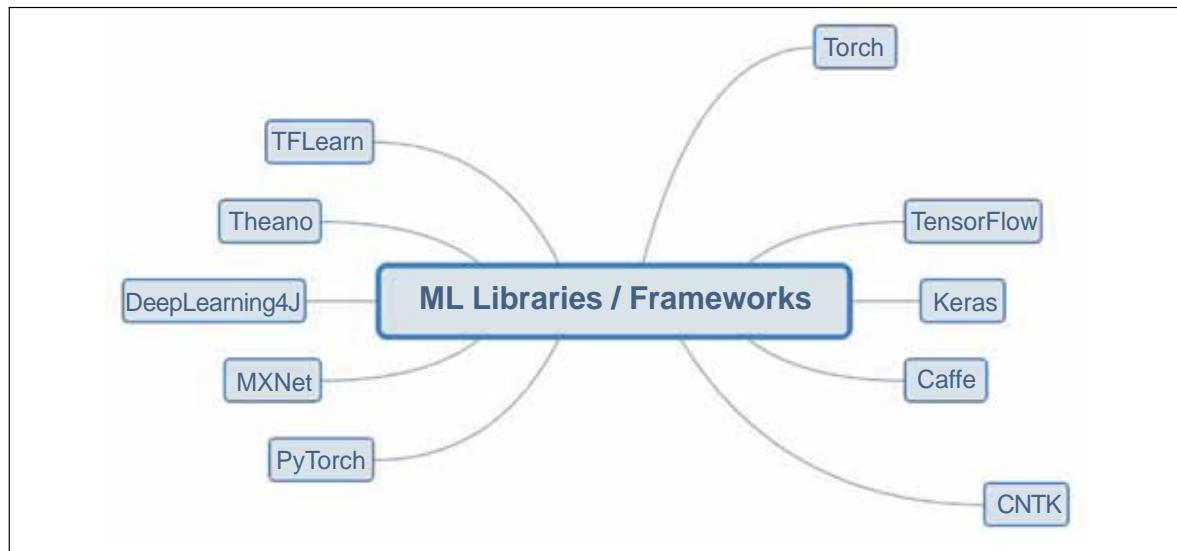


Figure 1: Machine learning frameworks/libraries

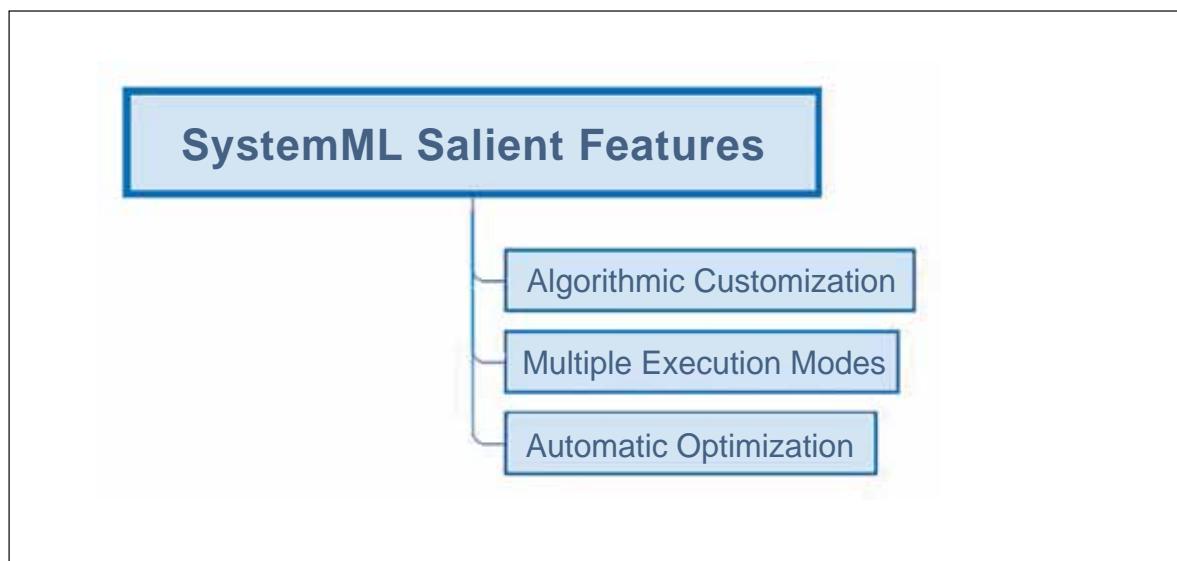


Figure 2: SystemML's salient features



Figure 3: Deep learning with SystemML

```
import systemml as sml
import keras, urllib
from PIL import Image
from keras.applications.resnet50 import preprocess_input, decode_predictions,
ResNet50

keras_model = ResNet50(weights='imagenet', include_
top=True, pooling='None', input_shape=(3,224,224))
keras_model.compile(optimizer='sgd', loss= 'categorical_crossentropy')

sysml_model = Keras2DML(spark,keras_model,input_shape=(3,224,224),
weights='weights_dir', labels='https://raw.githubusercontent.com/apache/
systemml/master/scripts/nn/examples/caffe2dml/models/imagenet/labels.txt')
sysml_model.summary()
urllib.urlretrieve('https://upload.wikimedia.org/wikipedia/commons/f/f4/Cougar_
sitting.jpg', 'test.jpg')
img_shape = (3, 224, 224)
input_image = sml.convertImageToNumPyArr(Image.open('test.jpg'), img_shape=img_
shape)
sysml_model.predict(input_image)
```

As the SystemML is still evolving, the road map for future features includes enhanced deep learning support, support for distributed GPUs, etc.

To summarise, SystemML aims to position itself as SQL for machine learning. It enables developers to implement and optimise machine learning code with ease and effectiveness. Scalability and performance are its major advantages. The ability to run on top of Spark makes automatic scaling possible. With the planned expansion of deep learning features, SystemML will become stronger in future releases. If you are a machine learning enthusiast, then SystemML is a platform that you should try.

# H2O: The Versatile Tool for Deep Learning

*The H2O project developed by H2O.ai provides users tools for data analysis, allowing them to fit thousands of potential models when trying to discover patterns in data. It is a very versatile tool since it is supported by various programming languages like R, Python and MATLAB.*

Deep learning is a superset of the artificial neural network architecture and is gradually becoming an essential tool for data analysis and prediction. Leading programming languages like R, Python and MATLAB provide powerful tools and support for the implementation of data analysis using deep learning. Among such different tools, both TensorFlow and H2O have supportive packages in R and Python. Hence, both these languages are steadily becoming indispensable for deep learning and data analysis.

## **Deep learning with H2O**

The main objective of H2O.ai is to add intelligence to business. This works on the principle of deep learning and computational artificial intelligence. H2O provides easy solutions for data analysis in the financial services, insurance and healthcare domains, and is gradually proving itself as an efficient tool for solving complex problems.

This deep learning tool follows the multi-layer feed forward neural networks of predictive models, and uses supervised learning models for regression analysis and classification tasks. To achieve process-parallelism over large volumes

of data distributed over a cluster or grid of computational nodes, it uses the MapReduce algorithm. With the help of various mathematical algorithms, MapReduce divides a task into small parts and assigns them to multiple systems. H2O is scalable from small PCs to multi-core servers and even to multi-core clusters. To prevent over-fitting, different regularisation techniques are used. Commonly used approaches are L1 and L2 (Lasso and Ridge). Other than these approaches, H2O uses the dropout, HOGWILD! and model averaging methods also. For non-linear activation functions, H2O uses the Hyperbolic tangent, Rectifier Linear and Maxout functions. The performance of each function depends on the operational scenarios and there is no one best rule to base one's selection upon. For error estimation, this model uses either one of the Mean Square Error, Absolute, Huber or Cross Entropy functions. Each of these loss functions are strongly associated with a particular data distribution function and are used accordingly.

H2O provides both manual and automatic optimisation modes for faster and a more robust convergence of network parameters to data analysis and classification problems. To reduce oscillation during the convergence of network parameters, H2O uses the Learning Rate Annealing technique to reduce the learning rate as the network model approaches its goal.

H2O performs certain essential preprocessing of data. Other than categorical encoding, it also standardises data with respect to its activation functions. This is essential, as most of the activation functions generally do not map data into the full spectrum of the real numbers scale.

## **H2O and R**

H2O supports standalone as well as R, Python and Web based interfaces. Here I shall discuss H2O in the R language platform. It is installed from the CRAN site with the *install.packages("h2o")* command from the command line. After successful installation, the package is loaded into the current workspace by the library (*h2o*) function call. Since H2O is a multi-core distributed system and can be loaded in a cluster of the system, it is invoked into the present

computation environment by the `h2o.init()` command. In this case, to initialise the package into the local machine with all its available cores, `h2o.init(nthreads = -1)` is used. The '`-1`' indicates all the cores of the local host. By default, H2O uses two cores. In case H2O is installed in a server, the `h2o.init()` function can establish a connection between the local host and the remote server by specifying the server's IP address and port number as follows:

```
h2o.init(ip="172.16.8.90", port=5453)
```

**Example:** To demonstrate the strength and perfection of the deep learning approach here, I have taken up a problem related to optical character recognition (OCR). In general, OCR software first divides an alphabetic document into a grid containing a single character (glyph) in each cell. Then it compares each glyph with a set of all the characters to recognise the character of the glyph. The characters are then combined back into words and the system performs spelling and grammar checks as final processing.

The objective of this example is to identify each of the black-and-white rectangular pixels displayed as one of the 26 capital letters in the English alphabet. The glyph images are based on 20 different fonts and each letter within these 20 fonts has been randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus has been converted into 16 primitive numerical attributes called statistical moments and edge counts, which have then been scaled to fit into a range of integer values from 0 through 15.

For this example, the first 16,000 items are taken for training of the neural model and then the trained model is used to predict the category for the remaining 4000 font-variations of the 26 letters of the English alphabet. The used data set is by W. Frey and D.J. Slat and is available from <http://archive.ics.uci.edu/ml/datasets/Letter+Recognition>. Each character is represented by a glyph and the task is to match each with one of the 26 English letters for their classification. There are 20,000 rows and 17 attributes of the character data set, as shown in Table 1.

Table 1: Attribute information

1.	letter	capital letter	(26 values from A to Z)
2.	x-box	horizontal position of box	(integer)
3.	y-box	vertical position of box	(integer)
4.	width	width of box	(integer)
5.	high	height of box	(integer)
6.	onpix	total # on pixels	(integer)
7.	x-bar	mean x of on pixels in box	(integer)
8.	y-bar	mean y of on pixels in box	(integer)
9.	x2bar	mean x variance	(integer)
10.	y2bar	mean y variance	(integer)
11.	xybar	mean x y correlation	(integer)
12.	x2ybr	mean of x * x * y	(integer)
13.	xy2br	mean of x * y * y	(integer)
14.	x-ege	mean edge count left to right	(integer)
15.	xegvy	correlation of x-ege with y	(integer)
16.	y-ege	mean edge count bottom to top (integer)	(integer)
17.	yegvx	correlation of y-ege with x	(integer)

## Classification using H2O

The 16 attributes (2<sup>nd</sup>-17<sup>th</sup> rows) as stated in the above table measure different dimensional characteristics of the glyph (1st row)—the proportions of black versus white pixels, and the average horizontal and vertical position of the pixels, etc. We have to identify the glyph on the basis of these attributes, and then classify all the similar glyphs to one of the 26 characters. To start with, first set the environment with the desired working directory and load the necessary libraries.

```
>path<- "I:\\DEEPNET"
>setwd(path)

>library(data.table)
>library(h2o)
>localH2o <- h2o.init(nthreads = -1)
```

Then download the *letters.csv* file from the above data archive. As per the requirements for H2O, the data set is then converted to the H2O data frame.

```
letterimage<- fread("letterdata.csv", stringsAsFactors = T)
letter.h2o <- as.h2o(letterimage)
```

Now it is time to set the dependent and independent variables from the letter data frame ‘letterimage’. The first column containing the English letters is the dependent variable and the rest of the columns are the independent variables.

```
#dependent variable (Letter)
>y.dep<- 1
#independent variables (dropping ID variables)
>x.indep<- c(2:ncol(letterimage))
```

To simulate the deep learning model, the H2O data frame *letter.h2o* is divided into training and test data sets. First, 16,000 rows are assigned to the training data set and the remaining 4000 records are considered as the test data set.

```
>train<- letter.h2oframe[1:16000,]
>test<- letter.h2o[16001:nrow(letter.h2oframe),]
```

Now we are ready to form the deep learning neural network model. The H2O function *h2o.deeplearning()* is used here to create a feed-forward multi-layer artificial neural network model on the training H2O data frame ‘train’.

```
>dlearning.model<- h2o.deeplearning
(
  y = y.dep,
  x = x.indep,
  training_frame = train,
  epoch = 50,
  hidden = c(100,100),
  activation = "Rectifier",
  seed = 1122
)
```

This function forms a neural model with two hidden layers with [100,100] synaptic nodes. The activation function of the model is set to the rectifier function. The initialisation of weightage and bias vectors has been done with random number sets with the seed value 1122. The model also sets the maximum number of iterative epochs over the model as 50. The training data set, along with the response and predictor variables (x, y), tunes the model as a universal classifier to identify and classify letters into their respective categories. The performance of the model can be studied with the help of the *h2o.performance()* function. To give you an idea about its performance, this function is used here over the created model itself.

```
>h2o.performance(dlearning.model)
```

This model can now be used over the test data to verify the performance of this deep neural model. The final performance study is done by comparing the outcome of the model with the actual test data. As comparison requires variables in the *R data.frame* format, both the predicted and the test data are converted into data frames.

```
>predict.dl2 <- as.data.frame(h2o.predict(dlearning.model, test))
>test.df<- as.data.frame(test)
```

A thorough look into both the data frames is helpful to gauge the correctness of the classification task. Often the verification of the performance is done by the confusion matrix. This is done by factorising both the data frames with the help of the *table* function, as shown in Table 2.

```
>table(test.df[,1],predict.dl2[,1])
```

From the confusion matrix it is apparent that though there are few false positive and false negation classifications, the overall performance is quite high. A tabular matrix display is also helpful to study the performance. This can be done from the test data frame with the help of the following command sequences:

Table 2

	A	B	C	D	E	F	G	H
A	147	0	0	1	0	0	0	....
B	0	124	0	2	0	0	0	....
C	1	0	137	0	1	0	1	....
D	0	2	0	165	0	0	0	....
E	0	0	3	0	134	2	5	....
F	0	0	0	0	0	144	0	....
...	...	...	..	...	...	...	..	..

```
>predictions <- cbind(as.data.frame(seq(1,nrow(test.df))),
test.df[,1],predict.dl2[,1])
>names(predictions) <- c("Sr Nos","Actual","Predicted")
```

```
>result <- as.matrix(predictions)
```

Sr	Nos	Actual	Predicted
[1,]	1	U	U
[2,]	2	N	N
[3,]	3	V	V
[4,]	4	I	I
[5,]	5	N	N
[6,]	6	H	H
[7,]	7	E	E
[8,]	8	Y	Y
[9,]	9	G	G
....	...	....	...

Since the number of mismatches is too low, to identify a mismatch and to study the performance it is better to use the *table()* function over the difference between the test and predicted values.

```
>performance <- test.df[,1] == predict.dl2[,1]
```

```
>table(performance)
```

```
performance
FALSE TRUE
222 3778
```

Performance shows that, out of 4000 test data cases, deep neural net failed to identify the correct letter only in 222 cases.

To compare the performance with other methods it is better to have a percentage evaluation of the classification task. Here is a simple way to do this.

```
>table(performance)*100/4000
```

```
FALSE TRUE
5.55 94.45
```

The result shows that the success rate of this experiment is 94.45 per cent. In classification exercises, it is always better to have a comparative study of methods. I have done the above classification task with Support Vector Machine to explore a better alternative and to study how good this method is for this task.

## **Using Support Vector Machine (SVM)**

SVM is also useful to classify image data. I have used the SVM model to classify the same data to compare the performance between the deep learning and SVM models. Though the performance of individual models is highly dependent on different model parameters and there is every possibility to get different results in different runs, my objective here is to gauge the performance difference between these two supervised learning schemes. Readers may explore this further by applying this experiment to other neural network models. As in the earlier case, appropriate libraries are loaded and then the training and test data are prepared.

```
>library(kernlab)#load SVM package
>letters_train<- letterimage [1:16000, ]
>letters_test<- letterimage [16001:20000, ]
```

Next, train the SVM (`ksvm`) with the letter column as the only response variable and the rest of the columns as predictors. As Radial Basis Function (RBF) works better for classification tasks in many cases, I have used the RBF as the SVM kernel function. Readers may use different functions, based on their preferences.

```
>letter_classifier_SVM<- ksvm(letter ~ ., data = letters_train, kernel =  
"rbfdot")
```

To verify this SVM model, we can use the test data to predict the outcome of this data:

```
>letter_predictions<- predict(letter_classifier_SVM, letters_test)
```

As in the earlier case, this performance can also be verified with the confusion matrix:

```
>agreement<- letter_predictions == letters_test$letter  
>table(agreement)
```

```
agreement  
FALSE TRUE  
643 3357
```

In percentage terms, this can be done in either of the following two ways.

1. By using the following command:

```
>table(agreement)*100/4000
```

```
agreement  
FALSE TRUE  
7.025 92.975
```

## 2. By using the *prop.table()* function.

From the classification results of H2O and SVM, it is apparent that both the supervised artificial neural network models methods are suitable for Optical Character Recognition classification and can be used to achieve higher performance. While H2O requires a more advanced computational platform, the requirements of SVM are less. But H2O provides a sophisticated versatile approach; so classifier models can be designed and monitored more flexibly than with the SVM approach.

# ONNX: Helping Developers Choose the Right Framework

*ONNX or Open Neural Network Exchange ([onnx.ai](https://onnx.ai)) is a community project created by Facebook and Microsoft. It is intended to provide interoperability within the AI tools community. ONNX unlocks the framework dependency for AI models by bringing in a new common representation for any model, which allows easy conversion of a model from one framework to another.*

Deep learning with neural networks is accomplished through computation over dataflow graphs. Developers use frameworks such as CNTK, Caffe2, Theano, TensorFlow, PyTorch and Chainer to represent a computational graph. All these provide interfaces that make it simple for developers to construct computation graphs and runtimes that process the graphs in an optimised way. The graph serves as an intermediate representation (IR) that captures the specific intent of the developer's source code, and is conducive for optimisation and translation to run on specific devices (CPU, GPU, FPGA, etc).

The pain point with many frameworks is that each one of them follows its own representation of a graph with similar capabilities. In simple words, a model is bound to a framework with the stack of API, graph and runtime. We can't take a model from one framework to another. Besides this, frameworks are typically optimised for certain specific characteristics, such as fast training, supporting complicated network architectures, inference on mobile devices, etc. It's up to the developer to select a framework that is optimised for one of these characteristics. Additionally, these optimisations may be better suited for particular stages of development. This leads to significant delays between research and production due to the necessity of conversion.

ONNX was launched with the goal of democratising AI, by empowering developers to select the framework that works best for their project, at any stage of development or deployment. The Open Neural Network Exchange (ONNX) format is a common IR to help establish this powerful ecosystem. By providing a common representation of the computation graph, ONNX helps developers choose the right framework for their task, allows authors to focus on innovative enhancements, and enables hardware vendors to streamline optimisations for their platforms.

ONNX provides the definition of an extensible computation graph model, as well as definitions for built-in operators. Each computation graph is structured as a list of nodes forming an acyclic graph. Here, the node is defined as a piece of a computational block with one or more inputs and one or more outputs. Each node is a call to an operation. The graph also has metadata to document or describe the model. Operators are implemented externally to the graph, but the built-in operators are portable across frameworks. Every framework supporting ONNX will provide implementations of these operators on the applicable data types.

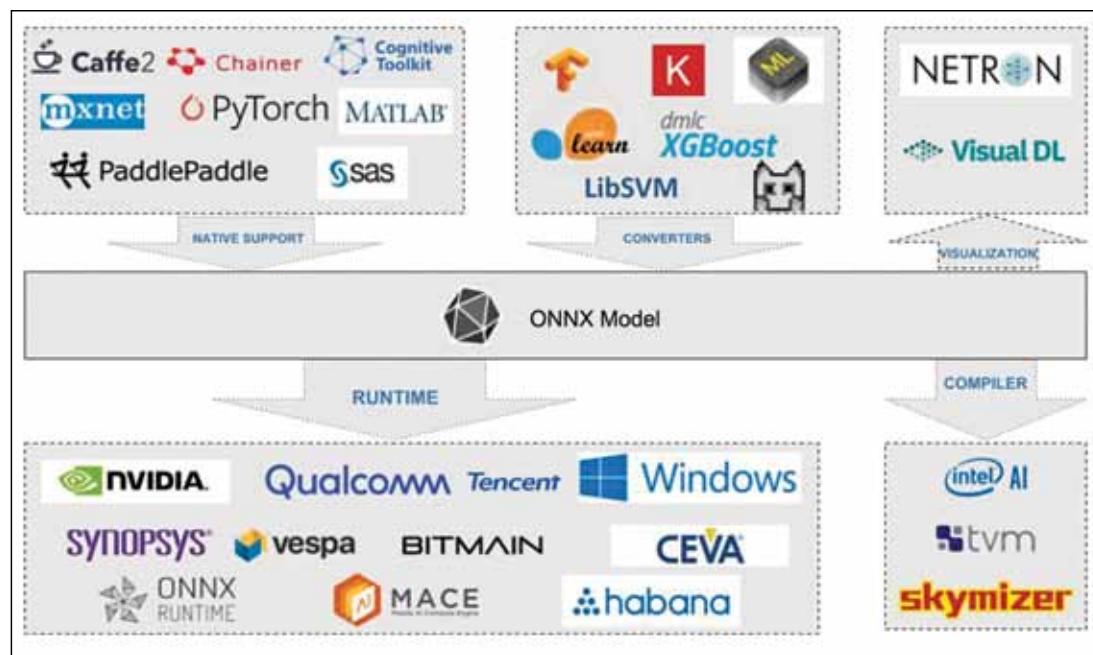


Figure 1: The ONNX ecosystem

As of today, ONNX supports various frameworks like Caffe2, Chainer, Cognitive toolkit, MxNet, PyTorch, PaddlePaddle, MATLAB and SAS. ONNX also supports various converters for TensorFlow, Keras, CoreML, Scikit Learn, XGBoost, LibSVM and NCNN(Tencent). The execution runtimes supported by ONNX are NVIDIA, Qualcomm, Bitmain, Tencent, Vespa, Windows, Synopsys, Ceva, Mace, Habana and, recently, Onnx Runtime. There are also compilers for ONNX models from Intel AI, Skymizer and TVM. NETRON and VisualDL help ONNX to visualise and manipulate graphs.

With the wide range of frameworks, converters, runtimes, compilers and visualisers, ONNX unlocks the dependencies and allows developers to make their own choices regarding frameworks and tools. ONNX has two variants. The base variant is for machine learning based on neural networks. Another variant, ONNX-ML, adds additional operators and data types for classical machine learning algorithms.

More details about ONNX Intermediate Representation (IR), operator information and graph utilities can be found at <https://github.com/onnx/onnx>.

## Setting it up

ONNX is released as a Python package and can be installed and verified. ONNX graph representation is based on protobuf; hence, protobuf is a dependency for ONNX. On Ubuntu Linux, the following command can install protobuf dependencies:

```
test@test-pc:~$ sudo apt-get install protobuf-compiler libprotobuf-dev
```

The ONNX Python package can be installed and verified from the Python package manager as shown below:

```
test@test-pc:~$ pip3 install onnx
```

```
test@test-pc:~$ python3 -c "import onnx"
```

ONNX is just a graphical representation and when it comes to executing an ONNX model, we still need a back-end. In this example, we use the TensorFlow back-end to run the ONNX model and hence install the package as shown below:

```
test@test-pc:~$ pip3 install onnx_tf
test@test-pc:~$ python3 -c "from onnx_tf.backend import prepare"
```

Now we have the setup ready to build a model in ONNX and execute it in the TensorFlow back-end.

Alternatively, ONNX is also published as Docker images for CPU and GPU environments. Those who are familiar with Docker can use the following command to set up the ONNX Docker environment:

```
docker run -it --rm onnx/onnx-docker:cpu /bin/bash
```

Use the following command for environments with GPU support. This option requires NVIDIA-Docker as a prerequisite.

```
nvidia-docker run -it --rm onnx/onnx-docker:gpu /bin/bash
```

## An example

This simple example demonstrates how to build a simple graph and execute it on the TensorFlow back-end. Copy the following snippet of code to add two tensors as *test.py*:

```
# Imports from ONNX
from onnx import helper, TensorProto
from onnx_tf.backend import prepare
import numpy as np

# Input shape and dtype
```

```
in_shape = (2, 3)
dtype = "float32"
out_shape = in_shape

# Make use of helper to construct a node to produce "out" by adding "in1" and
# "in2".
z = helper.make_node("Add", ['in1', 'in2'], ['out'])

# Make a graph that produce z from inputs named "in1" and "in2"
graph = helper.make_graph([z],
    '_test',
    inputs = [helper.make_tensor_value_info("in1",
        TensorProto.FLOAT, list(in_shape)),
        helper.make_tensor_value_info("in2",
        TensorProto.FLOAT, list(in_shape))],
    outputs = [helper.make_tensor_value_info("out",
        TensorProto.FLOAT, list(out_shape))])

# Now make an ONNX model from graph.
model = helper.make_model(graph, producer_name='_test')

# Input data for graph inputs.
x = np.array([[1.0, 2.0, 3.0], [4.0, 5.0, 6.0]])
y = np.array([[3.0, 3.0, 3.0], [3.0, 3.0, 3.0]])

# Use the backend and prepare the ONNX model for execution.
tf_rep = prepare(model)

# Run the tensorflow representation with inputs.
output = tf_rep.run([x, y])

# Output
```

```
print("Output:", output)
```

Run the above sample code; it produces the following output, which is a result of adding two input tensors.

```
test@test-pc:~$ python3 test.py
Output: Outputs(out=array([[4., 5., 6.],
                           [7., 8., 9.]], dtype=float32))
```

In the above example, we built a simple graph by constructing ONNX nodes. This example uses the TensorFlow back-end for execution. ONNX offers a wide range of back-end support to execute the same ONNX model. The following section explains how to use a converter to import a pretrained graph from a different framework and then execute it.

## Using a converter

As its name suggests, ONNX is aimed at exchanging AI models across frameworks. ONNX has a project called 'onnxmltools' (<https://github.com/onnx/onnxmltools>) which has converters for various frameworks like CoreML, Scikit Learn, Keras, SparkML, LightBGM, LibSVM and XGBoost. For TensorFlow models, the project is 'tensorflow-onnx' (<https://github.com/onnx/tensorflow-onnx>), which provides the converter utility.

These converters basically enable converting an AI model from one framework to another. Let's look at the example of importing a pretrained TensorFlow model into ONNX, and then infer and save it in the ONNX format.

Download and extract the Mobilenet pretrained model from [http://download.tensorflow.org/models/mobilenet\\_v1\\_2018\\_08\\_02/mobilenet\\_v1\\_1.0\\_224.tgz](http://download.tensorflow.org/models/mobilenet_v1_2018_08_02/mobilenet_v1_1.0_224.tgz). Extracting this will have a TensorFlow protobuf as shown below:

```
test@test-pc:~$ ls mobilenet_v1_1.0_224
Mobilenet_v1_1.0_224.ckpt.data-00000-of-00001
```

```
Mobilenet_v1_1.0_224.ckpt.meta  
Mobilenet_v1_1.0_224_frozen.pb  
Mobilenet_v1_1.0_224.ckpt.index
```

Use the following sample and load the TensorFlow model, convert to an ONNX model and infer using the ONNX Runtime back-end.

```
# Tensorflow imports  
import tensorflow as tf  
  
# Numpy  
import numpy as np  
  
#Onnx  
from onnx import load  
  
# Tensorflow to ONNX converter  
import tf2onnx  
  
# OnnxRuntime backend  
import onnxmlite.backend as backend  
  
# Import the Tensorflow protobuf into session  
tf.reset_default_graph()  
with tf.gfile.FastGFile("./mobilenet_v1_1.0_224_frozen.pb", 'rb') as f:  
    graph_def = tf.GraphDef()  
    graph_def.ParseFromString(f.read())  
    graph = tf.import_graph_def(graph_def, name='')  
  
with tf.Session() as sess:  
    # Use tf2onnx converter and build onnx_graph  
    onnx_graph = tf2onnx.tfonnx.process_tf_graph(sess.graph, input_
```

```
names=[“input:0”], output_names=[“MobilenetV1/Predictions/Reshape_1:0”])

# Make ONNX model from graph
model_proto = onnx_graph.make_model(“test”)

# Optionally the ONNX model can be saved as a serialized protobuf.
with open(“mobilenet_v1.onnx”, “wb”) as f:
    f.write(model_proto.SerializeToString())

# A ONNX saved model can be loaded by load utility.
model_new = load(“mobilenet_v1.onnx”)

# Prepare some random test data
in_shape = (1, 224, 224 , 3)
data = np.random.uniform(size=in_shape).astype(‘float32’)

# We can now use any ONNX backend to execute the graph
# Here we use onnxruntime backend.

# Prepare backend session with ONNX model

sess = backend.prepare(model_new, ‘CPU’)

# Execute on backend
output = sess.run(data)

# Output
print(“Output:”, output)
```

The above example uses the *tf2onnx* package, which is a converter for TensorFlow to ONNX. For many frameworks, these converters are natively available, and for others there is an exclusive tool for conversion. A complete list of converters available for various frameworks is shown in Figure 2.

Framework / tool	Installation	Exporting to ONNX (frontend)	Importing ONNX models (backend)
Caffe	apple/coremltools and onnx/onnxmлltools	Exporting	n/a
Caffe2	part of caffe2 package	Exporting	Importing
Chainer	chainer/onnx-chainer	Exporting	coming soon
Cognitive Toolkit (CNTK)	built-in	Exporting	Importing
Apple CoreML	onnx/onnx-coreml and onnx/onnxmлltools	Exporting	Importing
Keras	onnx/keras-onnx	Exporting	n/a
LibSVM	onnx/onnxmлltools	Exporting	n/a
LightGBM	onnx/onnxmлltools	Exporting	n/a
MATLAB	onnx converter on matlab central file exchange	Exporting	Importing
Menoh	pfnet-research/menoh	n/a	Importing
ML.NET	built-in	Exporting	Importing
Apache MXNet	part of mxnet package docs github	Exporting	Importing
PyTorch	part of pytorch package	Exporting, Extending support	coming soon
SciKit-Learn	onnx/sklearn-onnx	Exporting	n/a
TensorFlow	onnx/onnx-tensorflow and onnx/tensorflow-onnx	Exporting - ONNX-Tensorflow Exporting - Tensorflow-ONNX	Importing [experimental]
TensorRT	onnx/onnx-tensorrt	n/a	Importing

Figure 2: ONNX converter support (<https://github.com/onnx/tutorials>)

## ONNX Model Zoo

To enable developers to start using ONNX, the community has hosted Model Zoo (<https://github.com/onnx/models>), which is a collection of pretrained state-of-art models in the ONNX format for image classification, object detection, image segmentation, gesture analysis, image manipulation, speech and audio processing, machine translation, language modelling and the visual question answering dialogue. A wide variety of Jupyter notebooks is available under this project, which helps developers to start with ease.

## Deployment

ONNX deployment is possible in various targets starting from Android and iOS devices to AWS Lambda, Amazon SageMaker and Azure back-ends. A range of tutorials demonstrating various deployment scenarios can be found at <https://github.com/onnx/tutorials#end-to-end-tutorials>.

## ONNX Runtime

ONNX Runtime is a new initiative from Microsoft towards ONNX's very own deployment runtime environment for ONNX models. It supports CUDA, MLAS (Microsoft Linear Algebra Subprograms), MKL-DNN and MKL-ML for computation acceleration. There is ongoing collaboration to support Intel MKL-DNN, nGraph and NVIDIA TensorRT. ONNX Runtime supports Python, C#, C and C++ API on Windows, Linux and Mac operating systems.

ONNX Runtime is released as a Python package in two versions—*onnxruntime* is a CPU target release and *onnxruntime-gpu* has been released to support GPUs like NVIDIA CUDA. With hardware acceleration and dedicated runtime for ONNX graph representation, this runtime is a value addition to ONNX.

## Contributors

ONNX is licensed under MIT. It is supported by a wide range of community members from across academic institutions like the Technical University of Munich and MIT, from Facebook and Microsoft, along with many freelancers and anonymous contributors.

# Weka: A Free and Open Source Suite for Machine Learning and Deep Learning Algorithms

*Weka or the Waikato Environment for Knowledge Analysis is a machine learning suite that is written in Java, having been developed by the University of Waikato in New Zealand. This article is an introduction to an effective ML tool.*

Machine learning, deep learning and predictive analytics are the key domains of research in engineering, finance, economics, real-time imaging and many other fields. Researchers are working on different tools and technologies in these fields so that a higher degree of accuracy can be achieved.

The deep learning industry is very closely integrated with machine learning (ML), resulting in a higher degree of performance and accuracy with the minimum error rate. Figure 1 gives the predicted CAGR in revenues from deep learning in the USA, during the period 2014 to 2025.

Weka is a free and open source tool for machine learning and Big Data analytics (URL: <https://www.cs.waikato.ac.nz/ml/weka/>). Table 1 lists the prominent tools and software libraries used for the machine learning and data science based implementations.

Although there are a number of software libraries being widely used, Weka is a powerful tool preferred by researchers and data scientists. It has a huge set of machine learning and data science based algorithms including Big Data analytics. Weka can be used with the command line interface as well as the graphical user interface (GUI) for the implementation of algorithms. Besides the inbuilt and pre-loaded packages in Weka, there are assorted extension packages that can be integrated for advanced applications.

## Predictions using machine learning algorithms in Weka

Let us look at an example of a classifier based machine learning algorithm being used in Weka. For this, the classification problem of the data set of students is used. In this example, the data set *Students.arff* is used for training the classifier model. The marks of students are given in three phases. On the basis of marks obtained in sequence, the final class attribute is determined. In the first record, if a student scored 90, 89 and 89 marks, respectively, then the student was given admission to Stream 1. In the third record, Stream 2 is allocated to the students who obtained marks 78, 67 and 78, respectively.

```
Students.arff
@relation students
@attribute marks1 numeric
@attribute marks2 numeric
@attribute marks3 numeric
@attribute class {1, 2}
@data
90, 89, 89, 1
89, 90, 99, 1
78, 67, 78, 2
67, 71, 78, 2
69, 78, 78, 2
60, 79, 78, 2
```

The test data set *teststudents.arff* is used to predict or determine the class or stream of the students who obtained marks in a specific sequence. This problem is solved using Weka with the integration of the machine learning algorithm of J48 classifier. In the following data set, we have to determine the classes (streams) of the students on the basis of their performance (scores) in the examination.

```
teststudents.arff
@relation students
```

```

@attribute marks1 numeric
@attribute marks2 numeric
@attribute marks3 numeric
@attribute class {1, 2}
@data
99, 91, 90, ?
89, 67, 78, ?
78, 67, 78, ?
77, 71, 78, ?
90, 78, 78, ?
10, 10, 10, ?
40, 40, 78, ?
30, 30, 80, ?
98, 97, 94, ?

```

Table 1: List of major ML tools and software libraries

<b>CNTK</b>	<b>Apache SystemML</b>	<b>Caffe</b>
Deeplearning4j	ELKI	GNU Octave
H2O	Keras	KNIME
Mahout	Mallet	Mlpack
MXNet	OpenNN	Orange
PyTorch	RapidMiner	scikit-learn
Shogun	Spark MLlib	TensorFlow
Theano	Weka	Yooreeka

Figure 2 depicts the Weka GUI Chooser interface in which there are multiple options to work with the data science and machine learning based implementations. In this option, there are multiple options including Explorer, Experimenter, Knowledge Flow, Workbench and Simple CI. For traditional implementations, Explorer is used by data scientists, as it has user friendly interfaces to choose the data set and apply different algorithms without cramming any instructions or syntax for the algorithmic implementation.

From the Preprocess Tab in Weka, as shown in Figure 3, the training data can be selected. In the option to open the file, the data scientist can select the data set that is required to be trained for the modelling and processing of the classifier, as per the current scenario.

Once the training data set is selected and imported to the Weka interface, the target class is required to be mentioned. As per the training data, the attribute ‘class’ is used here as the target. It means that the ‘class’ is the determined value on the combinations and associations of other attributes — marks1, marks2 and marks3.

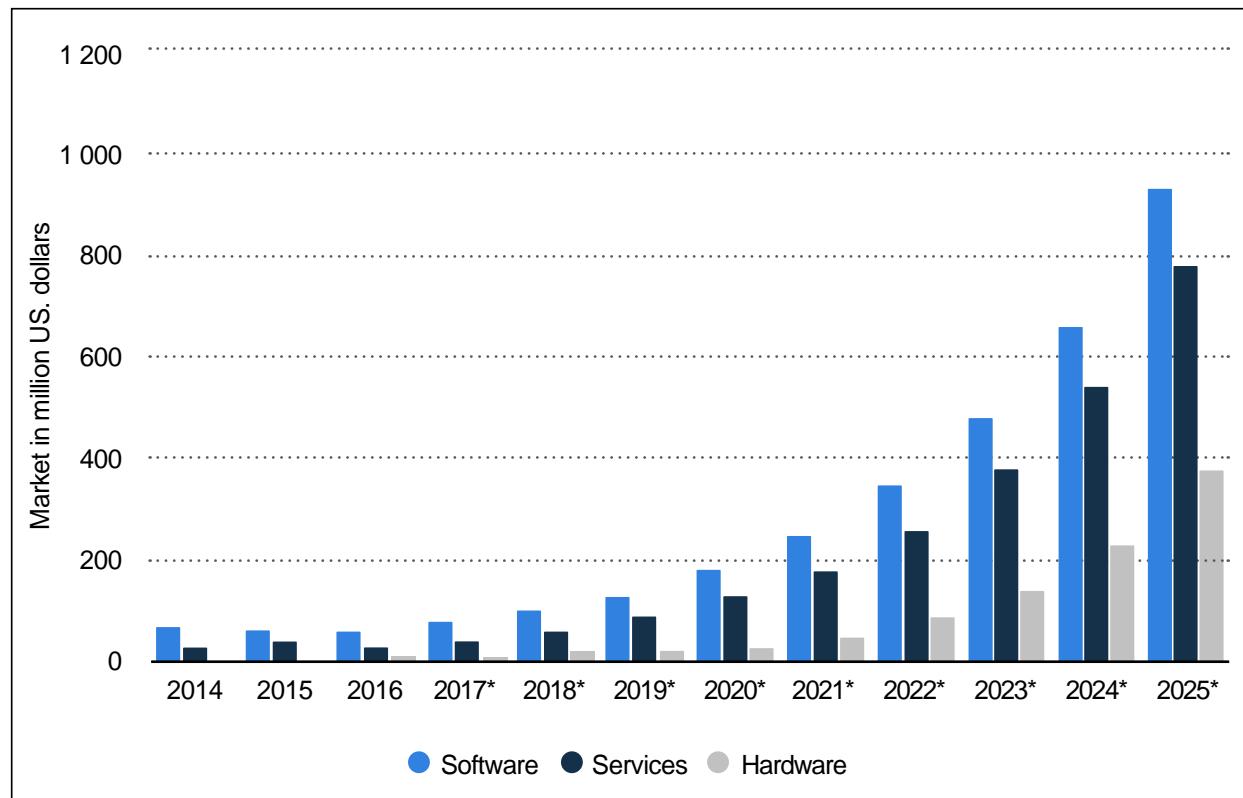


Figure 1: Predicted revenues from the US deep learning market

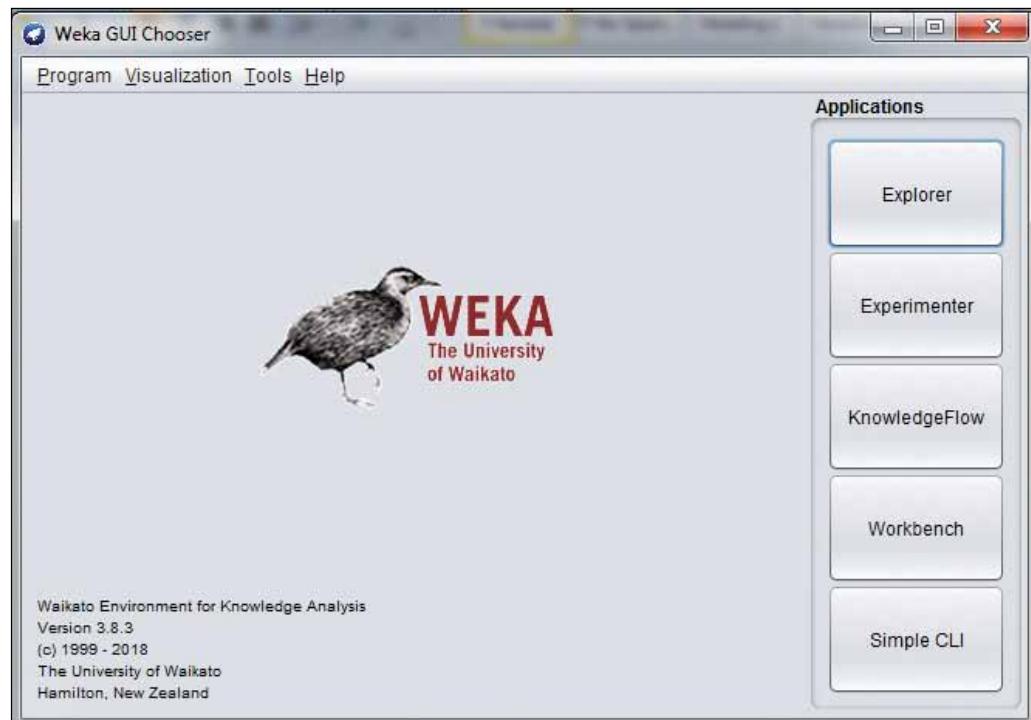


Figure 2: Weka Explorer

In Weka, there are assorted algorithms for data science and machine learning that can be called and attached with the data set to be processed. Figure 5 presents the option to select the classification algorithm of J48 so that the classification model can be built on the basis of the training data selected earlier.

Once the data set and classification model are invoked, there is a need to run the classifier so that the model can be trained. In this way, the classification model is correlated with the association of determining attributes and the determined attribute. In this example of students, the determining attributes are marks1, marks2 and marks3. The determined attribute or target is the class that has an association with the determining attributes or dependent attributes in the training data set.

The classification algorithm for machine learning is executed and then has to be saved so that the prediction on the testing or validation data set can be done. This means that the trained model has the association functions and the mathematical modelling of all the attributes. These mathematical modelling functions are further required for the prediction of test data that does not have the classes. The predicted class of the testing or validation data is determined on the functions created by the trained classification model as per the algorithm executed.

For the prediction of the training data set, the test data set is matched with the saved model. The saved classification model is loaded in the Weka panel and then the option of 'Supplied test set' is used for testing data. The testing data set is called using the 'Set' option so that it can be predicted with the saved classification model.

After loading the saved classification model of machine learning, the test (validation) data set is read so that the unknown classes (represented as '?' ) in the testing data can be predicted.

To view the predicted classes, the option of Output Predictions' in the 'Classifier Evaluation Options' is set to 'PlainText' so that the unknown classes

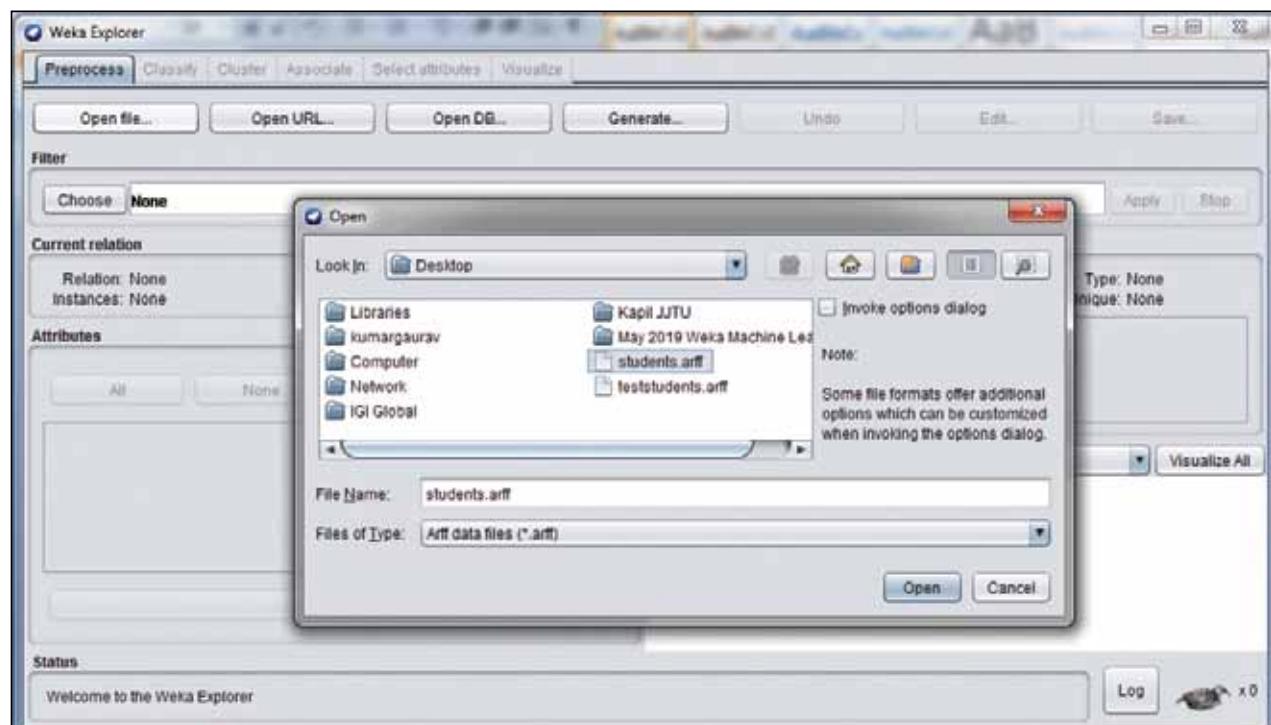


Figure 3: Reading the training data set for machine learning

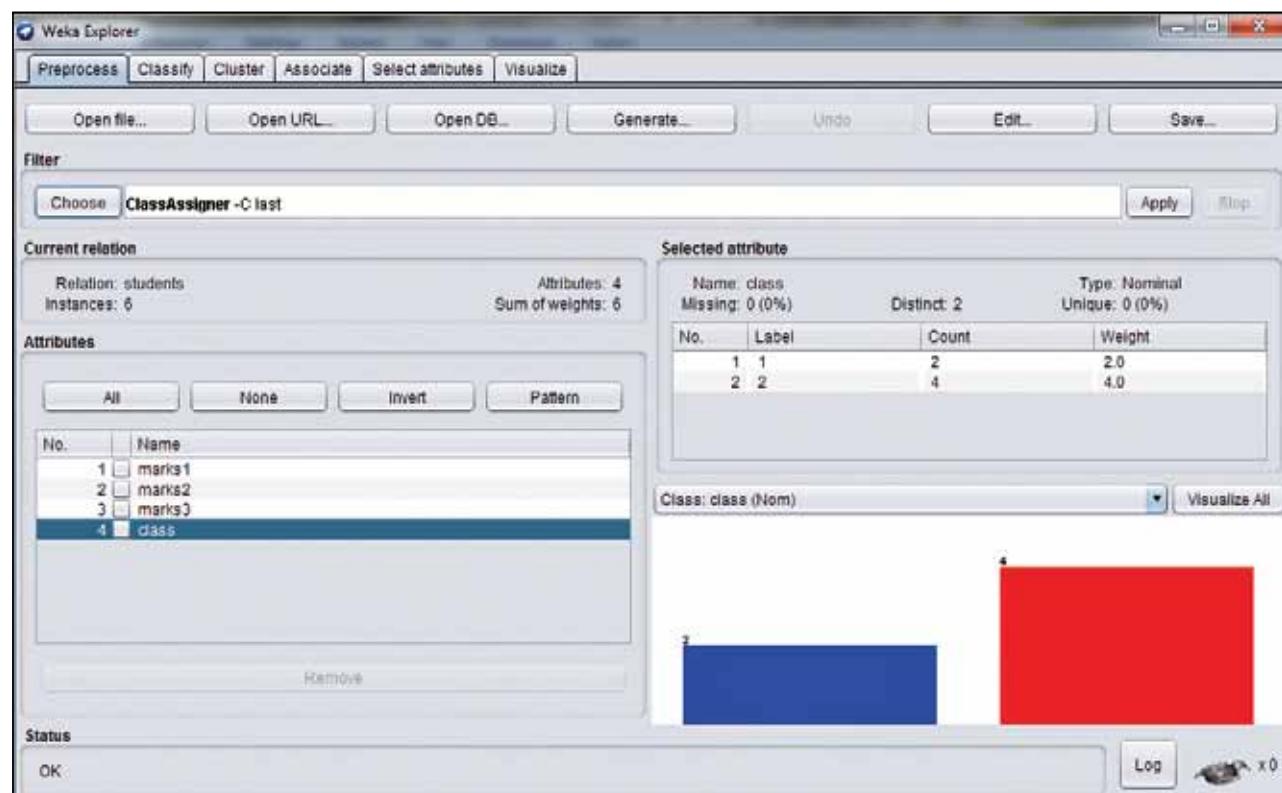


Figure 4: Assigning the target class

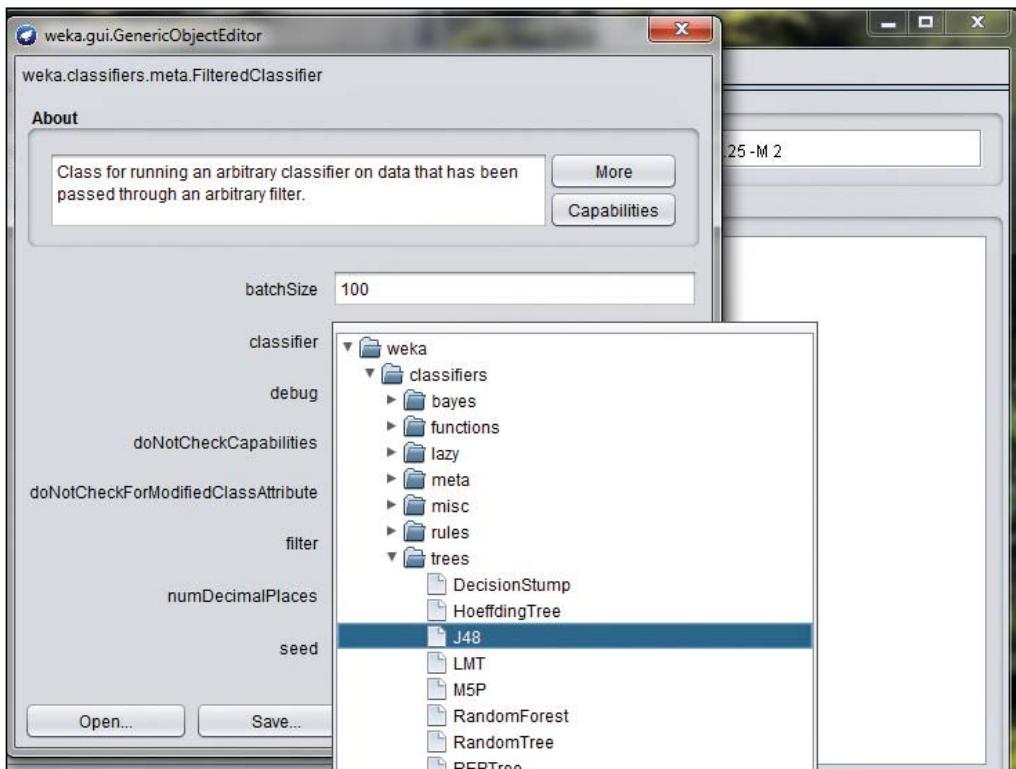


Figure 5: Selection of the classification algorithm

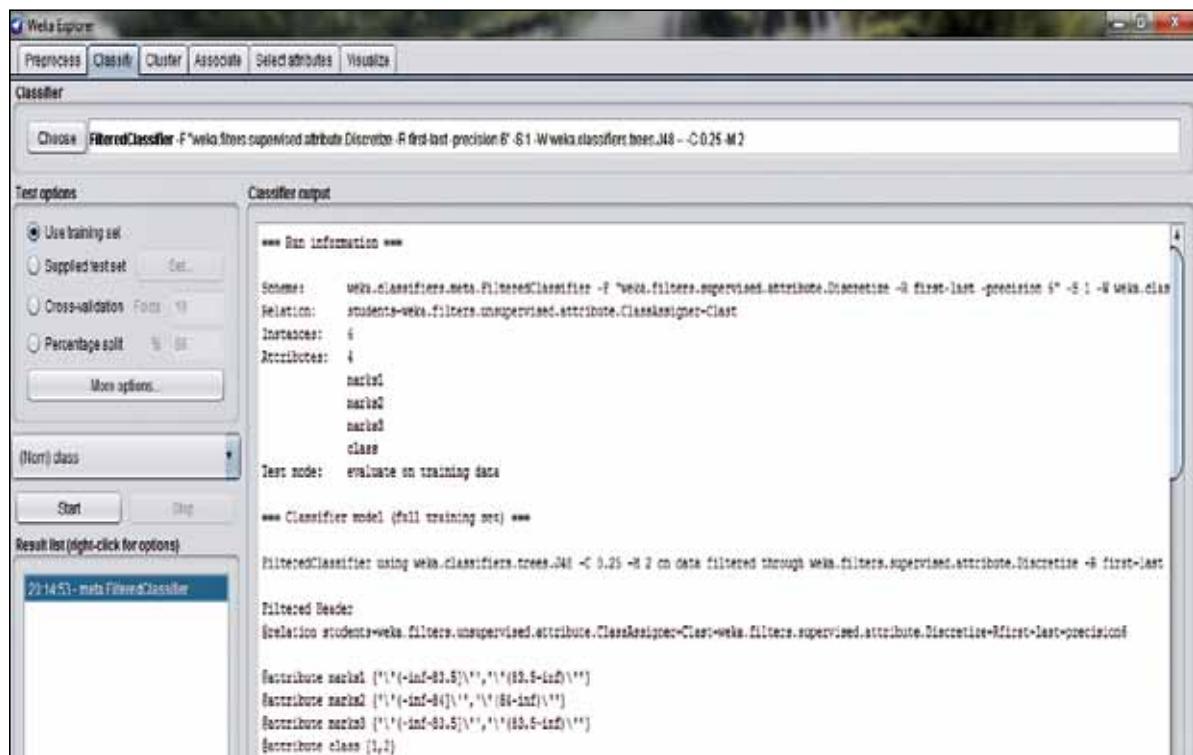


Figure 6: Running the classifier on the training data set

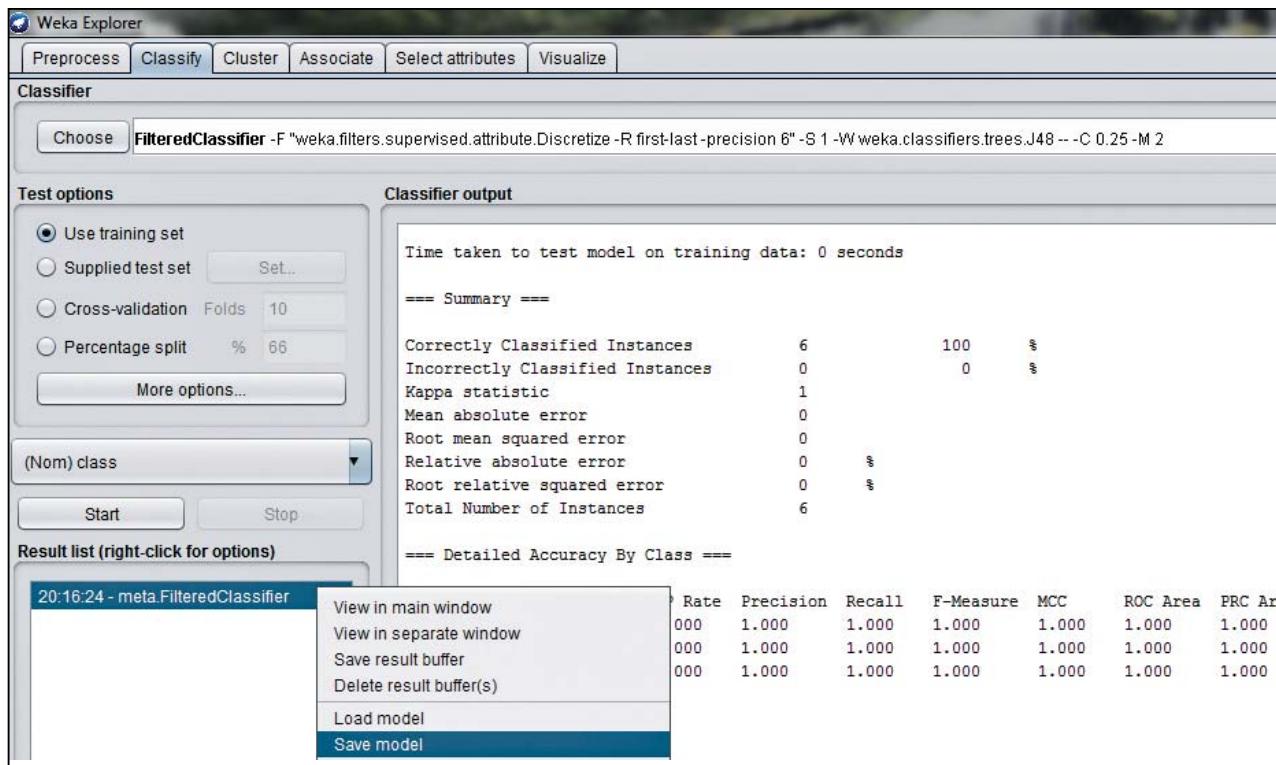


Figure 7: Saving the model for the prediction of test data

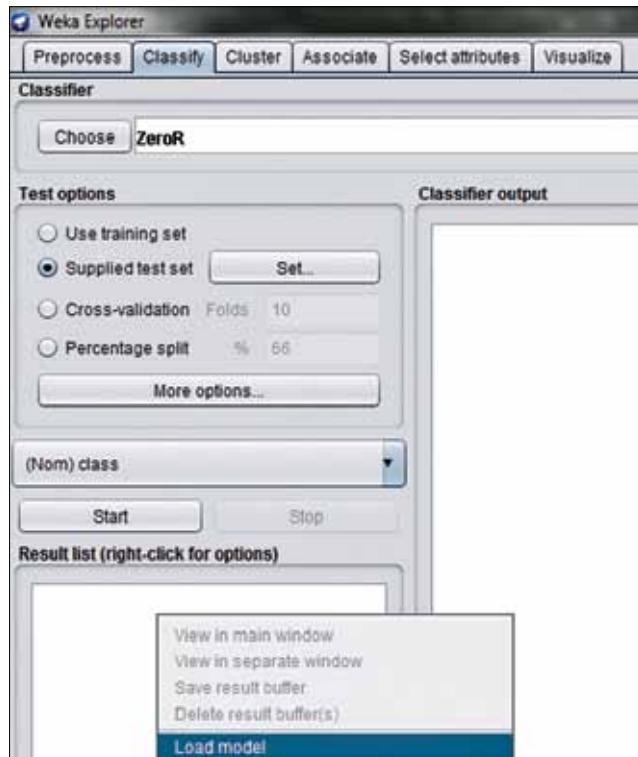


Figure 8: Loading the model for the evaluation and prediction of test data

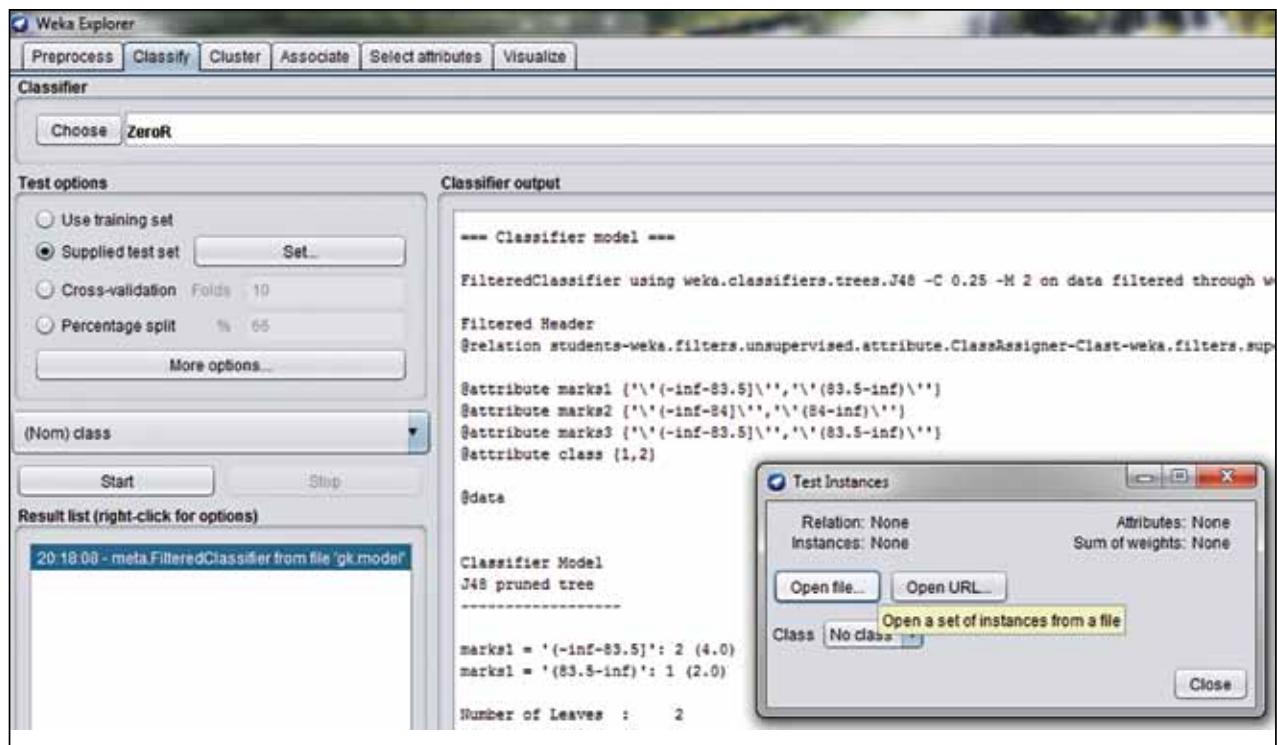


Figure 9: Selecting the test data after loading the model

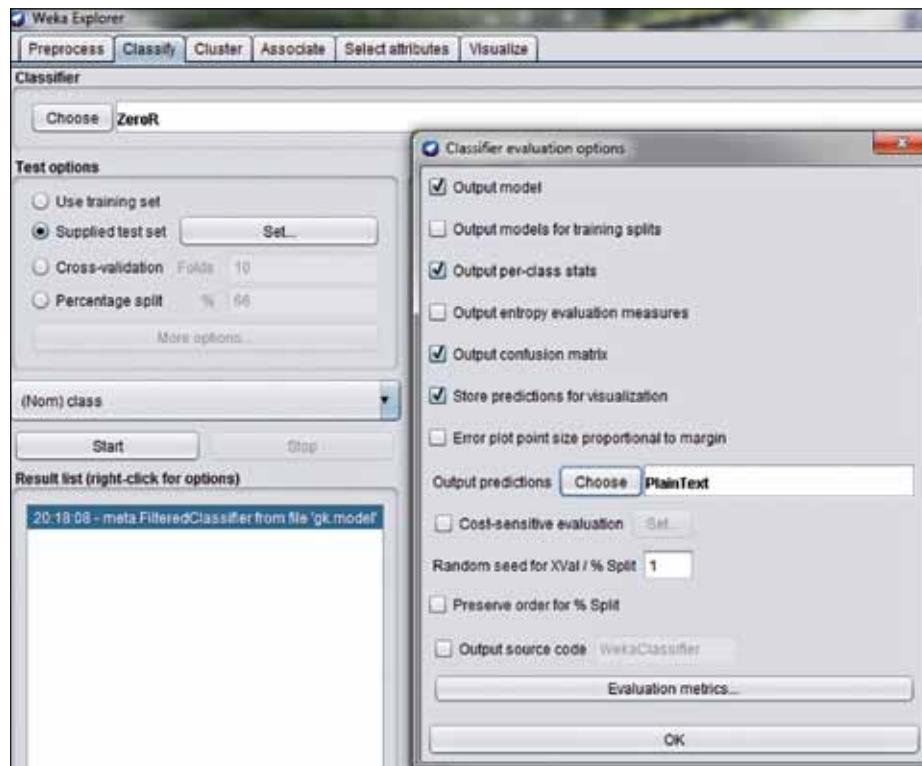


Figure 10: Enabling the plaintext option for the prediction of a class

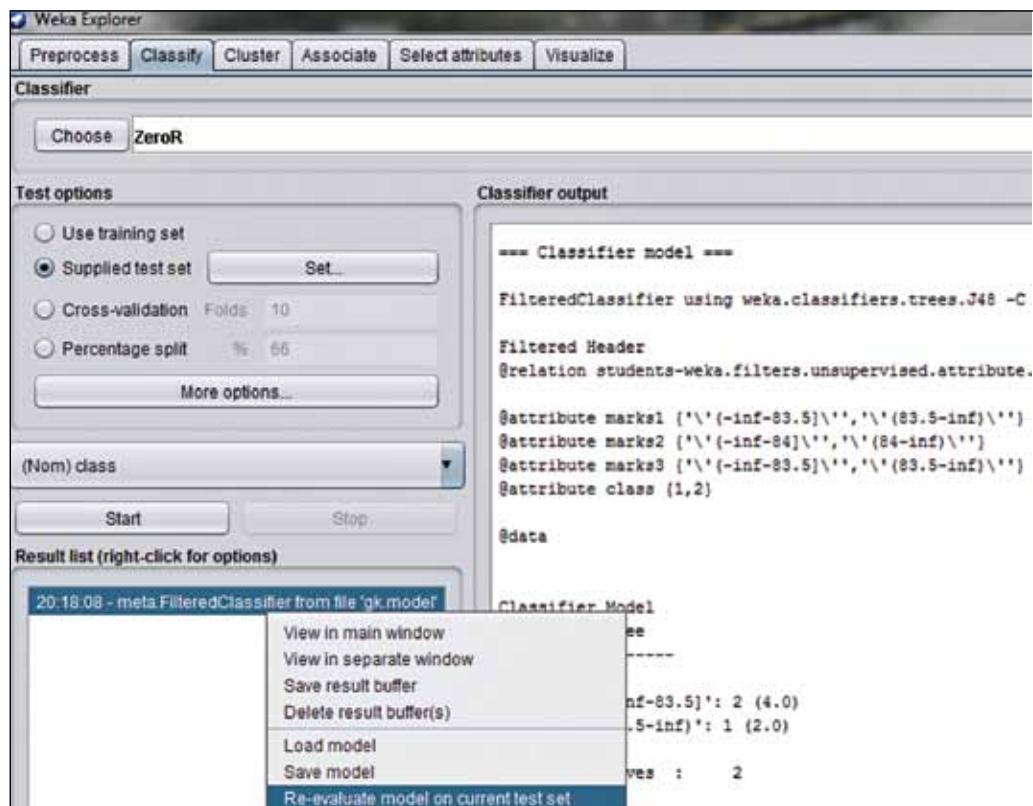


Figure 11: Executing the classification model on the test or validation data

can be viewed on the Weka interface.

After loading the saved model and invoking the testing data set, the re-evaluation of the model is done specifically for the supplied test set. It is used so that the supplied testing data set can be assigned the predicted classes on the same mathematical functions and model as used in the earlier classification model.

The predicted classes can be viewed in the right side panel of Weka after re-evaluation of the model. As in Figure 12, there are multiple attributes including the instance number, actual and predicted. In the predicted attribute, the unknown class (represented as '?') is assigned with the determined class on the same algorithm of machine learning. For example, in the first instance, the predicted class is 1, which was unknown in the test data set. In a similar way, the other values can be predicted.

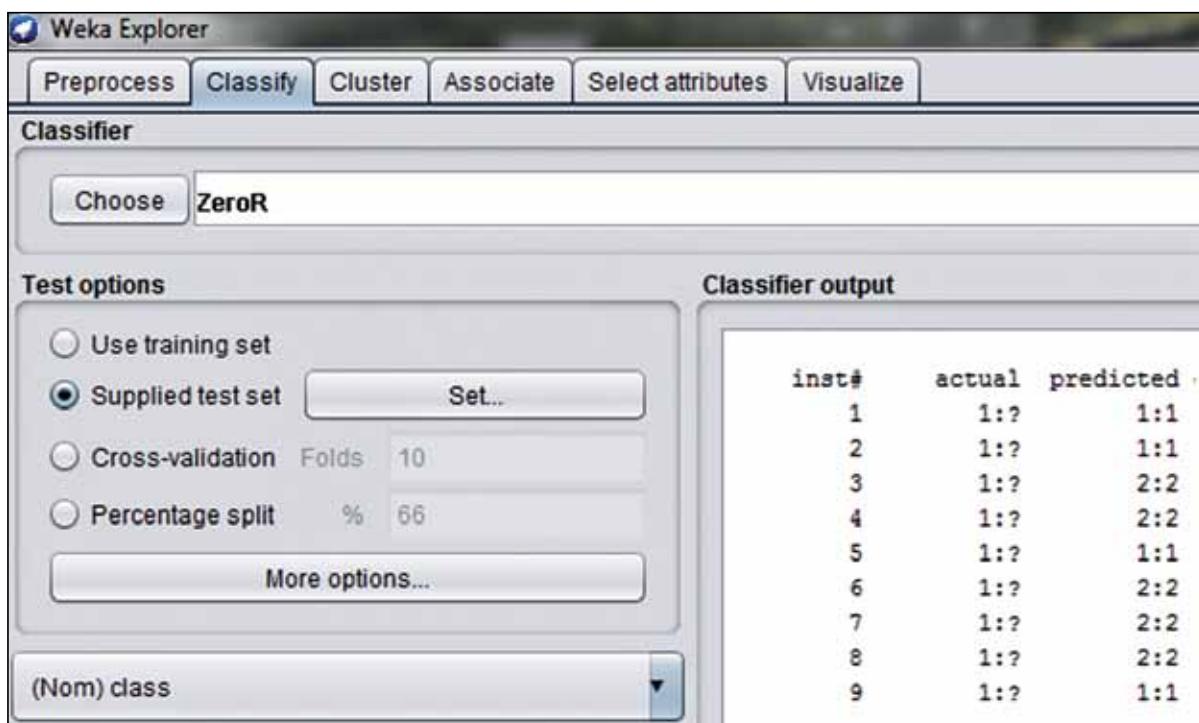


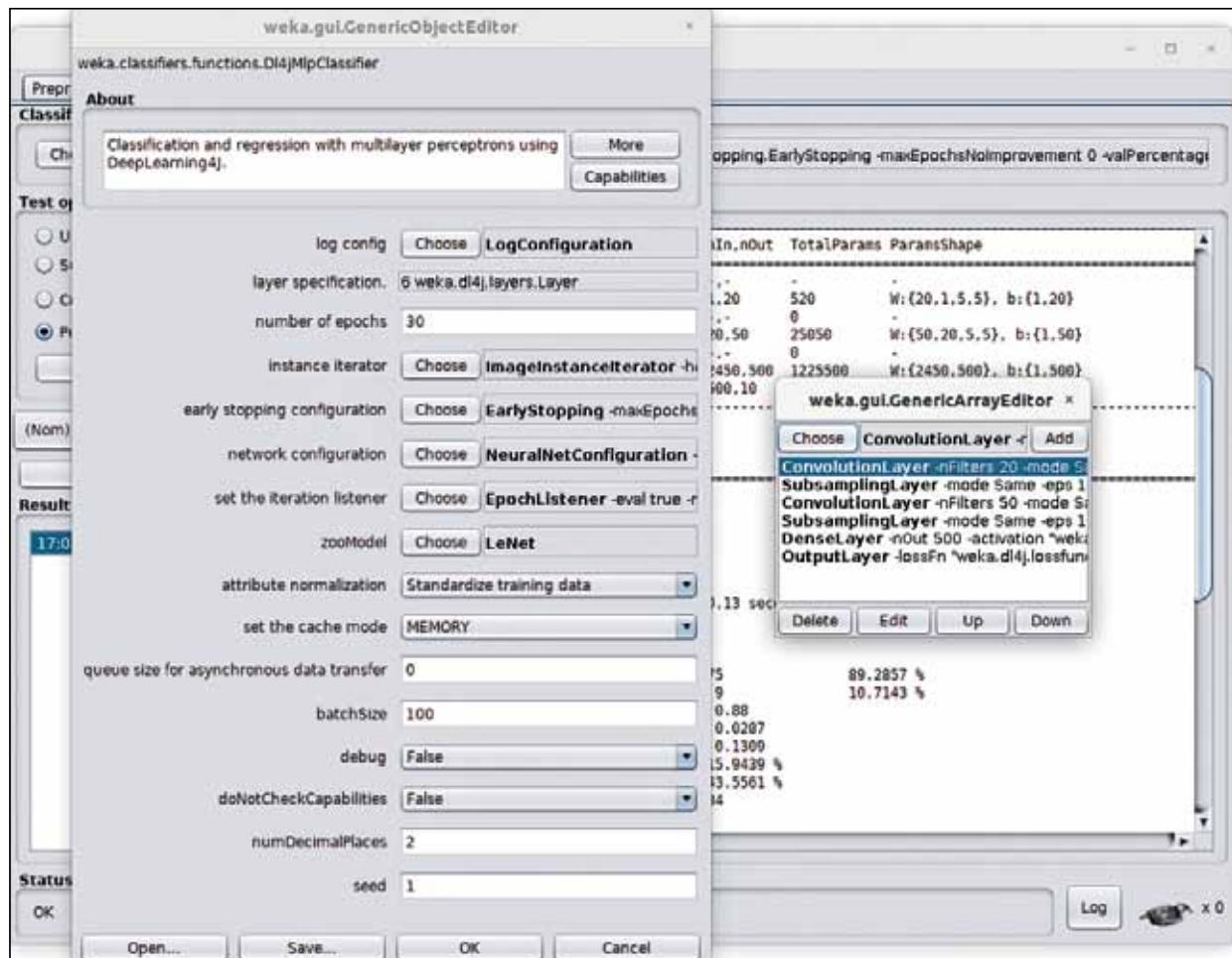
Figure 12: Analysis of the predicted classes

## Using WekaDeepLearning4j for deep learning in Weka

WekaDeepLearning4j (<https://deeplearning.cms.waikato.ac.nz/>) is the dedicated package for the implementation of deep learning in different applications. This library has been released so that the features and accuracy of deep learning can be used with the data analytics and predictive mining based applications. The power of Java programming is associated at the back-end of Weka with deep learning modules.

The following are the key features and layers integrated in Weka with deep learning:

- Convolution layer
- Dense layer
- Sub-sampling layer
- Batch normalisation
- Long short term memory (LSTM)
- Global pooling layer
- Output layer

Figure 13: Object editor in WekaDeeplearning4j (Source: <https://deeplearning.cms.waikato.ac.nz/>)

## Scope for research and development

The algorithms of machine learning, deep learning, data science and knowledge discovery are closely associated with scientific and engineering applications. They can be used for the development of new algorithms and for solving the engineering optimisation problems in the social as well as scientific domains. These algorithms have custom functions that can be updated as per the requirements of dynamic data sets to achieve a higher degree of accuracy and performance with related degrees of effectiveness.

# **ML.NET: The New Open Source Machine Learning Framework from Microsoft**

*Machine learning is the new rage. It is indeed an exciting time when new frameworks and new tools are being developed constantly. This is a write up on Microsoft's foray into open source with a new offering—ML.NET.*

The number of developers working on machine learning (ML) is growing fast. One of the main reasons for this is the availability of an increasing number of easy-to-use, powerful ML frameworks from many industry leaders such as Google, Microsoft, etc. ML.NET is a recent ML framework from Microsoft with cross-platform open source features. This article will introduce you to the features of the ML.NET framework, along with code examples.

Machine learning is the new superstar in cyberspace. It has applications across all domains and has solved problems that were earlier considered unsolvable. Prominent examples are image recognition, behaviour understanding, speaker neutral voice recognition with almost 100 per cent accuracy, etc.

Research in ML can be classified into two types. One is the actual algorithm development, and the other is exploring and building various applications using ML. The first one requires a deep understanding of various mathematical concepts. The second one, however, doesn't require you to understand the inner working mechanisms of the algorithms. The applications part of ML is gaining popularity across a wide spectrum of developers. The primary catalyst for this is the availability of various ML and deep learning (DL) libraries such as TensorFlow, Scikit Learn, Caffe, etc.

ML.NET was released by Microsoft on May 7, 2018. It is written in C# and C++, and its important attributes are listed below.

- ML.NET is cross-platform; it is supported on Linux, Windows and MacOS.
- It is open source (<https://github.com/dotnet/machinelearning>).
- It can be made to co-work with TensorFlow, CNTK, etc.
- If you are a .NET developer, you will find ML.NET very familiar and powerful for your .NET applications because of its machine learning features.
- The power of ML.NET has been proven with popular Microsoft features such as Windows Hello, Bing Ads.
- It has support for various ML scenarios such as sentiment analysis and recommendations. It supports deep learning (DL) scenarios also, such as image classification.

The complete list of ML.NET's components is shown in Figure 2 (Source: <https://blogs.msdn.microsoft.com/dotnet/2018/05/07/introducing-ml-net-cross-platform-proven-and-open-source-machine-learning-framework/>).

ML.NET has components that support all aspects of ML, which include core data types, customisable pipelines, high performance math, data structures for heterogeneous data, tooling support, etc.

## **Installing and building your first app with ML.NET**

As stated earlier, the ML.NET framework is cross-platform. It can be installed in the Windows environment with the following steps.

- *Step 1:* Download and install the .NET software development kit (.NET SDK).
- *Step 2:* Create the .NET app from the command prompt with the following commands:

```
> dotnet new console -o myApp  
> cd myApp
```

The above command builds a command line based .NET app in the directory named *myApp* and fills it with the required files for the app.

- **Step 3:** Install the ML.NET package by executing the following command:

```
>dotnet add package Microsoft.ML --version 0.4.0
```

For the Ubuntu 18.04 environment, the installation requires the following commands. First register the Microsoft key and the product repository, and then install the required dependencies:

```
wget -q https://packages.microsoft.com/config/ubuntu/18.04/packages-microsoft-prod.deb  
sudo dpkg -i packages-microsoft-prod.deb
```

Next, install the .NET SDK as shown below:

```
sudo apt-get install apt-transport-https  
sudo apt-get update  
sudo apt-get install dotnet-sdk-2.1
```

Download the ML.NET package with the following code:

```
$ dotnet add package Microsoft.ML
```

- **Step 4:** Any ML application has two components — the data set and the algorithm used to perform the actual task. For an ML app to succeed, both these components should be powerful. In this example application, we are going to work with a popular data set used by learners of ML applications – the iris data set. The purpose of this data set is to predict the type of iris flower. The iris data set can be downloaded from <https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data>. In Visual Studio, this data set can be copied on to the app's output directory.



Figure 1: ML.NET attributes

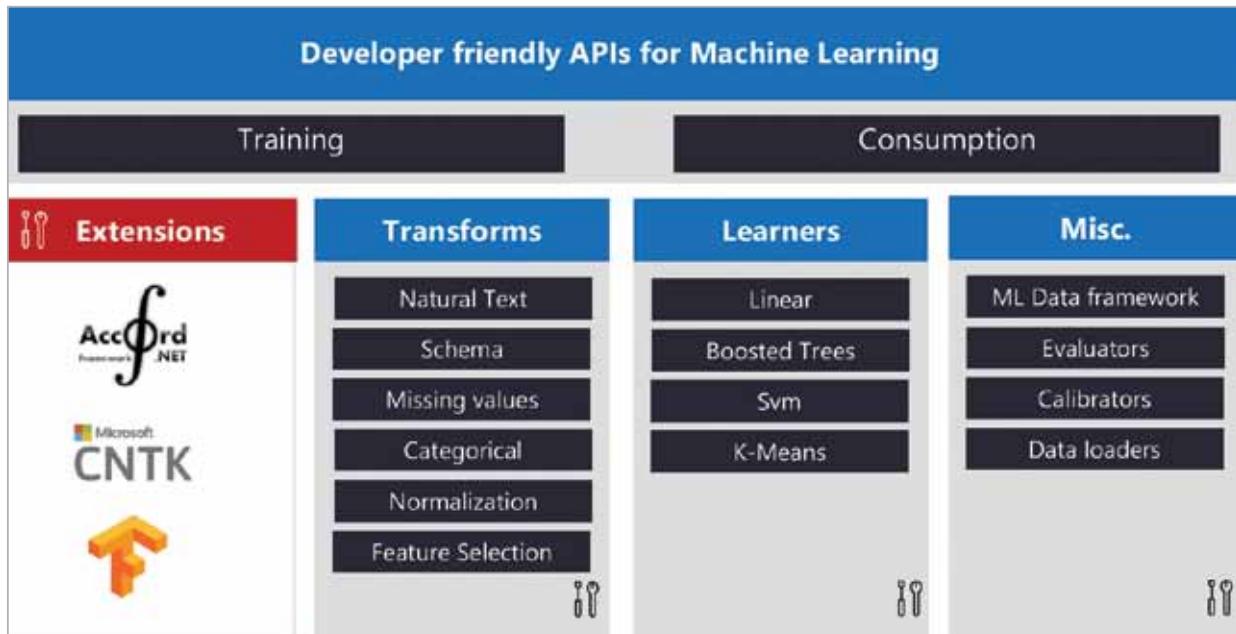


Figure 2: ML.NET components

- Step 5: The *Program.cs* file is the core file containing the code. This file starts with the inclusion of libraries (Code: <https://www.microsoft.com/net/learn/machinelearning-ai/ml-dotnet-get-started-tutorial>):

```

using Microsoft.ML;
using Microsoft.ML.Data;
using Microsoft.ML.Runtime.Api;
using Microsoft.ML.Trainers;
using Microsoft.ML.Transforms;
using System;
  
```

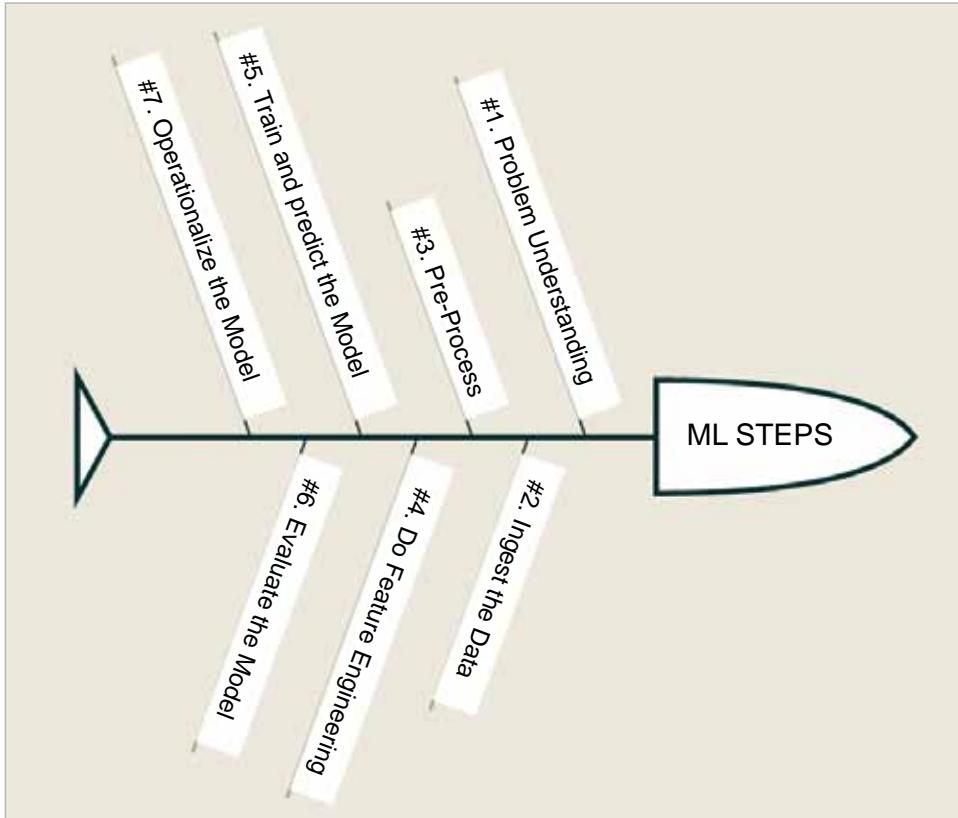


Figure 3: Using ML.NET to build a ML solution

Indicate the input features and class labels with the following code:

```
public class IrisData
{
    [Column("0")]
    public float SepalLength;

    [Column("1")]
    public float SepalWidth;

    [Column("2")]
    public float PetalLength;

    [Column("3")]
    public float PetalWidth;
```

```
[Column("4")]
[ColumnName("Label")]
public string Label;
}
```

For the prediction results, use the following class:

```
public class IrisPrediction
{
    [ColumnName("PredictedLabel")]
    public string PredictedLabels;
}
```

The core functionality is coded in the following function:

```
static void Main(string[] args)
{
    // STEP 2: Create a pipeline and load your data

    var pipeline = new LearningPipeline();

    // If working in Visual Studio, make sure the 'Copy to Output Directory' //
    // property of iris-data.txt is set to 'Copy always'

    string dataPath = "iris-data.txt";
    pipeline.Add(new TextLoader(dataPath).CreateFrom<IrisData>(separator: ','));

    // STEP 3: Transform your data // Assign numeric values to text in the "Label"
    // column, because only // numbers can be processed during model training

    pipeline.Add(new Dictionarizer("Label"));
}
```

```
// Puts all features into a vector

pipeline.Add(new ColumnConcatenator("Features", "SepalLength", "SepalWidth",
"PetalLength", "PetalWidth"));

// STEP 4: Add learner
// Add a learning algorithm to the pipeline.
// This is a classification scenario (What type of iris is this?)

pipeline.Add (new StochasticDualCoordinateAscentClassifier());

// Convert the Label back into original text (after converting to number in
step 3)

pipeline.Add(new PredictedLabelColumnOriginalValueConverter() {
PredictedLabelColumn = "PredictedLabel" });

// STEP 5: Train your model based on the data set

var model = pipeline.Train<IrisData, IrisPrediction>();

// STEP 6: Use your model to make a prediction // You can change these numbers
to test different predictions

var prediction = model.Predict(new IrisData() { SepalLength = 3.3f, SepalWidth
= 1.6f, PetalLength = 0.2f, PetalWidth = 5.1f, });

Console.WriteLine($"Predicted flower type is: {prediction.PredictedLabels}");
}
```

- *Step 6:* The app can be run with the following code:

```
>dotnet run
```

There are many ML.NET tutorials available on the official website (<https://docs.microsoft.com/en-gb/dotnet/machine-learning/tutorials/>), including:

- Sentiment analysis
- Iris clustering
- Taxi fare predictor

The major steps that need to be taken to solve most of the problems with ML.NET are:

- Understand the problems
- Ingest the data
- Pre-process the data and do feature engineering
- Train the model to make predictions
- Evaluate the model
- Operationalise the model

The sentiment analysis model can be trained with the following code:

```
var pipeline = new LearningPipeline(); pipeline.Add(new TextLoader<SentimentData>(dataPath, separator: ","));  
  
pipeline.Add(new TextFeaturizer("Features", "SentimentText")); pipeline.Add(new FastTreeBinaryClassifier()); pipeline.Add(new PredictedLabelColumnOriginalValueConverter(PredictedLabelColumn = "PredictedLabel"));  
var model = pipeline.Train<SentimentData, SentimentPrediction>();
```

The prediction can be done with the following code:

```
SentimentData data = new SentimentData  
{  
    SentimentText = "Today is a great day!"  
};  
SentimentPrediction prediction = model.Predict(data);  
Console.WriteLine("prediction: " + prediction.Sentiment);
```

A complete code example for sentiment analysis is available on the ML.NET official website at [\*https://docs.microsoft.com/en-gb/dotnet/machine-learning/tutorials/sentiment-analysis\*](https://docs.microsoft.com/en-gb/dotnet/machine-learning/tutorials/sentiment-analysis).

A detailed ML.NET cookbook with various examples is available at [\*https://github.com/dotnet/machinelearning/blob/master/docs/code/MLNetCookBook.md\*](https://github.com/dotnet/machinelearning/blob/master/docs/code/MLNetCookBook.md).

To summarise, ML.NET is a recent addition to the ML framework set. With its cross-platform support, it can be used in all major platforms such as Windows, Linux and MacOS. The ability to extend with other popular libraries such as CNTK makes ML.NET more powerful. .NET developers will find this framework very user friendly as it naturally fits into the .NET scheme of things. With its upcoming releases, the framework will mature and soon have the capability to solve much more complex machine learning and deep learning problems.

# Get Started with ML.NET, the Open Source and Cross-platform Machine Learning Framework for .NET

*IoT devices are becoming popular nowadays. The widespread use of IoT yields huge amounts of raw data. This data can be effectively processed by using machine learning to derive many useful insights that can become game changers and affect our lives deeply.*

Artificial intelligence (AI) is helping industries and consumers transform the way they work. Ubiquitous data, connectivity and the emergence of hyper-scale cloud platforms have opened the doors to innovative ways of solving business challenges. However, there is a need for experts, developers and technologies from all backgrounds to be able to leverage the power of AI to transform businesses.

Microsoft has been working towards the democratisation of AI technologies by aligning its research efforts towards solving problems that empower developers and data scientists to achieve more. The central theme of Microsoft's cloud and AI products is to deliver a platform that is productive, hybrid, intelligent, trusted and open. The Microsoft AI ecosystem spans intelligent apps (like Cortana) as well as deep learning toolkits and services, which developers can use to create bespoke intelligent solutions.

## Why should developers care about AI?

Cloud technologies allow APIs for intelligent services to be invoked from applications (regardless of the programming language). This is a faster and more productive way for applications to tap into the power of intelligent platforms. Developers are uniquely positioned to integrate AI with their solutions as they can identify key integration points where AI can add value

and make apps feature-rich. Features like locating certain objects in an image or scanning for text in an app running on the mobile can be implemented through API calls to the cloud AI. Hence, it is important to also identify which libraries and APIs are most suitable for app development and enhance productivity without having to refactor a lot of code.

Machine learning libraries and APIs have always appealed to data scientists and developers, as they help achieve results through the configuration of the APIs being invoked rather than writing the code from the ground up, thus saving time.

While the majority of AI-specific solutions are written in Python and R, a lot of SDKs allow developers to create and/or consume ML models in .NET Core (Microsoft's open source version of .NET), Java, JavaScript and other languages. This presents a significant opportunity for application developers to leverage these platforms to learn about ML and AI through their favourite languages.

## **Dotnet Core and machine learning**

Dotnet Core (aka .NET Core) is a free, cross-platform, open source developer platform for building many different types of applications. It offers support for writing apps in Windows, Linux and Mac OS on mobile devices, the Web, for gaming, IoT and ML in a language of your choice: C#, F# or Visual Basic. These languages have been around for a while and continue to be popular among developer audiences as application development platforms. While the use of .NET has been predominantly on the frontend/backend or middleware layers of a solution, the introduction of .NET for ML creates a new dimension on how application development platforms are evolving.

If you are skilled in open source platforms or have always been a .NET developer on Windows, this is the right time for you to start exploring this cross-platform, open source platform through proven programming languages in the stack.

While AI solutions have mostly been developed in Python and R, .NET Core introduces a new paradigm where the development and use of models is powered by a framework called ML.NET. The framework also has built-in implementations of popular algorithms like logistics, linear regressions, SVM and many others that are commonly used in the field of ML for model development.

ML.NET enables you to develop and integrate custom ML models into your applications even while you navigate through the basics of ML. Being an open source cross-platform framework for .NET developers, ML.NET is an extensible platform that powers Microsoft features like Windows Hello, Bing Ads, PowerPoint Design Ideas and more. This article focuses on ML.NET version 0.6.

## **ML.NET capabilities and features**

There are primarily three tasks in any ML experiment (or application).

1. Data processing
2. Creating a training workflow
3. Creating a scoring workflow

ML.NET provides namespaces and classes to achieve the above tasks through built-in classes and functions. For example, Microsoft.ML.Runtime.Data and Microsoft.ML.Core.Data have classes that help you to extract, load and transform data into the memory of the program for further processing by the ML engine. Similarly, there are classes in the Microsoft.ML.StaticPipe namespace which help you to create a pipeline for activities and configurations related to the ML aspects of AI. Finally, there are classes that help you to score or predict the basis of the developed model and get an output. There are also classes within the Microsoft.ML.Runtime.Data namespace that help you to output the performance metrics of the trained model. These metrics help you to understand how the model has been able to perform against training and test data. For example, to measure the performance of a binary logistic regression classifier (a model which outputs only two classes), we study the

attribute values of 'F1 Score', 'Accuracy', 'AUC', etc.

Given that ML.NET is built on the strong fundamentals and the same platform as other .NET Core technologies, you can use this inside any of your .NET Core apps like the console, Web (ASP.NET Core) and class libraries or apps running inside Docker containers on Linux.

## Time to get your hands dirty!

The ML.NET documentation on Microsoft Docs has a few walk-throughs to get started with some of the scenarios. We will look at using the Logistic Regression classifier on the MNIST data set.

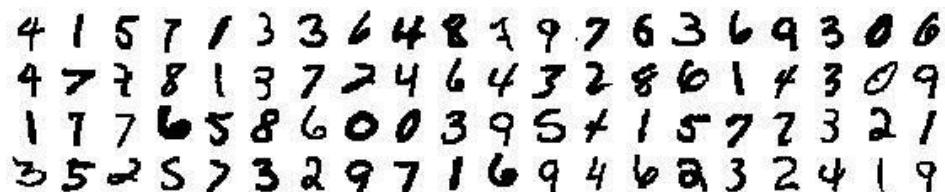


Figure 1: Images in the MNIST data set

The MNIST data set is like the 'Hello World!' of your machine learning journey. In almost any ML technology that you pick, you will come across an exercise that helps you understand the process using this data set.

The MNIST data set contains handwritten images of digits, ranging from 0 to 9. An example of what the source images look like is shown in Figure 1.

Identifying the digit in an image is a classification task and there are algorithms such as the Random Forest classifier or the Naïve Bayes classifier which can easily deal with multi-class classification. On the other hand, Logistic Regression is a binary classifier at the core, and it gives a binary outcome between 0 and 1 by measuring the relationship between the label and one or more features in the data set. Being a simple and easy-to-implement algorithm, you can use it to benchmark your ML tasks.

Prima facie, being a binary classifier, it doesn't seem suitable for MNIST data set classification, but when it is trained using the One-vs-All (OvA) method, you get a distinct classifier that can identify a specific class from the data set. When you perform the prediction, you just need to find out which classifier returns the best score to identify the digit on the image.

## Preparing yourself

If you have not yet installed the .NET Core framework, do so before proceeding any further. You can find compatibility and support information at <https://www.microsoft.com/net/download/dotnet-core/2.1>.

Let's create a directory called 'mnist' anywhere you like, and execute the following commands to create a .NET Core console app inside that directory and to add the ML.NET framework to our project:

```
dotnet new console  
dotnet add package Microsoft.ML
```

We can now move ahead on our journey to machine learning in .NET. Fire up your favourite text edit and open the 'Program.cs' file in your project directory. You will see the following 'Hello World!' code shown below. Let's start with this and change it as we move forward:

```
using System;  
  
namespace mnist  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            Console.WriteLine("Hello World!");  
        }  
    }  
}
```

If you are trying to build a .NET Core app for the first time, you can also build and run it by running ‘dotnet run’ in the project directory. Moving forward, the first order of business is to add all the namespaces that we will need to get this project running. Add the following lines to the top of the *Program.cs* file, as follows:

```
using System;
using System.IO;
using System.Linq;
using Microsoft.ML;
using Microsoft.ML.Core.Data;
using Microsoft.ML.Runtime.Api;
using Microsoft.ML.Runtime.Data;
using Microsoft.ML.StaticPipe;
using System.Threading.Tasks;
```

The next thing we need to set up is a few variables to point to our training and test data set, as well as the name and path of our ML model to save once we are done. Add the following lines under the class *Program* just before the *Main()* method:

```
static readonly string _datapath = Path.Combine(Environment.
CurrentDirectory, "Data", "optdigits-train.csv");
static readonly string _testdatapath = Path.Combine(Environment.
CurrentDirectory, "Data", "optdigits-test.csv");
static readonly string _modelpath = Path.Combine(Environment.
CurrentDirectory, "Data", "Model.zip");
```

## Data processing

You may have noticed that we have referred to a directory called *Data* in the code snippet above. This directory will host our MNIST data files as well as the trained model once we are done. Please download the MNIST 8x8 data set\* from the UCI Machine Learning Repository at <http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits>.

Click on the ‘Data Folder’ link on the page and download the following files in a directory called *Data* in your project directory:

```
optdigits.tra  
optdigits.tes
```

Just to make it easier for us, we will rename these files to *optdigits-train.csv* and *optdigits-test.csv*. If you haven’t yet looked at them, this is a good time to do so.

The MNIST data set we are using contains 65 columns of numbers. The first 64 columns in each row are integer values in the range from 0 to 16. These values are calculated by dividing 32 x 32 bitmaps into non-overlapping blocks of 4 x 4. The number of ON pixels is counted in each of these blocks, which generates an input matrix of 8 x 8. The last column in each row is the number that is represented by the values in the first 64 columns. These columns are our features and the last column is our label, which we will predict using our ML model once we are finished with this project.

It’s time to set up the data classes that we will use when training and testing our ML model. To do that, let’s add the following classes under the *mnist* namespace in our *Program.cs* file, as follows:

```
public class MNISTData  
{  
    [Column("0-63")]  
    [VectorType(64)]  
    public float[] PixelValues;  
  
    [Column("64")]  
    public string Number;  
}
```

```
public class NumberPrediction
{
    [ColumnName("PredictedLabel")]
    public string NumberLabel;

    [ColumnName("Score")]
    public float[] Score;
}
```

The *MNISTData* data class is our input data class. It contains the definitions of our features and the labels. We are using the *Column* attribute and a range of 0-63 to treat our 64 columns as an array of floats and are calling it *PixelValues*. We are also defining a size of 64 for these columns since the ML.NET trainer requires vectors of a known size. Next up is our label in the last column that we call *Number*.

The *NumberPrediction* class is used for the output of our ML model. The *PredictedLabel* column is of *uint* type and is called *NumberLabel*, and the *Score* column is a float array called *Score*.

## Creating a training workflow

With these basics in place, we can move ahead with building our ML pipeline. Remove or comment the 'Hello World!' line from the *Main()* method and add the following lines:

```
var env = new LocalEnvironment();
var model = Train(env);
```

We are creating an ML environment using *LocalEnvironment* and passing that to the *Train* method. Add the *Train()* method after the *Main()* method with the following code:

```
public static dynamic Train(LocalEnvironment env)
```

```
{  
    try  
    {  
        var classification = new MulticlassClassificationContext(env);  
        var reader = TextLoader.CreateReader(env, ctx => (  
            PixelValues: ctx.LoadFloat(0, 63),  
            Number: ctx.LoadText(64)  
>,  
            separator: ',',  
            hasHeader: false);  
  
        var learningPipeline = reader.MakeNewEstimator()  
            .Append(r => (  
                label: r.Number.ToKey(),  
                Features: r.PixelValues))  
            .Append(r => (  
                r.label,  
                Predictions: classification.Trainers.  
                    MultiClassLogisticRegression(r.label, r.Features)))  
            .Append(r => (  
                PredictedLabel: r.Predictions.predictedLabel.ToValue(),  
                Score: r.Predictions.score));  
  
        var data = reader.Read(new MultiFileSource(_datapath));  
        var model = learningPipeline.Fit(data).AsDynamic;  
        return model;  
    }  
    catch (Exception ex)  
    {  
        Console.WriteLine(ex.Message);  
        return null;  
    }  
}
```

*LocalEnvironment* is a class that creates an ML.NET environment for local execution. In this method, we are creating a classification context of type *MultiClassClassificationContext()* since our little project here deals with classifying the images as one out of ten numbers. We then use the *TextLoader* class to point to our data files and the structure of the data as explained in the data processing section of this article. The *TextLoader* class performs lazy reading, so the data will not be read until we perform a *Read* operation just before the training.

The learning pipeline is built using the *MakeNewEstimator()* class. An estimator is a class that represents the attributes related to an ML algorithm. We start by appending our label and features as the *Number* and *PixelValues* as defined in our input data classes, respectively. We now add the *MultiClassLogisticRegression* trainer to the pipeline and finally, define the structure of the predictions that we expect.

After this is done, we read the data from our training data file and perform the training task by calling the *Fit* method. This step will produce a model that we can save and reuse for making predictions with our testing and/or real-world inputs.

You can run this program now to see how the training task is performed by running *dotnet run* in the project directory. If no errors pop up during the execution of this program, our training was successful; our model will be returned, and can be saved for reuse later.

You can add the following code to the *Main()* method to save the file to disk:

```
using (var stream = File.Create(_modelpath))
{
    model.SaveTo(env, stream);
}
```

After these changes, you should be able to run the program, and see the model file saved to the file system for reuse, if needed.

## Testing the model

Continuing with this program, we will use our test data from *optdigits-test.csv* to see how the predictions from our model turn out. Here's how we can test the predictions. Add the following line in the *Main()* method:

```
Test(env);
```

Then, add the following *Test()* method to your project:

```
public static void Test(LocalEnvironment env)
{
    ITransformer loadedModel;
    using (var f = new FileStream(_modelpath, FileMode.Open))
        loadedModel = TransformerChain.LoadFrom(env, f);

    using (var reader = new StreamReader(_testdatapath))
    {
        int iCount = 0, cCount = 0;

        while (!reader.EndOfStream)
        {
            var line = reader.ReadLine();
            var pixels = line.Substring(0, line.Length - 2);
            var number = line.Substring(line.Length - 1, 1);

            var testData = Array.ConvertAll(pixels.Split(','), float.Parse);
            var predictor = loadedModel.MakePredictionFunction<MNISTData, NumberPrediction>(env);
            var prediction = predictor.Predict(new MNISTData() {
```

```
PixelValues = testValue, Number = number });

var predictedNum = int.Parse(prediction.NumberLabel);

    if (predictedNum != uint.Parse(number))
    {
        iCount++;
        Console.WriteLine($"'{iCount}' Incorrect prediction
{predictedNum} for {number}");
    }
    else
    {
        cCount++;
        Console.WriteLine($"'{cCount}' Correct prediction
{predictedNum} for {number}");
    }
}
Console.WriteLine($"Incorrect predictions: {iCount.
ToString()}");
Console.WriteLine($"Correct predictions: {cCount.ToString()}");
}
```

In this method, we are just reading our test data set and converting the values found in each line according to the *MNISTData* input data class defined earlier. The prediction is returned as a predicted label and since the return value is a string, we just convert this value to *int* type to get the actual number this model is predicting.

In the rest of code in this method, we are just comparing the predicted value against the actual number in the last column of our data set to see if our predictions are correct, and writing them to the console to make it easier to see how our model is performing. The following code is a truncated output from one of our executions, for your reference:

```
$ cd mnist
$ dotnet run
...
...
1713 Correct prediction 8 for 8
1714 Correct prediction 9 for 9
1715 Correct prediction 8 for 8
Incorrect predictions: 82
Correct predictions: 1715
```

Based on the total number of samples (1797) in our test data set, the above numbers indicate an accuracy of 95.43 per cent, which is a good number to have.

We have taken care to ensure you are able to follow the steps in this article, build this model from scratch and test it successfully.

Going forward, you could find ways to push the limits and increase the accuracy of this model. Some of the ways you can do that is by trying out a different classifier and/or using the higher resolution MNIST data set.

# An Introduction to Deeplearning4j, the Distributed Deep Learning Library

*There are many deep learning libraries that are becoming popular among the developer community such as Theano, Torch, Caffe, etc. Deeplearning4J is an open source and distributed deep learning library targeted at Java Virtual Machine (JVM). This article provides an introduction to its capabilities and features.*

Machine learning has led to a tremendous shift in the way we create algorithms to solve problems. It has made developers and researchers shift away from the traditional step-by-step solutions approach to a holistic model, which in many aspects mimics the methods that biological organisms employ while solving problems or acquiring a new skill. These days, machine learning models are employed in all those sectors that were originally difficult to deal with by the traditional algorithmic approaches. Real-time object detection from videos, classification and prediction systems are some of the tasks performed with the help of machine learning. The popular quote by Prof. Tom Mitchell, who is a pioneer in shaping the domain, clearly defines machine learning as follows: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E."

## Deep learning

Deep learning is a recent improvement in the machine learning domain. A basic requirement of machine learning is to identify and build a feature set. This task is generally carried out by experts in that domain, manually. So, for each problem

domain, customised features need to be built by expert humans. This often creates a bottleneck in the overall process flow.

Deep learning differs from traditional machine learning in the way features are built. It attempts to build the features automatically from the given *large* data set. Note the emphasis on the word 'large'. For deep learning to work at a decent accuracy level, we need considerably large sized data sets. This is significant, because the machine requires a fairly large amount of data in order to detect the discriminative features of a set. Feature engineering is an important component for the successful implementation of any machine learning related project.

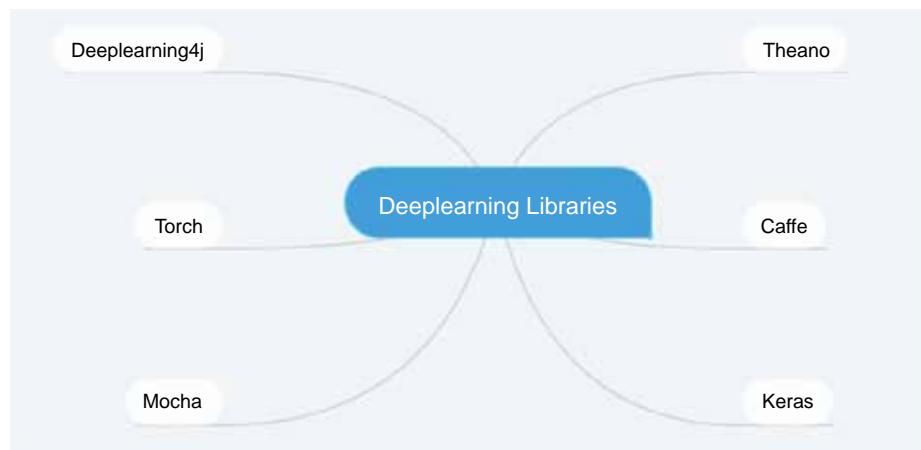


Figure 1: Popular deep learning libraries

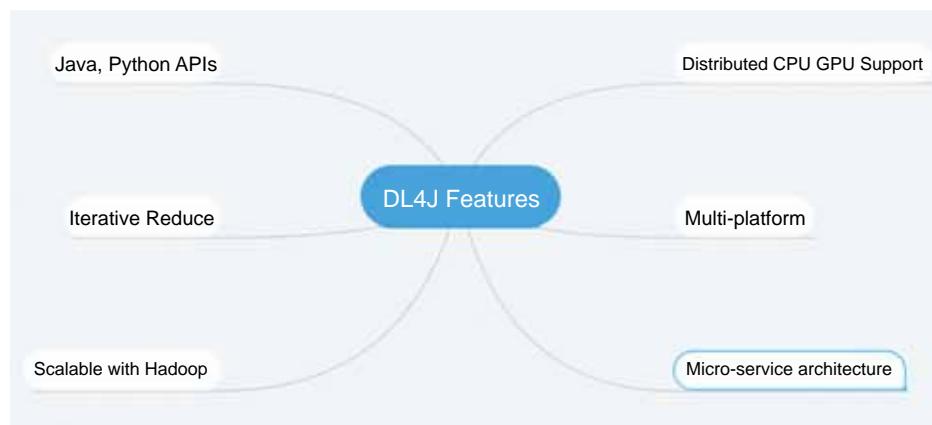


Figure 2: DL4J features

## Deep learning frameworks/libraries

There are many popular libraries that can be used to perform deep learning tasks. Some of them are listed below:

- Caffe
- Theano
- Torch
- Deeplearning4J
- Mocha

A detailed comparison of deep learning frameworks is available at <https://deeplearning4j.org/compare-dl4j-torch7-pylearn>.

## Deeplearning4j (DL4J)

This article explores the Deeplearning4J (DL4J) library. DL4J has been developed in Java and is targeted at Java Virtual Machine (JVM). An interesting feature of Deeplearning4J is the ability to build fast prototypes. The attributes of DL4J are listed below.

- *Distributed*: The training process can be accelerated because DL4J is distributed by nature. It can harness multiple GPUs, and the performance is on par with other major libraries such as Caffe.
- *Open Source*: DL4J is a production quality open source library available for performing deep learning tasks. The active community of developers keeps DL4J fresh.
- *Interoperability*: DL4J is written in Java, and hence all the JVM based languages such as Scala, Clojure and Kotlin are compatible. Interoperability with other major frameworks such as Caffe and Torch can be achieved through Keras.

## DL4J features

The major features of DL4J are:

- Java, Scala and Python APIs
- Parallel training through iterative reduce

- Scalable with Hadoop
- Distributed CPU and GPU support

DL4J incorporates both a distributed, multi-threaded deep learning framework and a single-threaded deep learning framework. Another important feature of DL4J is that it is the first deep learning framework adopted for a microservice architecture.

## Prerequisites

The prerequisites to start development with DL4J are listed below:

- Java 1.7 or above (DL4J supports only the 64-bit versions).
- *Apache Maven*: This is a dependency management and build tool.  
The Maven version in your system can be checked with the following command:

```
mvn --version
```

If you are new to Maven, an excellent ‘getting started’ guide (Maven in Five Minutes) is available at <http://maven.apache.org/guides/getting-started/maven-in-five-minutes.html>

- The official document recommends the use of IntelliJ IDE or Eclipse. The community edition of IntelliJ IDE can be downloaded from the official website.
- *Git*: Get the latest version with the following command:

```
$ git clone git://git.kernel.org/pub/scm/git/git.git
```

## Using the DL4J examples

To download and use the examples from DL4J, use the following commands:

```
$ git clone https://github.com/deeplearning4j/dl4j-examples.git  
$ cd dl4j-examples/
```

```
$ mvn clean install
```

1. Run IntelliJ. Choose the *Import Project* option. Select the folder 'dl4j-example'.
2. Select 'Import Project from external model' and make sure that Maven is chosen.
3. Follow the wizard's instructions. Click *Finish* to complete. It might take some time to get all the dependencies in the local system.
4. Select a particular example and right-click the file to run it.

The deep neural networks are made up of multiple layers. The MultiLayerConfiguration with parameters is customised to suit the requirements of the current problem. The hyperparameter variable decides the learning behaviour of the network. These parameters include the following:

- Number of times to update the weights of the model
- Initialisation mechanism for these weights
- The type of activation function to link with the nodes
- The specific optimisation algorithm to be used
- The speed with which the network should learn

A sample configuration is shown in the following code snippet:

```
MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()  
    .iterations(1)  
    .weightInit(WeightInit.XAVIER)  
    .activation("relu")  
    .optimizationAlgo(OptimizationAlgorithm.STOCHASTIC_GRADIENT_DESCENT)  
    .learningRate(0.05)  
    // ... other hyperparameters  
    .list()  
    .backprop(true)  
    .build();
```

A new layer can be added by invoking *layer()* on *NeuralNetConfiguration.Builder()*. In this process, we need to specify the following:

- Location (order) where the layer has to be added
- The number of input and output nodes (nIn and nOut)
- The type of layer

```
layer(0, new DenseLayer.Builder().nIn(784).nOut(250)
      .build())
```

After completing the configuration process, the training of the model can be carried out with the following command:

```
model.fit
```

## DL4J's neural networks

DL4J supports many powerful neural networks. Some of them are listed below:

- Restricted Boltzmann machines
- Convolutional nets
- Recurrent nets
- Deep-belief networks
- Stacked denoising autoencoders

## The reason for choosing JVM

Most of the deep learning/machine learning libraries are in Python. But DL4J is based on JVM. One may ask why this deliberate choice was made. The official documentation lists the major reasons for choosing the Java Platform:

- Many large government organisations and companies have Java as their major platform. There are millions of Java developers and Java is the largest programming language, as of now.
- Java is the basis of many powerful techniques/tools such as Hadoop, ElasticSearch, Lucene and Hive.
- One prime complaint against Python is its slow speed. Java is definitely

quicker than Python. Hence, when handling projects with massive data sets, Java may be a better choice.

- Java is inherently secure and cross-platform. It can be easily used on all major platforms such as Linux, Windows, OSX and Android.

## DL4J on Android

Deeplearning4j can be used with Android mobile phones as well. The prerequisites for running it on Android are:

- Emulator (with API level 21 or above) or a device
- Android Studio 2.2 or later

The following dependencies must be added to the *build.gradle* file:

```
compile 'org.deeplearning4j:deeplearning4j-core:0.8.0'  
compile 'org.nd4j:nd4j-native:0.8.0'  
compile 'org.nd4j:nd4j-native:0.8.0:android-x86'  
compile 'org.nd4j:nd4j-native:0.8.0:android-arm'  
compile 'org.bytedeco.javacpp-presets:openblas:0.2.19-1.3:android-x86'  
  
compile 'org.bytedeco.javacpp-presets:openblas:0.2.19-1.3:android-arm'
```

A clean and easy-to-follow tutorial is available at <https://deeplearning4j.org/android>.

This article has provided an introduction to the Deeplearning4J library. If you are interested in exploring more about deep learning, there are many interesting online courses and resources. A compiled list of links is available at <https://deeplearning4j.org/deeplearningforbeginners.html>.

# DeepLearning4j and PyTorch: Two Powerful Deep Learning Tools

*This article gives an introduction to two free and open source tools for deep learning and knowledge discovery—DL4J and PyTorch.*

Machine learning (ML) is a prominent area of research in the fields of knowledge discovery and the identification of hidden patterns in data sets. ML integrates predictive mining and the optimisation of results in multiple applications including bioinformatics, biometrics, computer vision, computational anatomy, criminal forensics, face detection, document forgery, fraud detection and many others. A comprehensive list of such applications is available on the Internet. ML focuses on the training of intelligent models so that the prediction can be accurate and fast.

Machines learn from the data and then get trained for prediction. For example, the preferences of customers are fetched from their existing records of interests.

## Deep learning

Deep learning (DL) is a more accurate and performance-aware flavour of machine learning, in which the error rate is very small compared to the latter. DL is also known as deep structured learning and hierarchical learning.

Let's look at speech recognition, for instance. If we install speech recognition software in a mobile phone for disaster/emergency purposes, when selected words like 'help' or 'kidnap' are uttered, a disaster message will be sent to chosen family members or friends. In this case, ML will detect only these two

words and will respond as per the data it has been trained on. In the case of DL, the messages analogous to the training data will also be processed, like ‘snatching’, ‘cannot see’ and similar words. This is the beauty and power of DL – it enables machines to learn on their own, from the training data set.

As shown in Figure 1, the process of feature extraction from the input is a separate one in the case of machine learning. Researchers need to create a separate method for feature extraction in the case of ML, using different mathematical formulae and scientific computations. The methods and approaches for feature extraction are integrated in the case of deep learning, which makes it more accurate and smart.

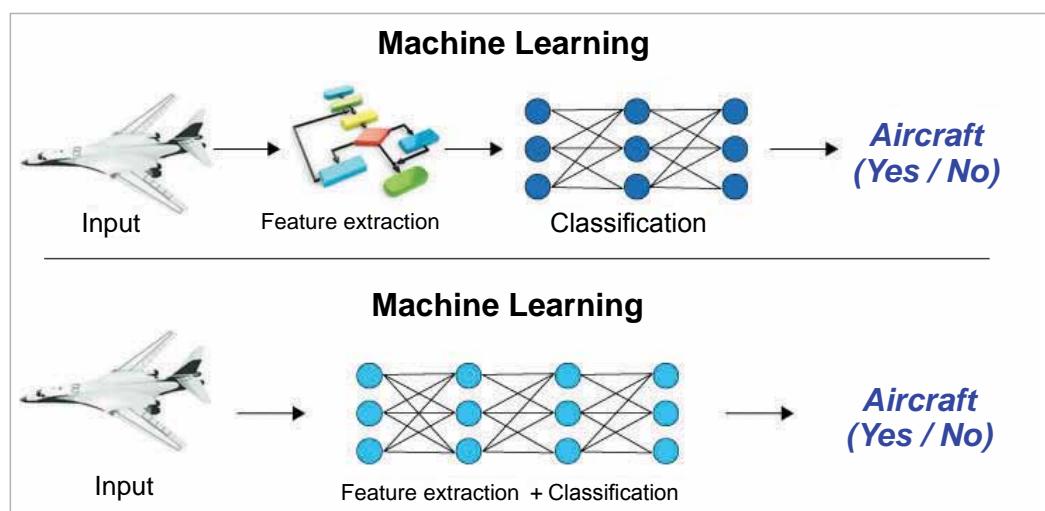


Figure 1: Differences between machine learning and deep learning

## DeepLearning4J: Deep learning in Java

DeepLearning4j (DL4J) (<https://deeplearning4j.org/>) is a Java based library that provides comprehensive and high performance implementations for DL in multiple applications. It is available as a free and open source deep learning library, with features like distributed computing for Java platforms, and can be used for knowledge discovery and deep predictive mining on graphics processing units (GPUs) as well as CPUs. The GPU is used for high performance and fast operations with a display of images, videos, animations and related multimedia objects.

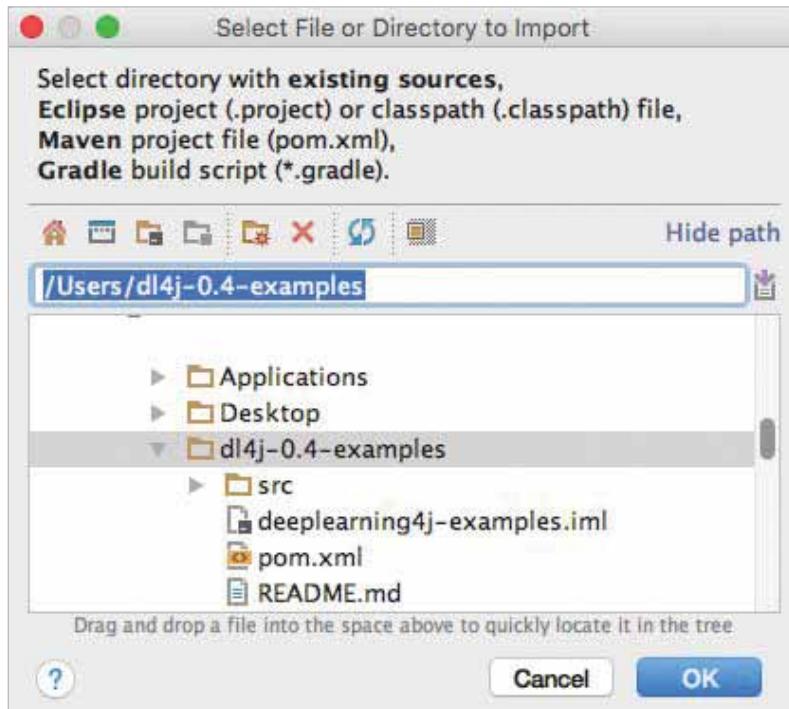


Figure 2: Import example project of DeepLearning4j

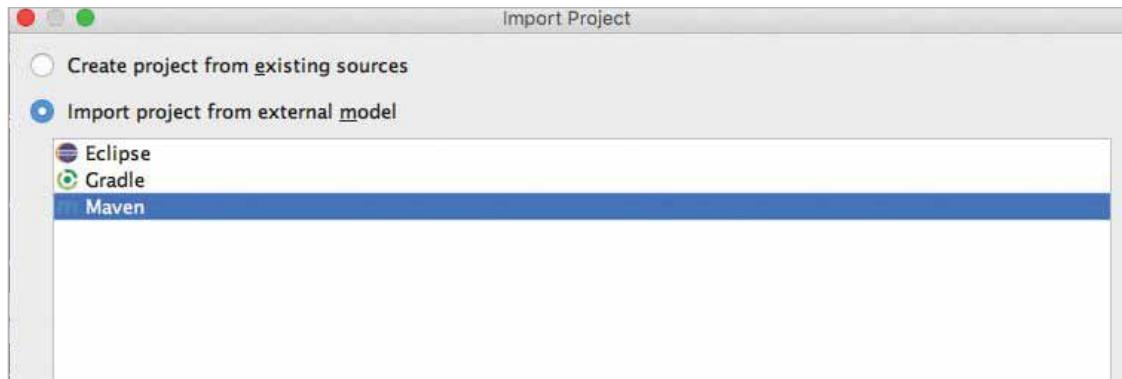


Figure 3: Import project and association with Maven

Using Eclipse DeepLearning4j, Big Data analytics can be performed along with Apache Spark and Hadoop, using Java as well as Scala programming. DL4J integrates the techniques and algorithms of artificial intelligence (AI), which can be used for business intelligence, cyber forensics, robotic process automation (RPA), network intrusion detection and prevention, recommender systems, predictive analysis, regression, face recognition, natural language processing, anomaly detection and many others.

In addition to the inherent high performance features, DL4J enables the import of models from other prominent DL frameworks including TensorFlow, Theano, Caffe and Keras. DL4J provides the interface for both Java and Python programming without any compatibility issues.

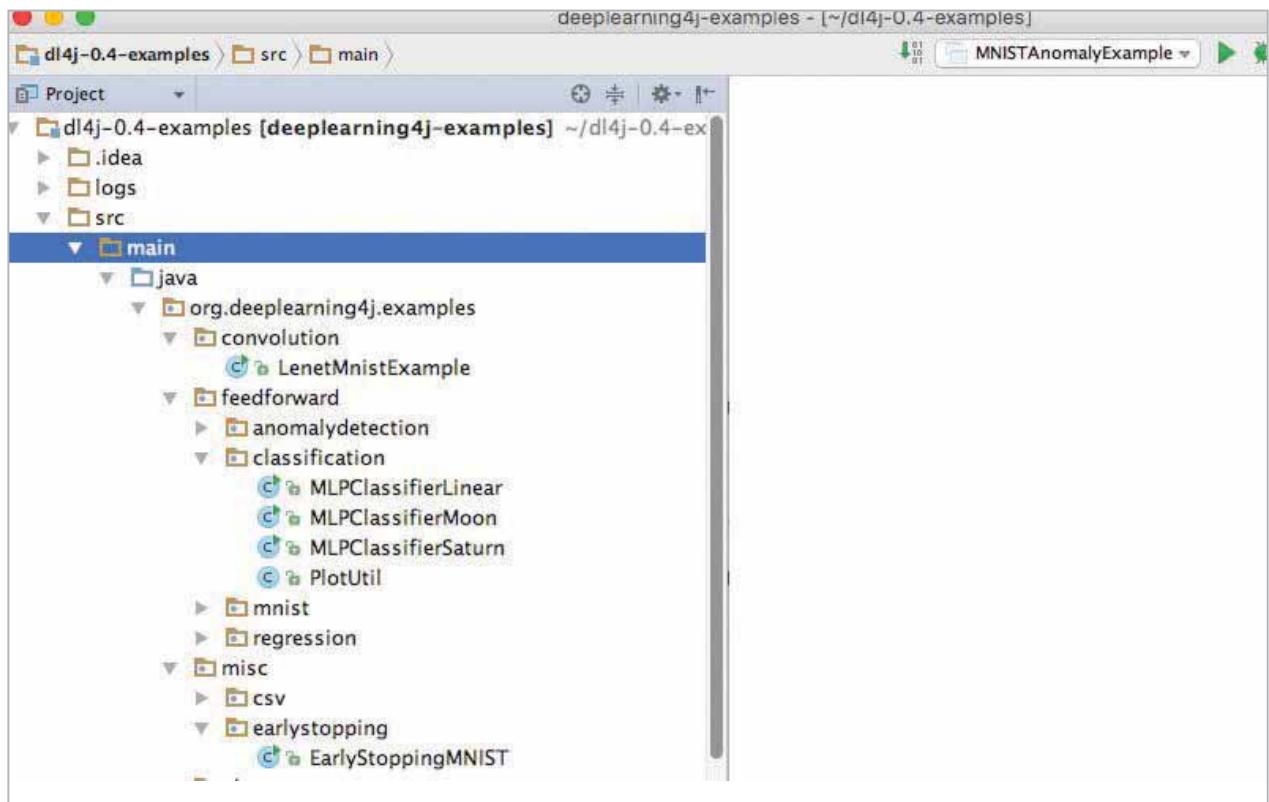


Figure 4: Select an example for execution

```

Run MLPClassifierLinear

o.d.o.l.ScoreIterationListener - Score at iteration 0 is 0.022690091133117676
o.d.o.l.ScoreIterationListener - Score at iteration 0 is 0.02802354335784912
o.d.o.l.ScoreIterationListener - Score at iteration 0 is 0.014803425073623658
Evaluate model.....

Examples labeled as 0 classified by model as 0: 100 times
Examples labeled as 1 classified by model as 1: 100 times

=====Scores=====

Accuracy: 1
Precision: 1
Recall: 1
F1 Score: 1

```

Figure 5: Output from executing the DeepLearning4j project

The key features embedded in DeepLearning4j include the following:

- Scalability on Hadoop for Big Data
- Microservices architecture
- Parallel computing and training
- Massive amounts of data can be processed using clusters
- Support for GPUs for scalability on Amazon Web Services (AWS) cloud
- APIs for Java, Python and Scala
- Distributed architecture with multi-threading
- Support to CPUs and GPUs

DL4J has the following components and libraries for assorted applications, in order to support multiple functions.

- *ND4J*: This is the integration of NumPy for Java Virtual Machine (JVM). ND4J can be used to call the libraries for fast processing of matrix data on the processing units. It is a library for numerical computations and processing with performance-aware execution of multi-dimensional objects for scientific applications, including signal processing, linear algebra, optimisation, transformations, gradient descent and many others. It is similar to scikit-learn and NumPy in Python.
- *JavaCPP*: This is the bridge and interface tool for Java and C++. It provides the interface and APIs to access C++ code in Java without any intermediate or third-party application.
- *DataVec*: This is the machine learning tool to extract, transform and load (ETL). The key function of DataVec is to transform the raw data to vector format with preprocessing to make it compatible with training in the ML implementations. It supports multiple formats including comma-separated values (CSVs), images, video, binary, text and many others.
- *RL4J*: This refers to reinforcement learning for Java platforms with the implementation of Deep Q Learning, asynchronous actor-critic agents (A3C) and many other algorithms for reinforcement learning for JVM.
- *Arbiter*: This is a tool to evaluate and test the algorithms of ML.

It supports searching for hyper-parameters to extract the best fit configuration of neural networks.

The following are the prerequisites to work with DeepLearning4j:

- Java Development Kit (JDK)
- Eclipse IDE or IntelliJ IDEA
- Git
- Apache Maven

After installation of the required software tools, the examples and use cases from GitHub can be downloaded for testing and further customisation of the code.

```
$ git clone https://github.com/deeplearning4j/dl4j-examples.git  
$ cd dl4j-examples/  
$ mvn clean install
```

From IntelliJ, the option of *Import Project* is selected, along with the directory ‘dl4j-examples’.

Select ‘Import project from external model’ and Maven as shown in Figure 3. Select the Deeplearning4j example for execution. In this example,

*MLPClassifierLinear* is used and run. A series of scores will be displayed at the bottom.

Figure 6 depicts the graph to show the classified data using multi-layer perceptron (MLP).

## **PyTorch: Deep learning in Python**

PyTorch (<http://pytorch.org/>) is a free and open source framework for DL using Python programming. Dynamic models and neural networks can be programmed using PyTorch. It supports CPU as well as GPU based computing for DL algorithms.

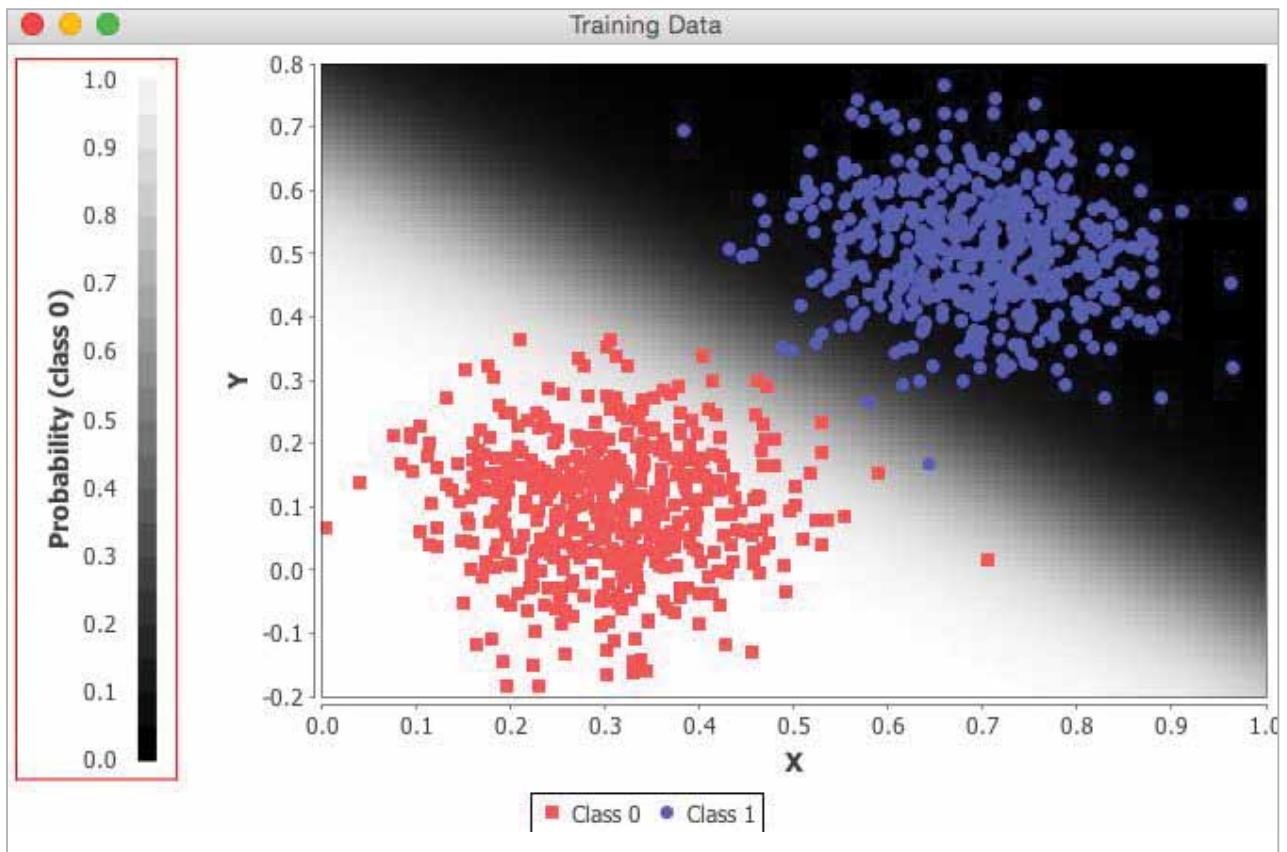


Figure 6: Graphical depiction of the classifier

PyTorch provides the following features for high performance DL:

- Tensor processing and computation with GPU support
- Deep neural networks
- Reusability of Python scientific packages on demand like SciPy, NumPy, scikit-learn and Cython
- Minimal overhead and complexities
- Fast processing regardless of the size of the neural network in computation
- Integration with acceleration and speed-up libraries including Intel MKL and NVIDIA to escalate the speed and minimise delay

The list of organisations, universities and companies working on developing PyTorch is shown in Figure 7.

The installation process of PyTorch varies depending upon the operating



Figure 7: Organisations developing PyTorch

system and the package manager, as specified in the official portal of PyTorch. The developers can install PyTorch by selecting the OS, package manager, Python version and CUDA support, as shown in Figure 8.

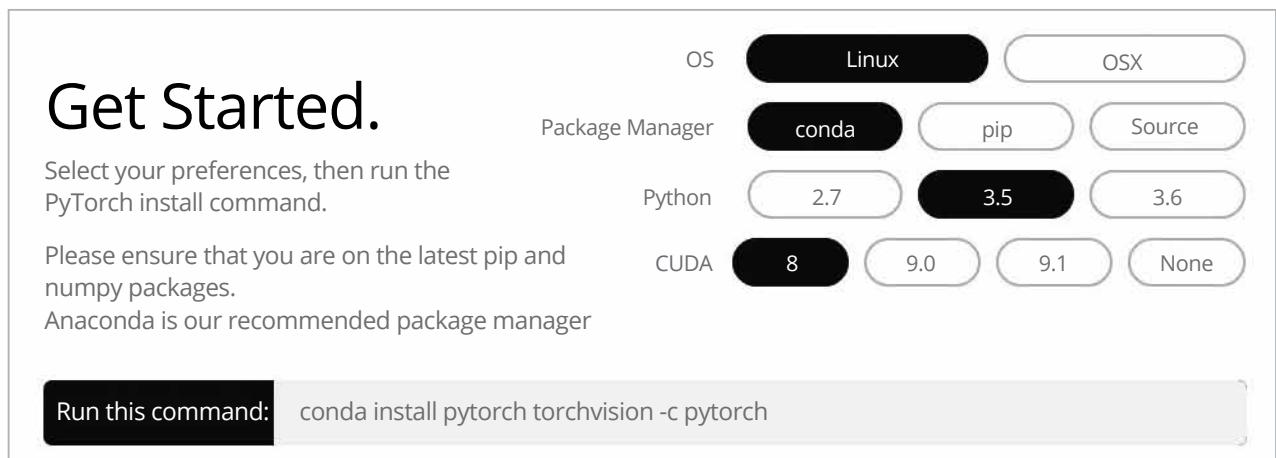


Figure 8: Installation instructions on different parameters

## Example of the classifier using deep learning in PyTorch

The features of deep learning are presented with a classification, using a benchmark data set CIFAR-10 as shown in Figure 9 (<https://www.cs.toronto.edu/~kriz/cifar.html>). This data set is used for training the model of deep learning. In this data set, multiple similar images are labelled. Using PyTorch, the image that does not exist in the data set can be predicted under a specific class and label category.



Figure 9: The benchmark data set of CIFAR-10



Figure 10: Testing the data set for classification and prediction

The output will be fetched as ‘plane horse cat bird’ because of the feature extraction and deep learning, based on the properties of these objects extracted from the training data set. Despite the fact that none of the images in Figure 10 exist in the training data, the deep learning algorithm is able to identify similar patterns and assign the class or label to the appropriate category.

## Scope for research and development

Deep learning libraries can be used for R&D in assorted applications including criminal face detection, flying objects analysis, intrusion detection at international borders, gesture recognition for criminal investigation, video forgery analysis, image tampering detection and many others, in which the identification of specific patterns is required. Deep learning libraries provide many models and approaches for prediction, with the maximum accuracy.

# The Best Machine Learning Libraries in Julia

*Machine learning (ML) is ubiquitous these days. This article introduces useful machine learning libraries in Julia, the developer base for which is growing fast. With the inherent advantages that Julia has in processing mathematical expressions, implementation of ML is very effective. This article takes you through popular libraries in Julia such as Flux, Knet, MLBase.jl and TensorFlow.jl.*

Machine learning is everywhere, and developers are trying to adopt it across all domains. The adoption of ML has changed the fundamental thought process behind building solutions. ML has given a new dimension to the process of building software solutions by changing it from a step-by-step approach into a learning-based approach. The inherent advantage here is the ability of the program to learn the intrinsic, hidden patterns in the data and to react to novel scenarios.

There are ML libraries available in many programming languages. C++, Python and R are popularly used in the ML arena. This article explores the popular machine learning libraries of Julia, a programming language that has combined the simplicity of Python with the speed of C++. Hence, by using Julia, you can get the best of both worlds – simplicity combined with speed. The Julia developer base has been steadily increasing over recent months. It has got applications in a wide spectrum of domains.

One of the prominent features of Julia is its ability to handle mathematical expressions with elegance and speed. If you have some exposure to ML, you might be aware that its core concepts are completely mathematical. Hence, by adopting Julia for machine learning, we get the inherent advantage of effective expression handling to optimise the machine implementations. Julia provides a wide variety of options for building ML solutions by offering

various libraries. Not just for ML, Julia in general is very rich in terms of the libraries and packages it offers. A complete list is provided in the Julia Observer page (<https://juliaobserver.com/>). Even the list of libraries that Julia offers to build ML solutions alone is very long, as shown in Figure 1.

This article provides a brief introduction to the following libraries in Julia:

- Flux
- Knet
- MLBase.jl
- TensorFlow.jl
- ScikitLearn.jl

## Flux

Flux is a popular choice for building machine learning solutions in Julia. This can be inferred from the Julia Observer Web page, which has higher ratings for Flux. The following are the advantages of using Flux:

- It is a very lightweight option.
- The entire Flux code is written in Julia. This is unlike libraries in some other languages for which the interface is in one language and the core components are in another such as C++.
- Another big advantage of Flux is that it comes with many built-in tools. It is fully loaded with features that are required for building a ML solution.
- Flux can be used along with other Julia components such as data frames and differential equation solvers.

You can start using Flux straight away without complex installation processes. You just need to use *Pkg.add*, as shown below:

```
Pkg.add("Flux")
```

As the official documentation says, Julia is simply executable math, just like

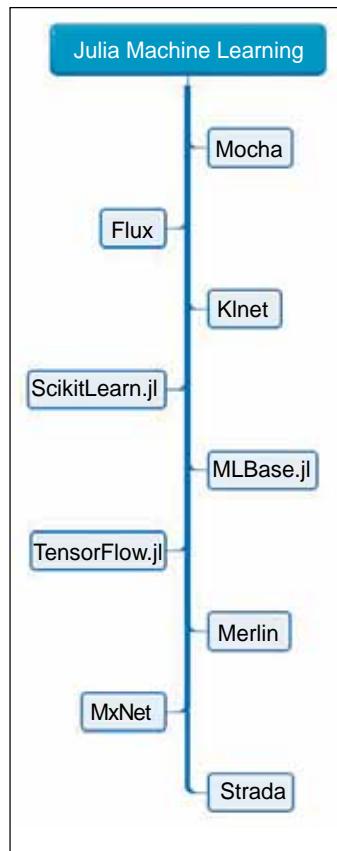


Figure 1: Julia machine learning options

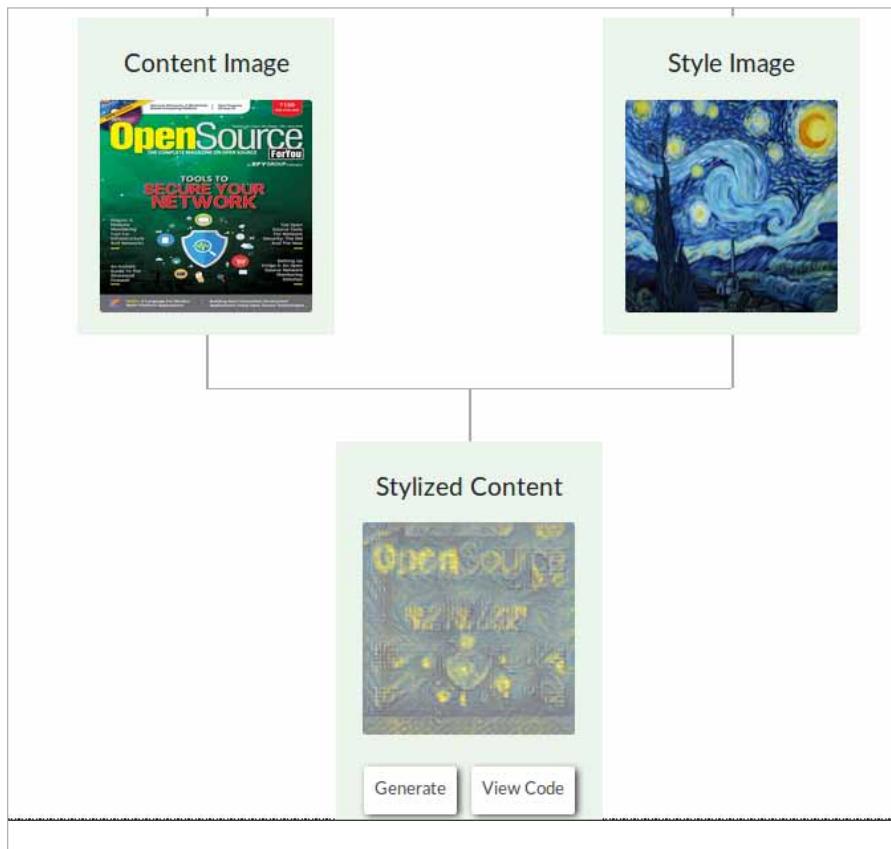


Figure 2: Style transfer demo with OSFY cover page

Python is the executable pseudo code. Hence the conversion of mathematical expressions into working code is elegant in Julia.

To start with, there are some executable Flux experiments listed in the <https://fluxml.ai/experiments/> page. You can navigate and see the live demos to get an insight into the power of Flux. For example, a style transfer demo application with the cover page of OSFY magazine is shown in Figure 2.

The official documentation (<https://fluxml.ai/Flux.jl/stable/>) lists the complete set of features provided by Flux.

The Model-Zoo has various demonstrations which can be very useful in

understanding the many dimensions of Flux (<https://github.com/FluxML/model-zoo/>). These models are categorised into the following folders:

- Vision: Large Convolutional Neural Networks (CNNs)
- Text: Recurrent Neural Networks (RNNs)
- Games: Reinforcement learning (RL)

If you are working in the field of computer vision, you may find Metalhead an interesting option. This package provides computer vision models that run on top of the Flux library. A sample code sequence to perform image content recognition is shown in Figure 3.

Overall, Flux is evolving into a major choice for ML development in Julia.

```
In [1]: using Metalhead
         using Metalhead: classify

In [2]: vgg = VGG19()
         img = load("Elephant.jpg")

Out[2]: A photograph showing an adult African elephant and its calf standing in a grassy, open landscape. The adult elephant is on the right, facing forward, while the calf is partially visible behind it, also facing forward. They have large ears and thick trunks.
```

```
In [3]: classify(vgg, img)

Out[3]: "African elephant, Loxodonta africana"

In [ ]:
```

Figure 3: Flux computer vision code sequence

## Knet

Knet has been built by Koc University, Turkey. It is a deep learning framework written in Julia by Dr Deniz Yuret along with other contributors (<https://github.com/denizyuret/Knet.jl>).

The Knet framework supports GPU operations. It has support for automatic differentiation, which is carried out using dynamic computational graphs.

Similar to Flux, you can start using Knet directly without any major installation process. You need to simply add it with `Pkg.add()` as shown below:

```
Pkg.add("Knet")
```

If you encounter any installation issues, follow the instructions given in the official Web page at <http://denizyuret.github.io/Knet.jl/latest/install.html>.

A simple example of using Knet for training and testing the LeNet model for the MNIST handwritten digit recognition data set is shown below (source: <https://github.com/denizyuret/Knet.jl>):

```
-  
using Knet  
  
# Define convolutional layer:  
struct Conv; w; b; f; end  
(c::Conv)(x) = c.f.(pool(conv4(c.w, x) .+ c.b))  
Conv(w1,w2,cx,cy,f=relu) = Conv(param(w1,w2,cx,cy), param0(1,1,cy,1), f)  
  
# Define dense layer:  
struct Dense; w; b; f; end  
(d::Dense)(x) = d.f.(d.w * mat(x).+ d.b)  
Dense(i::Int,o::Int,f=relu) = Dense(param(o,i), param0(o), f)  
  
# Define a chain of layers:
```

```
struct Chain; layers; end
(c::Chain)(x) = (for l in c.layers; x = l(x); end; x)

# Define the LeNet model
LeNet = Chain((Conv(5,5,1,20), Conv(5,5,20,50), Dense(800,500),
Dense(500,10,identity)))

# Train and test LeNet (about 30 secs on a gpu to reach 99% accuracy)
include(Knet.dir("data","mnist.jl"))
dtrn, dtst = mnistdata()
train!(LeNet, dtrn)
accuracy(LeNet, dtst)
```

Knet documentation has detailed instructions for handling major deep learning building blocks, some of which are listed below:

- Back propagation
- Convolutional Neural Networks
- RNN
- Reinforcement learning

If you are a researcher working in the deep learning domain, you may find the paper titled 'Knet: Beginning deep learning with 100 lines of Julia' by Dr Deniz Yuret interesting.

## **MLBase.jl**

Unlike the previous two libraries, the MLBase.jl library doesn't implement specific algorithms used in ML. But it provides a number of handy tools to assist building ML programs. The MLBase.jl may be used for the following tasks:

- Preprocessing and data manipulation
- Performance evaluation
- Cross-validation

- Model tuning

MLBase.jl depends on the StatsBase package. Detailed usage instructions are available at the official documentation page at <https://mlbasejl.readthedocs.io/en/latest/index.html>.

## TensorFlow.jl

TensorFlow.jl is a wrapper around the most popular ML framework from Google — TensorFlow.

TensorFlow.jl can be added using *Pkg.add()* as shown below:

```
Pkg.add("TensorFlow")
```

A basic usage example is shown below:

```
using TensorFlow
using Test

sess = TensorFlow.Session()

x = TensorFlow.constant(Float64[1,2])
y = TensorFlow.Variable(Float64[3,4])
z = TensorFlow.placeholder(Float64)

w = exp(x + z + -y)

run(sess, TensorFlow.global_variables_initializer())
res = run(sess, w, Dict(z=>Float64[1,2]))
@test res[1] ≈ exp(-1)
```

If you are a machine learning researcher, then you may find the paper titled ‘TensorFlow.jl: An Idiomatic Julia Front End for TensorFlow’ by Malmaud and

White very interesting. This paper has been published in the *Journal of Open Source Software*.

If you have worked with TensorFlow in Python, you may find the Julia version very similar. There are some inherent advantages of the Julia interface over Python which are illustrated in detail in the official documentation at [https://github.com/malmaud/TensorFlow.jl/blob/master/docs/src/why\\_julia.md](https://github.com/malmaud/TensorFlow.jl/blob/master/docs/src/why_julia.md).

## **ScikitLearn.jl**

ScikitLearn.jl is the Julia version of the popular Scikit-learn library. As you may be aware, ScikitLearn is a very popular option for building ML solutions.

The ScikitLearn.jl is a simple-to-use ML library in Julia with lots of features (<https://scikitlearnjl.readthedocs.io/en/latest/>). The major features of ScikitLearn.jl are listed below:

- Support for DataFrames
- Hyper parameter tuning
- Feature unions and pipelines
- Cross-validation

The availability of a uniform interface to Julia and Python models makes using ScikitLearn.jl effective and simple.

As stated in the beginning of this article, Julia is a powerful language when it comes to handling mathematical expressions. This core advantage is reflected in all the ML libraries built using Julia. In the near future, Julia could emerge as the leading choice for implementing machine learning solutions for a majority of developers.

# **Exploring Clustering Methods in Machine Learning**

*In machine learning, clustering is a technique that groups data points. Ideally, data points in a cluster should have similarities in properties or features, which differ from those not in the same cluster. This article discusses two clustering algorithms – k-means and DBSCAN.*

One of the most popular techniques in data science is clustering, a machine learning (ML) technique for identifying similar groups of data in a data set. Entities within each group share comparatively more similarities with each other compared to with those from other groups. Clustering means finding clusters in an unsupervised data set. A cluster is a group of data points or objects in a data set that are similar to other objects in the group and dissimilar to data points in other clusters. Clustering is useful in several exploratory pattern-analysis, grouping, decision-making and machine-learning situations, including data mining, document retrieval, image segmentation and pattern classification. However, generally, the data is unlabelled and the process is unsupervised in clustering.

For example, we can use a clustering algorithm such as k-means to group similar customers as mentioned and assign them to a cluster, based on whether they share similar attributes, such as age, education and so on.

Some of the most popular applications of clustering include market segmentation, medical imaging, image segmentation, anomaly detection, social network analysis and recommendation engines. In banking, analysts examine clusters of normal transactions to find the patterns of fraudulent credit card usage. Also, they use clustering to identify clusters of customers – for instance, to find loyal customers versus customers who have stopped using their services.

In medicine, clustering can be used to characterise patient behaviour, based on similar characteristics, so as to identify successful medical therapies for different illnesses. In biology, clustering is used to group genes with similar expression patterns or to cluster genetic markers to identify family ties. If you look around, you can find many other applications of clustering, but generally, it can be used for one of the following purposes: exploratory data analysis, summary generation or reducing the scale, outlier detection (especially to be used for fraud detection or noise removal), finding duplicates and data sets; or as a pre-processing step for either prediction, other data mining tasks or as part of a complex system.

## **A brief look at different clustering algorithms and their characteristics**

Partitioned based clustering is a group of clustering algorithms that produces clusters, such as k-means, k-medians or fuzzy c-means. These algorithms are quite efficient and are used for medium and large sized databases.

Hierarchical clustering algorithms produce trees of clusters, such as agglomerative and divisive algorithms. These groups of algorithms are very intuitive and are generally good with small-sized data sets. Density based clustering algorithms produce arbitrary shaped clusters. They are especially good when dealing with spatial clusters or when there is noise in your data set — for example, the DBSCAN algorithm.

In this article, I will take you through the types of clustering, the different clustering algorithms and compare two of the most commonly used clustering methods.

### **k-means clustering**

k-means is probably the most well-known clustering algorithm. It divides data into non-overlapping subsets (clusters) without any cluster internal structure. The objective of k-means is to form clusters in such a way that similar samples go into a cluster and dissimilar samples fall into different clusters. So we can say k-means tries to minimise the intra-cluster distances and maximise the inter-cluster distances.

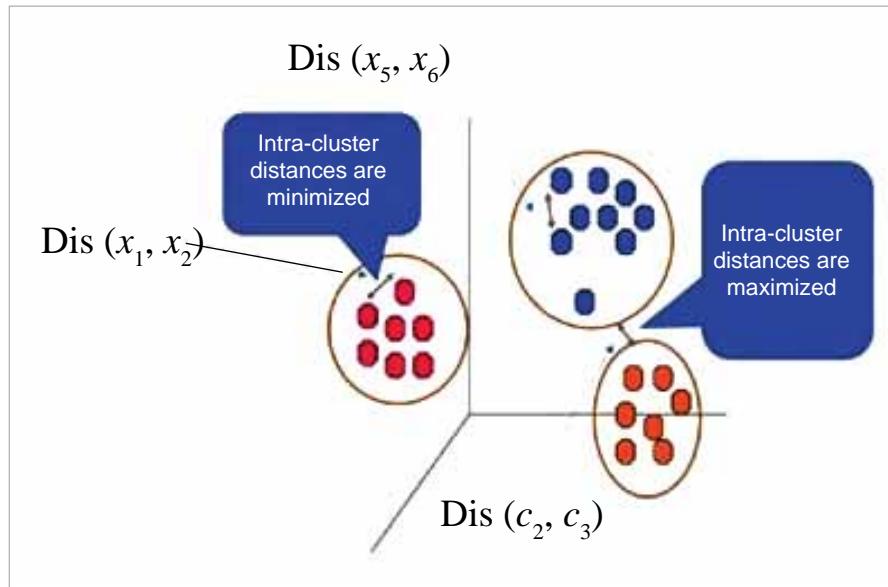


Figure 1: k-means clustering

$$\text{Dis}(x_1, x_2) = \sqrt{\sum_{i=0}^n (x_{1i} - x_{2i})^2}$$

Figure 2: Euclidean distance

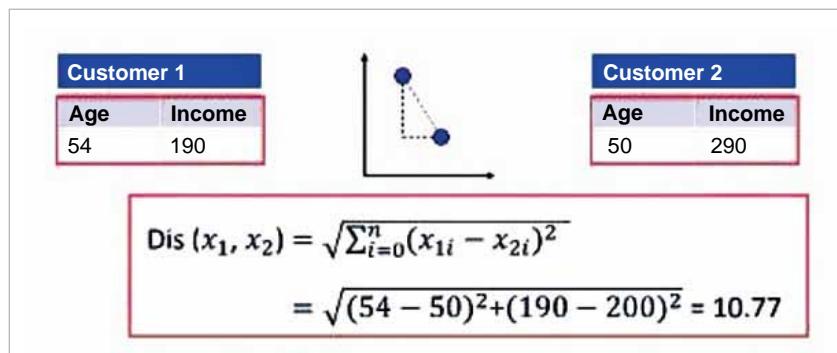


Figure 3: Euclidean distance in 2D

Now the question is about how to calculate the dissimilarity or distance of two cases such as two customers. We can easily use a specific type of Minkowski distance to calculate the distance of these two customers. Indeed, it is the Euclidean distance.

```
In [5]: from sklearn.cluster import KMeans
Kmean = KMeans(n_clusters=2)
Kmean.fit(X)

Out[5]: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
random_state=None, tol=0.0001, verbose=0)

In [6]: Kmean.cluster_centers_
Out[6]: array([[-1.13660621, -1.0979858 ],
 [ 2.06037688,  1.8179407 ]])
```

Figure 4: Centroid values

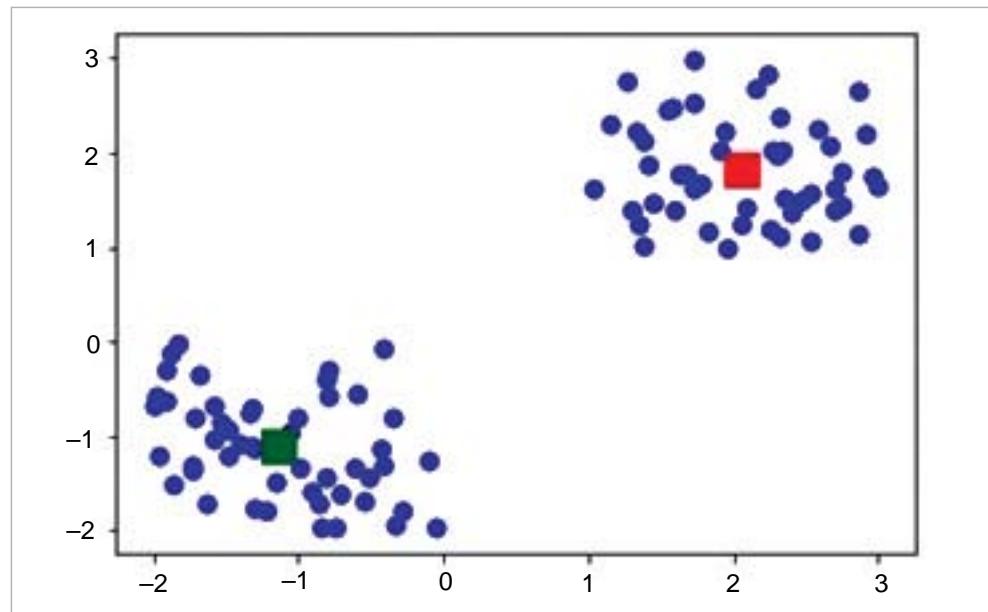


Figure 5: Displaying the cluster centroids (green and red colour)

And what if we have more than one feature? Well, we can still use the same formula, but this time, in a two-dimensional space.

The steps involved are listed below.

1. In the first step, we should determine the number of clusters. The key concept of the k-means algorithm is that it randomly picks a centre point for each cluster, which means that we must initialise 'k', which represents the number of clusters.
2. Then, the algorithm will randomly select the centroids of each cluster.

3. After the initialisation step, which defined the centroid of each cluster, we have to assign each point to the closest centre. For this purpose, we have to calculate the distance of each data point or in our case, each customer, from the centroid points, which will be assigned for each data point to the closest centroid (using Euclidean distance).

It is safe to say that this does not result in good clusters because the centroids were chosen randomly from the beginning. Indeed, the model will probably have a high error rate. Here, the error is the total distance of each point from its centroid.

4. So how can we turn them into better clusters with less error? We move the centroids. The main objective of k-means clustering is to minimise the distance of data points from the centroid of this cluster and maximise the distance from other cluster centroids. So in this step, we have to find the closest centroid to each data point.

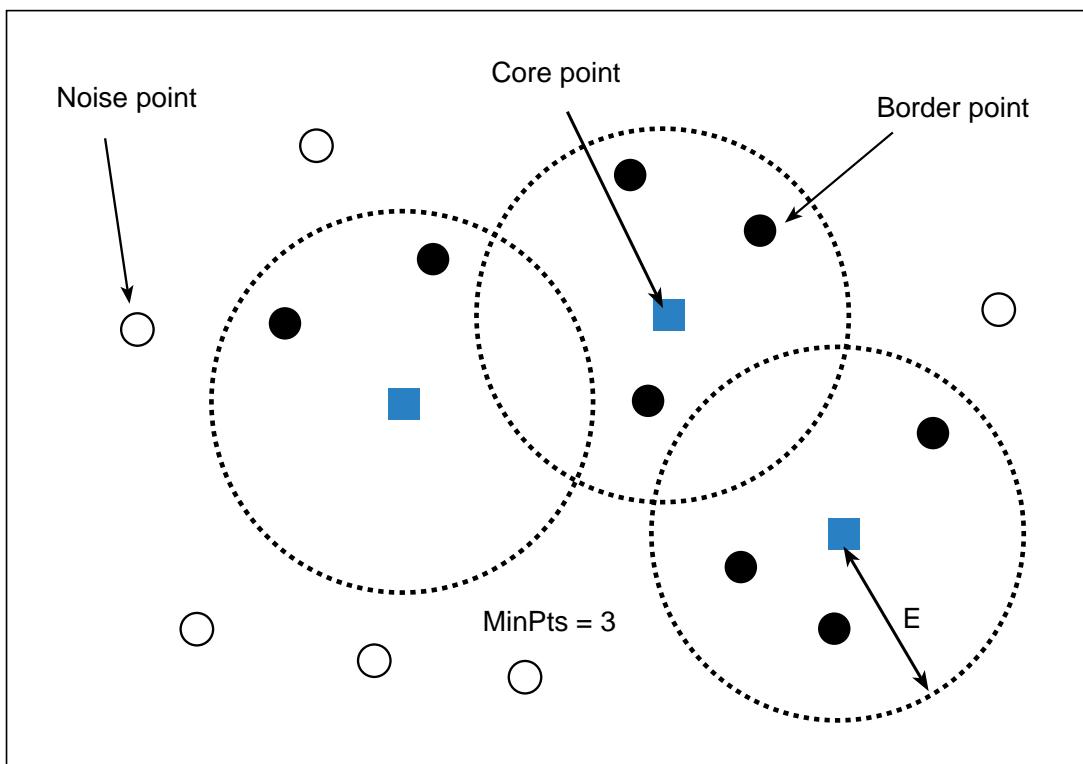


Figure 6: Data points

**5. Re-compute cluster centroids:** The new centroids will be calculated as the mean of the points that belong to the centroid of the previous step.

In a nutshell, the steps involved in the k-means algorithm are:

1. Randomly place 'n' centroids, one for each cluster.
2. Calculate the distance of each data point from each centroid.
3. Assign each data point to its closest centroid, creating a cluster.
4. Recalculate the position of centroids.
5. Since k-means is an iterative algorithm, repeat Steps 2 to 4 until it results in the clusters with minimum error or the centroids no longer move.

## Implementing the k-means algorithm

Initially, import the following libraries — Pandas, Numpy and Matplotlib. Next, generate some random data in a two-dimensional space. Here, a 100 data points have been generated and divided into two groups of 50 points each.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
X= -2 * np.random.rand(100,2)
X1 = 1 + 2 * np.random.rand(50,2)
X[50:100, :] = X1
plt.scatter(X[ :, 0], X[ :, 1], s = 50, c = 'b')
plt.show()
#In this case, we arbitrarily gave k (no of clusters) a value of two.
#Next, we need to find the centre of the clusters, centroid by using Kmean.
cluster_centres

from sklearn.cluster import KMeans
Kmean = KMeans(n_clusters=2)
Kmean.fit(X)
Kmean.cluster_centers_
```

```
#Hence we get the value of centroids, -1.13660621, -1.0979858 and 2.06037688,  
1.8179407  
plt.scatter(X[ :, 0], X[ :, 1], s =50, c='b')  
plt.scatter(-0.94665068, -0.97138368, s=200, c='y', marker='s')  
plt.scatter(2.01559419, 2.02597093, s=200, c='r', marker='s')  
plt.show()
```

The k-means algorithm identifies 'k' number of centroids and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible. The 'means' in k-means refers to averaging of the data, i.e., finding the centroid.

The correct choice of 'k' is often ambiguous because it's very dependent on the shape and scale of the distribution of points in a data set.

The elbow method is used for determining the correct number of clusters in a data set. It works by plotting the ascending values of 'k' versus the total error obtained when using that 'k'. This means increasing 'k' will always decrease the error. So, the value of the metric as a function of 'k' is plotted and the elbow point is determined where the rate at which the error decreases shifts sharply. It is the right 'k' for clustering.

So, k-means is partition based clustering, which is relatively efficient on medium and large sized data sets. It produces sphere-like clusters because the clusters are shaped around the centroids. Its drawback is that we need to pre-specify the number of clusters, which is not going to be an easy task.

## Density based clustering algorithms

Most traditional clustering techniques such as k-means, hierarchical and fuzzy clustering can be used to group data in an unsupervised way. However, when applied to tasks with arbitrary shaped clusters or to clusters within clusters, traditional techniques might not be able to achieve good results, i.e., elements in the same cluster might not share enough similarity or the performance may be poor.

In contrast, density based clustering locates regions of high density that are separated from one another by regions of low density. Density in this context is defined as the number of points within a specified radius.

DBSCAN (Density Based Spatial Clustering of Applications with Noise) is a specific and very popular type of density based clustering, which is particularly effective for tasks like class identification in a spatial context, and it can find out any arbitrary shaped cluster without getting affected by noise. Also, it is efficient for large databases.

## How DBSCAN works

DBSCAN works on the idea that if a particular point belongs to a cluster, it should be near lots of other points in that cluster. It works based on two parameters — radius and minimum points. R or 'eps' determines a specified radius which, if it includes enough points within it, is called a dense area. M determines the minimum number of data points we want in a neighbourhood to define a cluster. As a general rule, the minimum M can be derived from the number of dimensions D in the data set as,  $M \geq D+1$ . The minimum value of M that can be chosen is 3.

In this algorithm, we have the following three types of data points.

**Core point:** A point is a core point if it has more than one minimum point, M, within eps.

**Border point:** This is a point which has fewer than the minimum points within the radius (eps) but is in the neighbourhood of a core point.

**Noise or outlier:** A point that is not a core or border point. Epsilon  $\epsilon$  or, in short, eps, indicates the radius.

The DBSCAN algorithm can be abstracted in the following steps:

1. The algorithm starts with an arbitrary point which has not been visited. And if this point contains MinPts within the neighbourhood, cluster formation

- starts. Otherwise the point is labelled as noise.
2. For each core point, if it is not already assigned to a cluster, create a new cluster.
  3. Identify and assign border points to their respective core points.
  4. Iterate through the remaining unvisited points in the data set. Points that do not belong to any cluster are noise.

The main drawback of DBSCAN is that it doesn't perform as well as others when the clusters are of varying density. This is because the setting of the distance threshold  $\epsilon$  and MinPts for identifying the neighbourhood points varies from cluster to cluster, when the density varies. If the data and features are not so well understood by a domain expert, then setting up and MinPts could be tricky. Meanwhile, the DBSCAN algorithm overcomes the drawbacks of k-means, such as k-means forms spherical clusters only, is sensitive towards outliers and requires one to specify the number of clusters.

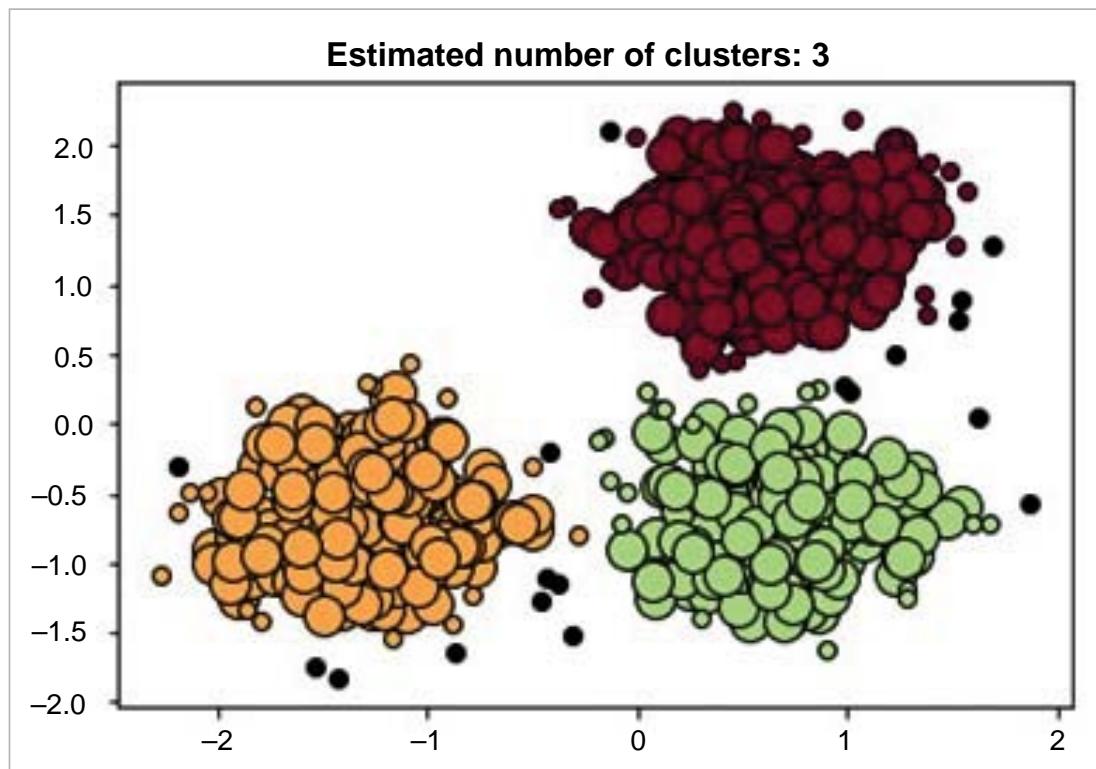


Figure 7: Results of the DBSCAN clustering algorithm

The DBSCAN clustering algorithm is demonstrated below:

```
import numpy as np
from sklearn.cluster import DBSCAN
from sklearn import metrics
from sklearn.datasets.samples_generator import make_blobs
from sklearn.preprocessing import StandardScaler

#####
# Generate sample data
centers = [[1, 1], [-1, -1], [1, -1]]
X, labels_true = make_blobs(n_samples=750, centers=centers, cluster_std=0.4,
                            random_state=0)

X = StandardScaler().fit_transform(X)

#####
# Compute DBSCAN
db = DBSCAN(eps=0.3, min_samples=10).fit(X)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_

#####
# Number of clusters in labels, ignoring noise if present.
n_clusters = len(set(labels)) - (1 if -1 in labels else 0)
n_noise = list(labels).count(-1)

print('Estimated number of clusters: ', n_clusters)
print('Estimated number of noise points: ', n_noise)
print("Homogeneity: %0.3f" % metrics.homogeneity_score(labels_true, labels))
print("Completeness: %0.3f" % metrics.completeness_score(labels_true, labels))
print("V-measure: %0.3f" % metrics.v_measure_score(labels_true, labels))
```

```
print("Adjusted Rand Index: %0.3f" % metrics.adjusted_rand_score(labels_true,
labels))
print("Adjusted Mutual Information : %0.3f " %metrics.adjusted_mutual_info_
score
( labels_true, labels, average_method='arithmetic'))
print("Silhouette Coefficient: %0.3f" % metrics.silhouette_score(X, labels))

#####
# Plot result
import matplotlib.pyplot as plt

# Black removed and is used for noise instead.
unique_labels = set(labels)
colors = [plt.cm.Spectral(each)
          for each in np.linspace(0, 1, len(unique_labels))]
for k, col in zip(unique_labels, colors):
    if k == -1:
        col = [0, 0, 0, 1]
    class_member_mask = (labels == k)
    xy = X[class_member_mask & core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
              markeredgecolor='k', markersize=14)
    xy = X[class_member_mask & ~core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),markeredgecolo
r='k', markersize=6)

plt.title('Estimated number of clusters: %d' % n_clusters)
plt.show()
```

The output is:

```
Estimated number of clusters:  3
Estimated number of noise points: 18
```

Homogeneity: 0.953

Completeness: 0.883

V-measure: 0.917

Adjusted Rand Index: 0.952

Adjusted Mutual Information: 0.916

Silhouette Coefficient: 0.626

To wrap up, we went through some basic concepts of the k-means and DBSCAN algorithms. A direct comparison with k-means clustering has been made to better understand the differences between these algorithms. Basically, the DBSCAN algorithm overcomes the drawbacks of the k-means algorithm.

Clustering is an unsupervised machine learning approach, but can be used to improve the accuracy of supervised machine learning algorithms. Hopefully, this will help you to get started with one of the most used clustering algorithms for unsupervised problems.

# **Using OpenCV for Machine Learning in Real-time Computer Vision and Image Processing**

*Catch up with Open Source Computer Vision (OpenCV), a computer vision and machine learning software library. As it is published under the BSD licence, you are free to develop and modify the source code.*

Computer vision and digital image processing are currently being widely applied in face recognition, biometric validations, the Internet of Things (IoT), criminal investigation, signature pattern detection in banking, digital documents analysis, smart tag based vehicles for recognition at toll plazas, etc. All these applications use image and real-time video processing so that the live capture of multimedia impressions can be made for detailed analysis and predictions.

Computer vision is widely integrated in different applications including 2D and 3D image analytics, egomotion estimation, feature points detection, human-computer interaction (HCI), face recognition systems and mobile robotics. Computer vision is also applied in the fields of gesture recognition, object identification, segmentation recognition, motion understanding, stereopsis stereo vision, motion tracking, structure from motion (SFM), pattern recognition, augmented reality, decision making, scene reconstruction, etc.

The following are the key research areas in computer vision and image analytics:

- Deep neural networks for biometric evaluation
- Facial sentiment analysis and emotion recognition

- Real-time video analysis and recognition of key features
- Gesture recognition
- Visual representation learning
- Face smile detection
- Vein investigation and training for biometrics

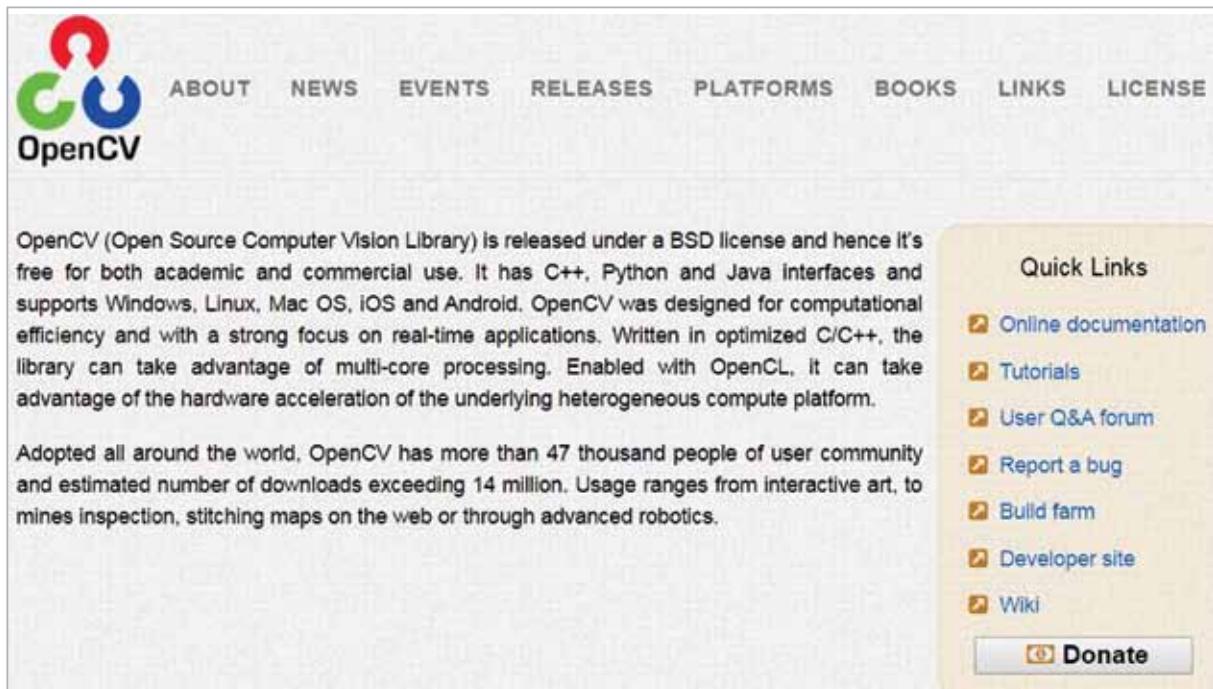


Figure 1: Official portal of OpenCV

Table 1: Tools for computer vision

Tool	URL
OpenCV	<a href="http://www.opencv.org/">http://www.opencv.org/</a>
BoofCV	<a href="http://www.boofcv.org/">http://www.boofcv.org/</a>
NASA Vision Workbench	<a href="http://ti.arc.nasa.gov/tech/asr/intelligent-robotics/nasa-vision-workbench/">http://ti.arc.nasa.gov/tech/asr/intelligent-robotics/nasa-vision-workbench/</a>
SimpleCV	<a href="http://simplecv.org/">http://simplecv.org/</a>
Tesseract	<a href="https://github.com/tesseract-ocr/tesseract">https://github.com/tesseract-ocr/tesseract</a>
SLIC Superpixels	<a href="http://ivrl.epfl.ch/supplementary_material/RK_SLICSuperpixels/">http://ivrl.epfl.ch/supplementary_material/RK_SLICSuperpixels/</a>
OpenMVG	<a href="https://github.com/openMVG/openMVG">https://github.com/openMVG/openMVG</a>
LIBVISO	<a href="http://www.cvlabs.net/software/libviso/">http://www.cvlabs.net/software/libviso/</a>
VisualSFM	<a href="http://homes.cs.washington.edu/~ccwu/vsfm/">http://homes.cs.washington.edu/~ccwu/vsfm/</a>
MeshLab	<a href="http://meshlab.sourceforge.net/">http://meshlab.sourceforge.net/</a>
Bundler	<a href="http://phototour.cs.washington.edu/bundler/">http://phototour.cs.washington.edu/bundler/</a>
Vid.stab	<a href="https://github.com/georgmartius/vid.stab">https://github.com/georgmartius/vid.stab</a>
ViSP	<a href="https://visp.inria.fr/">https://visp.inria.fr/</a>

- Multi-resolution approaches
- Radiomics analytics for medical data sets
- Forgery detection in digital images
- Content based image retrieval
- Automatic enhancement of images
- Identification and classification of objects in real-time
- Defects prediction in manufacturing lines using live images of machines
- Industrial robots with real-time vision for disaster management
- Image reconstruction and restoration
- Computational photography
- Morphological image processing
- Animate vision
- Photogrammetry

## Prominent tools for computer vision and image analytics

There are a number of tools and libraries available to implement computer vision and image analytics. Table 1 lists these.

### OpenCV

OpenCV (<http://opensource.org>) is a high performance library for digital image processing and computer vision, which is free and open source. It is equipped with a large set of functions and algorithms for real-time computer vision and predictive mining. OpenCV was devised and developed by Intel, and the current instances are supported by W. Garage and Itseez. It was developed so that real-time analytics of images and recognition can be done for assorted applications.

The statistical machine learning libraries used by OpenCV are:

- Deep neural networks (DNN)
- Convolutional neural networks (CNN)
- Boosting (meta-algorithm)
- Decision tree learning
- Gradient boosting trees
- Expectation-maximisation algorithm

- K-nearest neighbour algorithm
- Naive Bayes classifier
- Artificial neural networks
- Random forest
- Support vector machine (SVM)

## Installation of OpenCV

The installation of OpenCV can be done for different programming languages including Python, Java, C++ and many others. OpenCV provides open connectivity with other platforms and programming languages,



Figure 2: Scanned image of the original cheque



Figure 3: Signature used in another document by copying

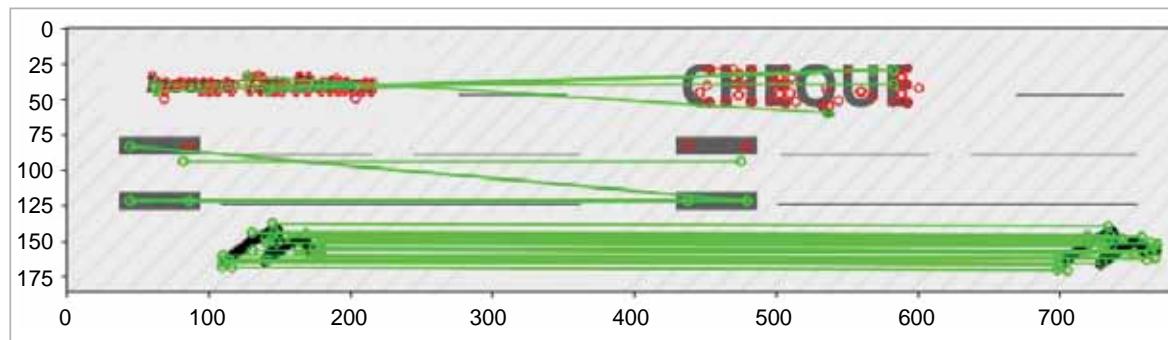


Figure 4: Identification of forged pixels in a cheque using OpenCV

so that the algorithms of computer vision can be implemented without any concerns about compatibility or dependencies. It can be installed on different platforms including Android, Windows, Linux, BlackBerry, FreeBSD, OpenBSD, iOS, Maemo or OS X.

OpenCV has open connectivity for installation with different software suites. To integrate OpenCV with Scilab programming, use the following command:

```
--> atomsInstall("scicv")
--> atomsInstall("IPCV")
```

...or:

```
--> atomsInstall("Source Zip File")
```

To install OpenCV for Python, use the following code:

```
$ pip install opencv-python
$ pip install opencv-contrib-python
```

## Real-time image capturing from a Web cam using OpenCV

In traditional implementations, the feature points of the images and computer vision files are recognised on the pre-saved disk images. This approach can be further enhanced using OpenCV, when the real-time video can be marked with the feature points or key points of the image frame in a live running video. The following is the source code of OpenCV that is executed in the Python environment. It can be used to implement real-time recognition of live feature points from a video shot on a Web cam.

```
import numpy as mynp
import cv2
cap = cv2.VideoCapture(0)
while(True):
    ret, frame = cap.read()
    MyGray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    MyGray = mynp.float32(MyGray)
    dst = cv2.cornerHarris(MyGray, 2, 3, 0.04)
```

```
tdst = cv2.dilate(tdst,None)
frame[tdst>0.01*tdst.max()]=[0,0,255]
cv2.imshow('tdst',frame)
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
cap.release()
cv2.destroyAllWindows()
```

## **Identification of image forgery using OpenCV**

With the emergence and implementation of digital authentication in assorted applications, the identification of original user impressions is one of the key challenges today. Nowadays, many organisations ask for documents and certificates signed and scanned by applicants for self-attestation. This also happens when applying for new mobile SIM cards. However, imposters can take scanned signatures from any other document and put them in the required document using digital image editing tools like Adobe Photoshop, PaintBrush or PhotoEditors. There are a number of image editing tools available for the transformation of an actual image into a new image. This type of implementation can be (and often is) used for creating the new forged copy of the document, without the knowledge or permission of the actual applicant. Figures 2 and 3 depict the process of forgery detection in a new document where the signatures are copied from another source.

The following source code written in Python and OpenCV presents the implementation of Flann based evaluation of images. OpenCV has enormous algorithms for the extraction of features in the images as well as in videos. Using such approaches, the manipulation in captured files can be identified and plotted.

```
import numpy as mynp
import cv2
from matplotlib import pyplot as plt
MIN_COUNT = 10
```

```

myimage1 = cv2.imread('ChequewithCopiedSignature.png',0)
myimage2 = cv2.imread('OriginalCheque.png',0)
sift = cv2.xfeatures2d.SIFT_create()
# find the keypoints and descriptors with SIFT
k1, im1 = sift.detectAndCompute(myimage1,None)
k2, im2 = sift.detectAndCompute(myimage2,None)
FLANN_INDEX_KDTREE = 0
index_params = dict(algorithm = FLANN_INDEX_KDTREE, trees = 5)
search_params = dict(checks = 50)
flann = cv2.FlannBasedMatcher(index_params, search_params)
matches = flann.knnMatch(im1,im2,k=2)
# store all the good matches as per Lowe's ratio test.
good = []
for m,n in matches:
    if m.distance < 0.7*n.distance:
        good.append(m)
if len(good)>MIN_COUNT:
    src_pts = mynp.float32([ k1[m.queryIdx].pt for m in good ]).reshape(-1,1,2)
    dst_pts = mynp.float32([ k2[m.trainIdx].pt for m in good ]).reshape(-1,1,2)
    M, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC,5.0)
    matchesMask = mask.ravel().tolist()
    h,w = myimage1.shape
    pts = mynp.float32([ [0,0],[0,h-1],[w-1,h-1],[w-1,0] ]).reshape(-1,1,2)
    dst = cv2.perspectiveTransform(pts,M)
    myimage2 = cv2.polylines(myimage2,[mypn.int32(dst)],True,255,3, cv2.LINE_AA)
else:
    print "Not enough matches are found - %d/%d" % (len(good),MIN_COUNT)
    matchesMask = None
draw_params = dict(matchColor = (0,255,0), # draw matches in green color
                   singlePointColor = None,
                   matchesMask = matchesMask, # draw only inliers
                   flags = 2)

```

```
myimg9 = cv2.drawMatches(myimage1,k1,myimage2,k2,good,None,**draw_params)
plt.imshow(myimg9, 'gray'),plt.show()
```

It is evident from Figure 4 that the pixels have been identified in the new image (in which the signature has been copied from another source). So the marking of pixel values can be done using machine learning and inbuilt methods for prediction in OpenCV. Even if the person makes use of highly effective tools for image editing, the pixels from where the image segment is taken can be identified.

# The Python Libraries that are Made for Machine Learning

*This article is a guide on Python libraries for machine learning. Various libraries as well as their features are described to help readers understand them better, and enable them to then explore more advanced features.*

Python is an excellent environment for building any machine learning (ML) application or system. ML involves data analysis, data engineering, feature extraction and building types of features, while Python has built-in capabilities for data analysis and data engineering. In addition, Python is very popular and is one of the *de-facto* programming languages as compared to Ruby, Perl or R. As Python provides the following advantages, one can definitely opt for it when building any ML based models or applications:

- It provides good libraries for data analysis.
- It provides an interactive computing feature including IPython and Jupyter Notebooks.
- It provides data visualisation features with multiple options to choose from.
- It provides a variety of libraries, packages and tools for data analysis.
- Python has good support for Pandas, Numpy, Matplotlib, SciPy and Scikit-learn libraries that are among the most used in ML.
- Python also provides scientific computing features including support for popular languages like C, C++ and FORTRAN.
- Python has inbuilt capabilities for doing research and prototyping.
- As Python supports multiple languages, building production environments is very easy.

In this article, I will start with introducing the various Python libraries, one by one. This will help in better understanding how ML models/applications can

be built using these. In addition, where possible, I will give examples with small code snippets. I have used the Anaconda distribution for Windows (<https://www.anaconda.com/distribution/>) for all my code examples here. Anaconda has inbuilt libraries and packages in its downloadable version, as shown in Figures 1 and 2.

## NumPy

NumPy is the short form for Numerical Python, which is used for all numerical based calculations. By default, it provides data structures, algorithms and libraries for many of the applications involving numerical data and calculations in Python.

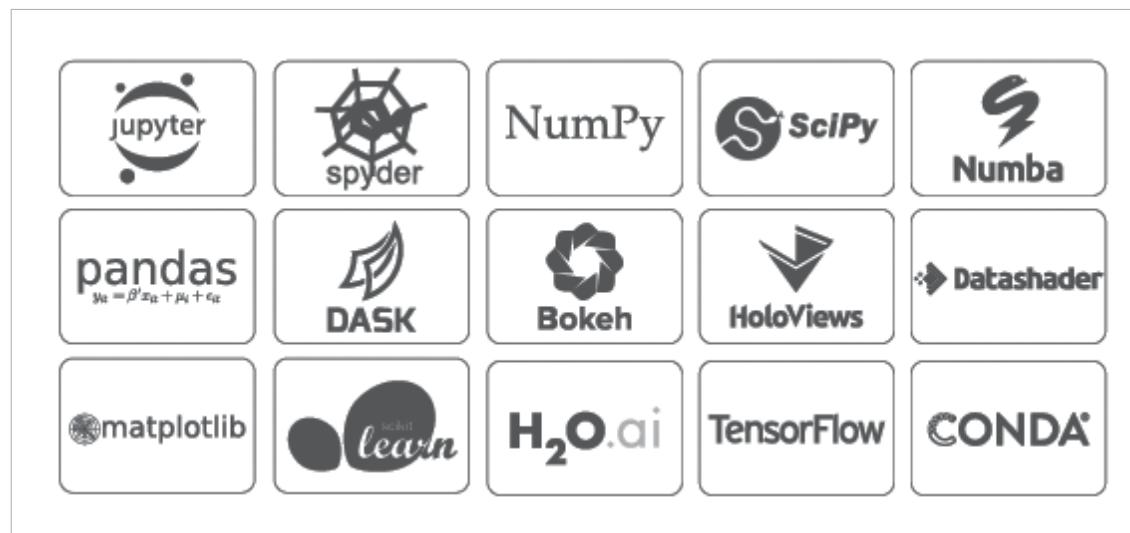


Figure 1: Anaconda distribution-supported features-libraries

```
(base) C:\Users\██████████>jupyter notebook --version
5.6.0

(base) C:\Users\██████████>python --version
Python 3.5.6 :: Anaconda custom (64-bit)

(base) C:\Users\██████████>
```

Figure 2: Jupyter Anaconda version

NumPy has a number of advantages as listed below:

- It is very fast and efficient as its engine is built using C code.
- It supports multi-dimensional arrays (using *ndarray* object).
- It has built-in functions to perform various mathematical operations and computations for arrays.
- All array based operations can be done very efficiently.
- It supports array operations like reading and writing data from disk, files and data repos.
- It supports C/C++ based APIs for extended Python for Numpy's data structures and computations.
- All the following mathematical operations are supported – random number generation, Fourier transforms, linear algebra and matrix operations.

To import and start using the NumPy library, use the following command:

```
import numpy as np
```

## Pandas

Pandas is one of the most popular Python libraries for data manipulation and data cleaning. Often, Pandas is used with NumPy and SciPy along with analytical libraries to build data visualisation. Pandas has the following built-in capabilities:

- It supports the matplotlib library.
- It supports Scikit-learn.
- It supports array based functions and computing.
- In addition to NumPy, Pandas is used for homogeneous numerical array data types.

**Pandas Series:** A Series is a one-dimensional array. This is almost similar to an object containing a sequence of values (only similar data types are allowed) along with the index having its labels.

In the following example, there are four integer data types in the array 'my\_data' and this data type is shown below as 'int64'.

```
1 my_data = pd.Series([1,9,-5,18])  
1 my_data
```

```
0    1  
1    9  
2   -5  
3   18  
dtype: int64
```

Various data operations can be performed on Series objects. Just as we can have our own index labels, math operations like *greater than*, *less than*, *equal to*, etc, can be done.

```
1 my_data2 = pd.Series([1,9,-5,18], index = ['q', 'w', 'e', 'r'])  
my_data2
```

```
q 1  
w 9  
e -5  
r 18  
my_data2['w']  
9  
my_data2['r'] = 98  
my_data2  
q 1  
w 9  
e -5  
r 98  
dtype: int64
```

```
2 my_data2[my_data2 < 100  
q 1  
w 9
```

```
e -5
r 98
dtype: int64
1 my_data2 = [my_data2 **2]
1 my_data2
[q 1
w 81
e 25
r 324
dtype: int64
```

**Pandas DataFrames:** A DataFrame in Pandas contains an ordered collection of columns. Each can be of a different value type (Boolean, numeric, string or integer). Unlike a Series object, DataFrame will have both row and column indexes. The data will be stored in the one- or two-dimensional array format. To import and start using the Pandas library, use the following set of commands and steps:

```
Import pandas as pd
print(my_list)
['shashi', 'dhar', 'soppin', 'is', 'writing', 'this', 'article']
df = pd.DataFrame(my_list)
```

```
df
0
0 shashi
1 dhar
2 soppin
3 is
4 writing
5 this
6 article
```

## SciPy

This is a fundamental library for scientific computing. This library provides many numerical routines that are efficient. There are some routines for numerical integration, optimisation, linear algebra and statistical purposes.

## SciKit-learn

SciKit-learn is one of the most used ML libraries in Python. It provides simple and efficient tools for carrying out data analysis with data mining related features. The following are the major groups of algorithms that are supported in the SciKit-learn library:

- Classification
- Regression
- Dimensionality reduction
- Model selection
- Preprocessing
- Clustering

# Machine Learning: Building a Predictive Model with Scikit-learn

*Scikit-learn was initially developed by David Cournapeau as a Google Summer of Code project in 2007. The project now has more than 30 active contributors with paid support from Inria, Google, Tinyclues and the Python Software Foundation. Scikit-learn provides algorithms for supervised and unsupervised learning, using a Python interface.*

Machine learning is one of the branches of computer science in which algorithms (running inside computers) learn from the data available to them. With this learning mechanism, various predictive models can be arrived at. The machine learning systems (or algorithms) can be broadly classified into many categories, based on various factors and methods. The most popular of these are:

- Supervised learning
- Unsupervised learning
- Semi-supervised learning
- Reinforcement learning
- Online/batch learning

**Supervised learning:** In supervised learning, the data we input to these algorithms also includes the expected solution, which is called labels. The supervised learning algorithm consists of sets of variables called dependent and independent variables. Dependent variables are derived from the target or outcome and independent variables are predictors. Supervised learning occurs when the machine is taught or trained using

data that is well labelled (which means that some data is already tagged with the correct answer or classification). Once this is done, the machine is provided with a new set of data or examples, so that this algorithm analyses the training data (set of training examples or data sets) and produces the desired outcome from labelled data.

The most popular and talked about supervised learning algorithm is the ‘Spam or Ham’ classification which is used for spam email classification. Another popular algorithm relates to fruit classification — if the shape of an object is round with a depression at its top and the colour is red, then it will probably be labelled as ‘apple’. And if the shape of an object is a long, curved cylinder that is green-yellow in colour, then it will probably be labelled as ‘banana’.

Some of the most important and heavily used supervised learning algorithms are:

- k-Nearest neighbours
- Linear regression
- Logistic regression
- Support vector machines (SVMs)
- Decision trees and random forests
- Neural nets (also called NNs)

***Unsupervised learning:*** Here, the training data is unlabelled. It is similar to a system trying to learn without a teacher. In unsupervised learning, the training of the machine or system is done using the information that is neither classified nor labelled. Because of this, the algorithm itself will act on information without any guidance. Here, the machine with this unsorted information tries to recognise similarities, patterns and differences, without any training.

Some of the most important and extensively used unsupervised algorithms are listed below.

*Clustering:* These classify customers or buyers based on their purchasing patterns:

- K-Means
- Hierarchical cluster analysis (HCA)
- Expectation maximisation

*Visualisation and dimensionality reduction:*

- Principal component analysis
- Kernel PCA
- Locally-linear embedding (LLE)
- T-distributed Stochastic neighbour embedding (t-SNE)

*Association rule learning:* These set rules based on certain recognisable patterns of behaviour, like the customer who buys product X may also buy product Y.

- Apriori
- Eclat

**Semi-supervised learning:** Some algorithms can deal with partially labelled data and much unlabelled data; or sometimes, with a little bit of labelled data. These kinds of algorithms are called semi-supervised learning algorithms.

A photo hosting service like Google Photos is a good example of semi-supervised learning.

**Reinforcement learning:** RL or reinforcement learning is an area of ML inspired mainly by behavioural psychology. It is totally different from supervised learning. In this algorithm, an agent learns how to behave in an environment by performing actions and seeing the results.  
Examples include Google's DeepMind and AlphaGo.

**Batch and online learning:** Online and batch processing are slightly different, although both perform well for parabolic performance surfaces.

One major difference between them is that the batch algorithm keeps the system weights constant while computing the error associated with each sample in the input. Since the online version is constantly updating its weights, its error calculation (and thus gradient estimation) uses different weights for each input sample. This means that the two algorithms visit different sets of points during adaptation. However, they both converge to the same minimum.

## **Scikit-learn**

In this article, we will primarily explore the Scikit library for machine learning purposes. Scikit is open source and released under the BSD licence. Most parts of the Scikit library are written in Python and some of its core algorithms are written in Cython (for efficiency and performance).

Scikit-learn's main purpose is to build models; so this library is not well suited for other activities like reading, manipulation of data or the summarisation of data.

The benefits of Scikit-learn are:

- It is one of the most consistent interfaces available today to build ML models. It provides the user with many tuning parameters.
- Many of the good functionalities are in-built in this library.
- It also has good documentation that is easy to understand, and has active community support.
- It covers most of the well-known machine learning models

However, it does have some drawbacks:

- Many geeks mention that in the beginning, Scikit-learn is somewhat harder to learn, compared to R.
- There is less emphasis on model interpretability.

Scikit-learn is a machine learning library for Python. It features several regression, classification and clustering algorithms including SVMs, gradient boosting, k-means, random forests, etc. I have given a brief overview of these algorithms earlier, keeping this in mind. This library is also designed to work

with other Python based libraries like that of NumPy and SciPy.

A combination of these three libraries is well suited for many machine learning models.

## Installation

The installation process of Scikit is easy, and it is assumed that the latest version of the NumPy (1.8.2 or above) and SciPy (0.13.3 or above) library packages are already preinstalled and supported in your systems. Once these prior dependent packages are installed, the installation of the Scikit library can proceed.

 **Note:** It is assumed that Python 3.3 or a later version is used for these models.

Depending on the framework or Python environment in use, Scikit can be installed with the following set of commands. For Pip installation, give the following command:

```
ubuntu@ubuntu:~$ pip install scikit-learn
```

If the Anaconda package is in use, then use the following command (this step varies from the Conda prompt based on Windows, Mac or Linux platforms). Use the IPython terminal (now called Jupyter):

```
# conda install scikit-learn
```

Once the installation is done, the Scikit library can be used to build and predict models in Jupyter or other frameworks of choice. I have used Jupyter for all the code examples.

## Building a model

Follow the steps shown below to build a model.

**Step 1:** For simplicity and ease of demonstration, the iris data set from *kaggle.com* is used here. Kaggle is one of the most popular websites and hosts a lot of data for machine learning model-building (<https://www.kaggle.com/jchen2186/machine-learning-with-iris-dataset>).

This data is a classification of the iris flower along with all its varieties, which are classified based on the length and width of their petals and sepals.

- There are 150 observations with four features each (sepal length, sepal width, petal length and petal width).
- There are no null values, so we don't have to worry about that.
- There are 50 observations for each species (setosa, versicolor, virginica, etc).
- The response variable is the iris species.
- This is a typical classification problem, as the response is categorical in nature.
- Import the iris data set from the archive <https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data>
- Load the 'sklearn' library.
- Load the data set to 'sklearn' library.

Please refer to Figure 1.

**Step 2:** Once the available data is collected, the algorithm to be chosen has to be decided. In classifying the iris flower, the KNN K-Nearest Neighbour algorithm will be used.

The following are the requirements to work with the 'scikit-learn' library:

- Feature and response are separate objects
- Feature and response should be numeric
- Feature and response should be NumPy arrays
- Feature and response should have specific shapes

Variable 'x' is a two-dimensional array and variable 'y' is single dimensional.

```
In [1]: from IPython.display import HTML
HTML('<iframe src = https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.dat width =300, height = 200></iframe>')
```

Out[1]:

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
```

```
In [2]: from sklearn.datasets import load_iris
```

```
In [3]: iris = load_iris()
type(iris)
```

Out[3]: sklearn.datasets.base.Bunch

```
In [4]: print(iris.data)
```

```
[[ 5.1  3.5  1.4  0.2]
 [ 4.9  3.  1.4  0.2]
 [ 4.7  3.2  1.3  0.2]
 [ 4.6  3.1  1.5  0.2]
 [ 5.  3.6  1.4  0.2]]
```

Figure 1: Code for Jupyter notebook 1

Please refer to Figure 2.

**Step 3:** As described earlier, the KNN algorithm has been selected and this involves the following steps:

```
In [5]: print (iris.target_names)
['setosa' 'versicolor' 'virginica']

In [6]: print (type(iris.data))
<class 'numpy.ndarray'>

In [8]: print (type(iris.target))
<class 'numpy.ndarray'>

In [9]: print (iris.data.shape)
(150, 4)

In [10]: print (iris.target.shape)
(150,)

In [11]: X = iris.data
y= iris.target
```

Figure 2: Code for Jupyter notebook 2

```
In [12]: from sklearn.neighbors import KNeighborsClassifier

In [13]: knn = KNeighborsClassifier(n_neighbors = 1)

In [14]: print (knn)
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=1, n_neighbors=1, p=2,
                     weights='uniform')

In [15]: knn.fit(X,y)

Out[15]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                             metric_params=None, n_jobs=1, n_neighbors=1, p=2,
                             weights='uniform')

In [16]: knn.predict([3,5,4,2])
C:\Users\Sony\Anaconda3\lib\site-packages\sklearn\utils\validation.py:395: DeprecationWarning: Passing 1d arrays as data is deprecated in 0.17 and will raise ValueError in 0.19. Reshape your data either using X.reshape(-1, 1) if your data has a single feature or X.reshape(1, -1) if it contains a single sample.
DeprecationWarning)
```

Figure 3: Code for Jupyter notebook 3

- Pick a value for 'k'.
- Search for the k-observation in the mentioned training data.
- Use the most popular response value from the k-nearest neighbour.

Please refer to Figure 3.

The following are the important steps and measures that are to be taken, while coding this:

- Import the 'estimator'.
- Have the object ready and instantiate it (this is also called 'hyper parameter' ).
- Fit the model (with feature and response).
- Finally, predict the response from the observation.

Please see Figure 4.

```

Out[16]: array([2])

In [17]: x_new = [[3,5,4,2],[5,4,3,2]]
          knn.predict(x_new)

Out[17]: array([2, 1])

In [18]: from sklearn.linear_model import LogisticRegression
          logreg = LogisticRegression()
          # fit the model
          logreg.fit(X,y)
          #predict the response
          logreg.predict(x_new)

Out[18]: array([2, 0])

```

Figure 4: Code for Jupyter notebook 4

**Note:** The output of Line-16, which is 'array([2])' is the encoding for the 'virginica' for the unknown Iris. In the Jupyter notebook, the code in Line-30 "DeprecationWarning" can be ignored. In the Jupyter notebook, Output Line-17 'array([2,1])' is the classification for 'Iris\_setosa'.

Once this is done, check for the accuracy of the model. In this case, the model has provided an accuracy of 0.96, which is 96 per cent, and is considered good. Please see Figure 5.

```
In [19]: print(iris.data[2,0])  
4.7  
In [20]: # store the predicted response values  
  
y_pred = logreg.predict(X)  
  
# check how many predictions were generated  
len(y_pred)  
from sklearn import metrics  
  
print(metrics.accuracy_score(y,y_pred))
```

0.96

Figure 5: Code for Jupyter notebook 5

As stated earlier, Scikit-learn, along with the combination of SciPy and NumPy, is one of the best options available today for machine learning predictions.

# MLDB: The Open Source Machine Learning Database for the Cloud and for Docker

*This article talks about MLDB in relation to the cloud and Docker. It details the prominent tools for ML, describes the features of MLDB, its installation on different platforms, and the various algorithms supported by it.*

Machine learning and predictive analytics are key areas of research in multiple domains including bioinformatics, computational anatomy, natural language processing, speech recognition, etc. Machine learning is used to train the software or hardware applications based on specific models for rule mining, prediction and knowledge discovery. Data scientists and analysts implement different supervised or unsupervised approaches to get accuracy and performance from raw datasets.

Machine learning (ML) and artificial intelligence (AI) are very closely related, but these terms have different perspectives. Artificial intelligence (AI) is the concept by which machines perform tasks similar to human beings, and we call such machines 'smart'.

Machine learning is the recent advanced application area of AI in which machines learn by themselves based on dynamic inputs. Machine learning based implementations are more accurate and have high optimisation. The prominent tools for machine learning and deep learning are MLDB, Keras, Edward, Lime, Apache Singa and Shogun. The details for each can be found on their respective websites.

Machine Learning Database (MLDB) is a powerful and high performance database system specifically developed for machine learning, knowledge discovery and predictive analytics. MLDB is FOSS and is compatible with assorted platforms. It uses the RESTful API for the storage of data, exploring the data using Structured Query Language (SQL) and, finally, it trains machine learning models.

The following are the key features of MLDB that suit a range of applications.

**Speed:** The training, modelling and discovery process in MLDB is highly performance-aware. It has huge processing power compared to H2O, Scikit-Learn or Spark MLlib, which are prominent machine learning libraries.

**Scalability:** MLDB supports vertical scaling with higher efficiency, so all memory modules as well as cores can be used simultaneously without any issues of delay or performance.

**Free and open source:** The community edition of MLDB is available and distributed on a powerful repository and hosting of GitHub (<https://github.com/mldbai/mldb>).

**SQL support:** This makes MLDB very user-friendly along with the support for Big Data processing. MLDB can process, train and make predictions using database tables that have millions of columns, with concurrent processing and no compromise on integrity.

**Machine learning:** MLDB is developed for high performance machine learning applications and models. It has support for deep learning with the graphs of TensorFlow that make it superior in knowledge discovery.

**Ease of implementation:** There are installation packages for multiple platforms and programming environments including Jupyter, Docker, JSON, Cloud, Hadoop, and many others.

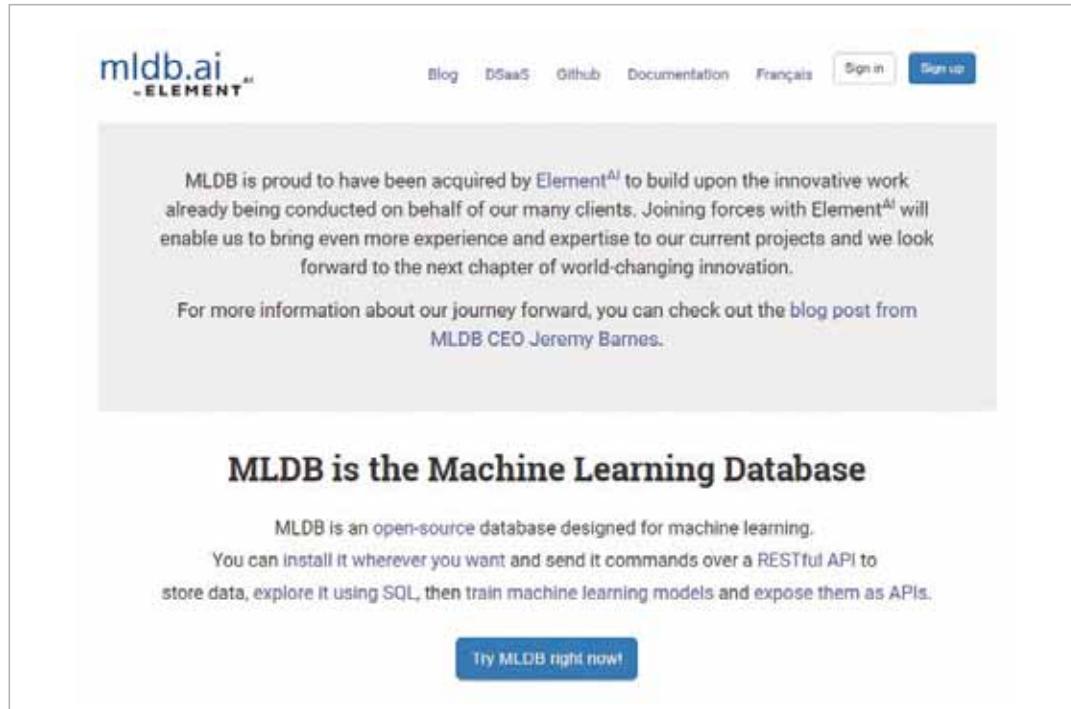


Figure 1: The official portal of Machine Learning Database (MLDB)

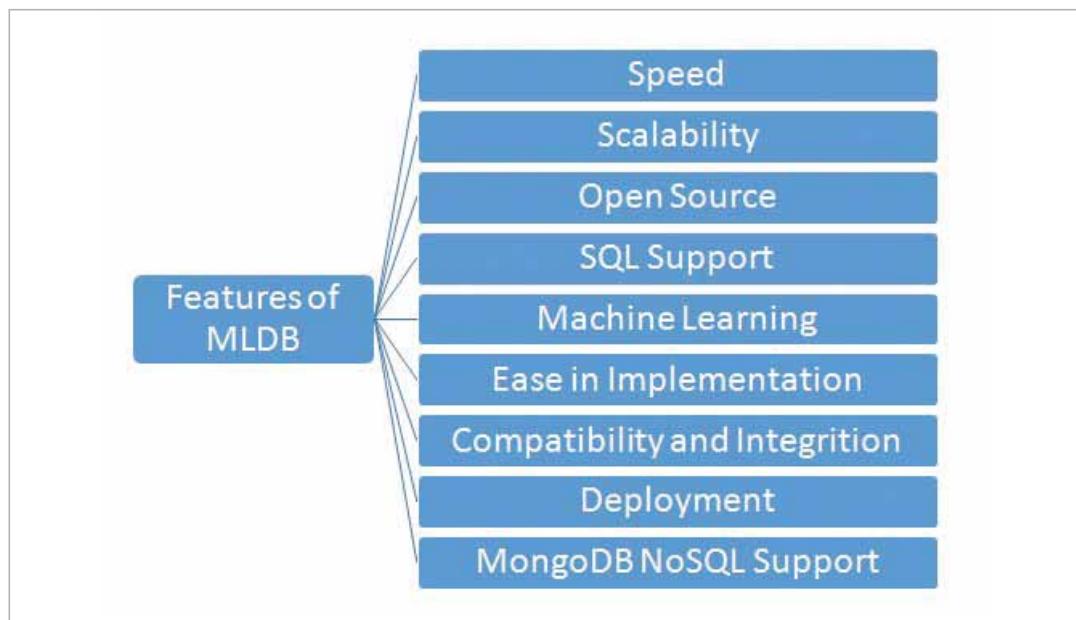


Figure 2: Features of MLDB

**Compatibility and integration:** MLDB provides a higher degree of compatibility with different application programming interfaces (APIs) and modules including JSON, REST and Python based wrappers.

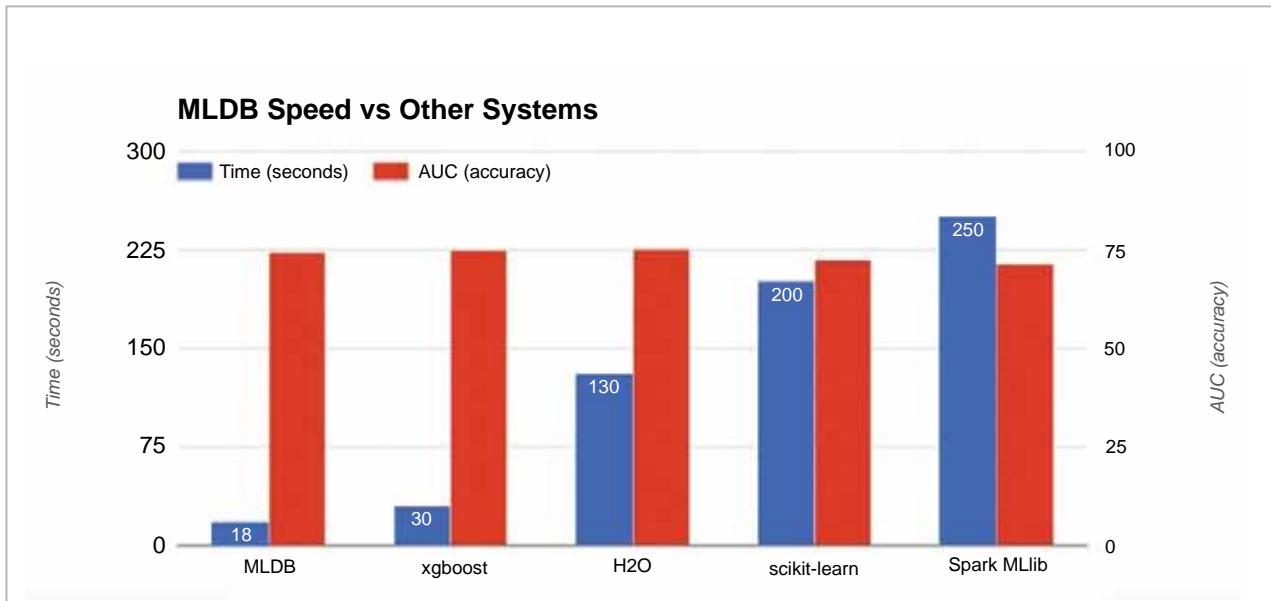


Figure 3: Performance of MLDB compared with other machine learning libraries

**Deployment:** MLDB can be deployed easily on an HTTP endpoint that provides easy interface and fast deployment.

**MongoDB and NoSQL support:** The bridge or interface of MongoDB and MLDB can be created to support MLDB SQL queries. These SQL queries can be executed on MongoDB collections, which give MLDB more powers to interact with NoSQL databases for unstructured and heterogeneous datasets.

Figure 3 depicts the performance of MLDB when compared to other libraries. An execution of the 100 Tree Random Forest approach is done on 1 million rows with one node using MLDB and other libraries. From the graphical results, it is evident that MLDB is comparatively better, takes less time and its accuracy compares well with other machine learning libraries. The performance of MLDB is comparable with that of xgboost, H2O, Scikit-Learn and Spark MLib.

## Support for algorithms in MLDB

The procedures of MLDB are used for the training of machine learning models and these are implemented using functions. Given below is the list of functions and procedures that can be used in MLDB with high performance.

## ***Supervised machine learning***

- Classification including multi-label classification: classifier.train
  - Logistic regressions
  - Generalised linear models
  - Neural networks
  - Decision trees
  - Random forests
  - Naive Bayes models using boosting and bagging
- Support vector machines (SVM): svm.train
- Classifiers calibration: probabilizer.train

## ***Deep learning***

- TensorFlow models: tensorflow.graph

## ***Clustering***

- K-Means models: kmeans.train

## ***Dimensionality reduction, manifold learning and visualisation***

- Truncated Singular Value Decompositions (SVD): svd.train
- t-distributed Stochastic Neighbour Embedding (t-SNE): tsne.train

## ***Feature engineering***

- SentiWordNet models: import.sentiwordnet
- Word2Vec: import.word2vec
- Term-Frequency/Inverse-Document-Frequency (TF-IDF) models: tfidf.train
- Count-based features: statsTable.train
- Feature Hashing/Vectorize features: feature\_hasher

## ***Installing MLDB on different platforms***

MLDB provides a Web based interface for the easiest implementation and hands-on experience. A free session of MLDB can be experienced for 90

minutes using a Web based panel after signing up (registration) on <https://mldb.ai/#signup>. There are many demos and a lot of documentation available so that a cloud based MLDB can be worked out without installation on the local system. Even the self-created datasets can be uploaded on this hosted session.

## Editions for local installation

There are two editions of MLDB that are free, and are distributed as community and enterprise editions. To run the MLDB enterprise edition, you need to enter the licence key to activate the software. A licence key can be created for first-time users on signing up at [https://mldb.ai/#license\\_management](https://mldb.ai/#license_management) and filling the required details in the registration form.

	Community edition	Enterprise edition (Free trial)
MLDB	Available	Available
Licensing	Apache License v2.0	Non-commercial
Cost / Pricing	Free	Free
Issues and support	GitHub issues	MLDB support
Download Instructions	<a href="https://github.com/mldbai/mldb/blob/master/Building.md">https://github.com/mldbai/mldb/blob/master/Building.md</a>	<a href="https://docs.mlbd.ai/doc/builtin/Running.md.html#packages">https://docs.mlbd.ai/doc/builtin/Running.md.html#packages</a>

## Docker image

The classical MLDB distribution is the Docker image. While other distributions are available for virtual machines, the Docker image is executed as a container. This method is used for Linux flavours or private cloud deployments.

After the installation of Docker, the MLDB container is launched with a pre-specified mapped directory.

```
$ mkdir </my/system/path/myMLDBdata>
```

With the execution of the following commands, a port can be set using the *mldbport* parameter.

```
docker run --rm=true \
-v </my/system/path/myMLDBdata>:/mldb_data \
-e MLDB_IDS="`id`" \
-p 127.0.0.1 (IP-Address):<mldbport>:80 (Port) \
quay.io/mldb/mldb:latest
```

For security and overall integrity, a tunnel is established for remote servers as in the following instruction using the SSH tunnel:

```
$ ssh -f -o ExitOnForwardFailure=yes <user>@<remotehost> -L
<localport>:127.0.0.1 (IP-Address):<mldbport> -N
```

Once the message ‘MLDB Ready’ is viewed, the browsing and activation of MLDB can be done on a Web browser using the URL *http://localhost:<localport>*.

## **Virtual appliance**

The installation of MLDB for virtualisation is very easy. The virtual application (OVA file) is available so that it can be imported using VirtualBox or any other virtualisation software.

After downloading VirtualBox from <https://www.virtualbox.org/wiki/Downloads>, the OVA file of MLDB Appliance at <http://public.mlbd.ai/mldb.ova> can be imported.

Simply double-click the OVA file or select ‘Import Appliance’ in the *File Menu* of VirtualBox and finally point out to the downloaded MLDB OVA distribution. The default user name to log in on OVA is ‘ubuntu’ and the password for successful authentication is ‘mldb’.

After these steps, the MLDB instance can be executed using the URL *http://localhost:8080/* on any Web browser.

# About the Authors

## Aakash Beniwal

The author works with Infosys Limited, Pune, as a testing engineer and has over two years' experience in this domain. He can be reached at [aakashnavodaya@gmail.com](mailto:aakashnavodaya@gmail.com).

## Abhinav Nath Gupta

The author is a software development engineer at Cleo Software India Pvt Ltd, Bengaluru. He is interested in studying machine learning and artificial intelligence techniques.

## Amit Doegar

The author is an assistant professor at the Department of Computer Science and Engineering, National Institute of Technical Teachers Training and Research (NITTTR). He can be contacted at [amit@nitttrchd.ac.in](mailto:amit@nitttrchd.ac.in).

## Anand Nayyar

The author works at Duy Tan University in Vietnam. He loves to work and research on open source technologies, sensor communications, network security and the Internet of Things. He can be reached at [anandnayyar@duytan.edu.vn](mailto:anandnayyar@duytan.edu.vn). YouTube channel: *Gyaan with Anand Nayyar* at [www.youtube.com/anandnayyar](http://www.youtube.com/anandnayyar).

## Ashish Sinha

The author is a software engineer based in Bengaluru. A software enthusiast at heart, he is passionate about using open source technology and sharing it with the world. He can be reached at [ashi.sinha.87@gmail.com](mailto:ashi.sinha.87@gmail.com). Twitter handle: @sinha\_tweet.

## Deepshikha Shukla

The author is a technology journalist at EFY.

## Dipankar Ray

The author has more than 20 years of experience with open source versions of UNIX operating systems and Sun Solaris. Presently, he works on data analysis and machine learning using neural networks and different statistical tools. You can contact him at [dipankarray@ieee.org](mailto:dipankarray@ieee.org).

## Dr Gaurav Kumar

The author is the MD of Magma Research and Consultancy Services, Ambala. He is associated with various academic and research institutes, where he delivers lectures and conducts technical workshops. He can be reached at [kumargaurav.in@gmail.com](mailto:kumargaurav.in@gmail.com).

## Dr K.S. Kuppusamy

The author is an assistant professor of computer science at the School of Engineering and Technology, Pondicherry Central University, Puducherry. He has 13+ years of teaching and research experience in academia and industry. His research interests include accessible computing, Web information retrieval, mobile computing, etc. He can be reached via mail at [kskuppu@gmail.com](mailto:kskuppu@gmail.com).

## Jatin Karthik Tripathy

The author takes a keen interest in graphics modelling, artificial intelligence and implementations of neural networks.

**Miren Karamta**

The author is a project scientist and IT systems manager at the Bhaskaracharya Institute for Space Applications and Geo-informatics (BISAG), Gandhinagar, Gujarat. You can reach him via email at [mirenkaramta@yahoo.com](mailto:mirenkaramta@yahoo.com). LinkedIn: <https://in.linkedin.com/in/miren-karamta-2b929122>

**Navneet Dwivedi**

The author is a FOSS enthusiast and can be contacted at [navneetdwivedi1999@gmail.com](mailto:navneetdwivedi1999@gmail.com).

**Neetesh Mehrotra**

The author works at NIIT Technologies as a senior test engineer. His areas of interest are Java development and automation testing. He can be reached at [mehrotra.neetesh@gmail.com](mailto:mehrotra.neetesh@gmail.com).

**Neethu C. Sekhar**

The author has over seven years of experience in teaching and works as an assistant professor in the Department of Computer Science at Amal Jyothi College of Engineering, Kerala. She has worked with many open source tools, Python and other related technologies.

**Palak Shah**

The author is an associate consultant at Capgemini and loves to explore new technologies. She is an avid reader and writer of technology related articles. She can be reached at [palak311@gmail.com](mailto:palak311@gmail.com).

**Priya Ravindran**

The author has an M.Sc (electronics) from VIT University, Vellore, Tamil Nadu. She loves to explore new avenues and is passionate about writing.

**Sagar Bhanudas Joshi**

The author works as a developer to help embrace the Universal Windows Platform and the Microsoft Azure platform for more than six years. His current role includes helping SaaS companies architect, design and onboard solutions to Microsoft Cloud Platform, with special focus on machine learning and AI. Joshi lives and works in Mumbai, India. You can contact him on Twitter: @sagarjms

**Sandeep Alur**

Director – partner engagements, Microsoft

**Sani Theo**

The author is a technical editor at EFY.

**Siddhartha Agarwal,**

group VP -product management and strategy, Oracle Cloud Platform

**Swapneel Mehta**

The author has worked with Microsoft Research, CERN and startups in the AI and cyber security domains. An open source enthusiast, he enjoys spending his time organising software development workshops for school and college students. You can connect with him at <https://www.linkedin.com/in/swapneelm> and find out more at <https://github.com/SwapneelM>

**Shashidhar Soppin**

The author is a senior architect who has 17+ years of experience in the IT industry, with specialised expertise in virtualisation, cloud computing, Docker, open source and Open Stack. He owns patents, and has written and presented papers in various forums. You can contact him at [shashi.soppin@gmail.com](mailto:shashi.soppin@gmail.com).

**Siva Rama Krishna Reddy B.**

The author is keenly interested in researching open source technologies like Android, artificial intelligence and deep learning

**Somanath T.V.**

The author has 15+ years' of experience in the IT industry and is currently working as technology consultant at FTL Technology Systems, Kochi, Kerala. He is interested in AI, VR, AR and related areas. He can be reached at [somanathtv@gmail.com](mailto:somanathtv@gmail.com); blog: <https://somji.blogspot.in/>.

**Surabhi Dwivedi**

The author works as a principal technical officer at the Centre for Development of Advanced Computing (CDAC). She has more than 12 years of experience with various application oriented R&D projects, in software development, in relational databases, etc.

**Tularam M. Bansod**

The author is proprietor of TMB Enterprises, a startup focused on embedded systems. He has authored two books on microcontroller programming.

**Vinayak Ramachandra Adkoli**

The author is a B.E. in industrial production. He has worked in three different polytechnics over the past ten years and can be contacted at [karnatakastory@gmail.com](mailto:karnatakastory@gmail.com).

**Vivek Ratan**

The author is a B. Tech in electronics and instrumentation engineering. He currently works as a senior automation test engineer at Tata Technologies, Pune and as a freelance educator at LearnerKul, Pune. He can be contacted at [ratanvivek14@gmail.com](mailto:ratanvivek14@gmail.com).

**Zeljko Loncaric**

The author is working as a marketing engineer at congatec AG.