

# PYTHON MACHINE LEARNING

An hands-on introduction to artificial intelligence coding,  
a project-based guide with practical exercises

- 7 DAYS -  
**CRASH COURSE**

Learn Python in a week and master it

Book 2

COMPUTER PROGRAMMING ACADEMY

# PYTHON MACHINE LEARNING

An hands-on introduction to artificial intelligence coding,  
a project-based guide with practical exercises

- 7 DAYS -  
**CRASH COURSE**

Learn Python in a week and master it

Book 2

COMPUTER PROGRAMMING ACADEMY

# **PYTHON MACHINE LEARNING**

***Learn Python in a Week and Master it***

---

7 Days Crash Course

---

An hands-on introduction to artificial intelligence coding, a project-based guide with practical exercises

**Computer Programming Academy**

**Copyright © 2020 – Computer Programming Academy**

## **All rights reserved**

The content contained within this book may not be reproduced, duplicated or transmitted without direct written permission from the author or the publisher.

Under no circumstances will any blame or legal responsibility be held against the publisher, or author, for any damages, reparation, or monetary loss due to the information contained within this book, either directly or indirectly.

**Legal Notice:**

This book is copyright protected. This book is only for personal use. You cannot amend, distribute, sell, use, quote or paraphrase any part, or the content within this book, without the consent of the author or publisher.

**Disclaimer Notice:**

Please note the information contained within this document is for educational and entertainment purposes only. All effort has been executed to present accurate, up to date, and reliable, complete information. No warranties of any kind are declared or implied. Readers acknowledge that the author is not engaging in the rendering of legal, financial, medical or professional advice. The content within this book has been derived from various sources. Please consult a licensed professional before attempting any techniques outlined in this book. By reading this document, the reader agrees that under no circumstances is the author responsible for any losses, direct or indirect, which are incurred as a result of the use of information contained within this document, including, but not limited to errors, omissions, or inaccuracies.

# Table of Contents

## Introduction

### Day 1: Introduction to Machine Learning

Core concepts of machine learning  
Basic machine learning terminologies  
Importance of machine learning  
Review Quiz

### Day 2: Machine Learning Models

Rule-based systems  
Case-Based Reasoning  
Artificial Neural Networks (ANN)  
Genetic Algorithms  
Cellular Automata  
Fuzzy Systems  
Multi-Agent Systems  
Swarm Intelligence  
Reinforcement Learning  
Hybrid systems  
Statistics  
Probabilistic Programming  
Review Quiz

### Day 3: Supervised Machine Learning Algorithms

Regression  
Classification  
Ensemble Methods  
Review Quiz

### Day 4: Unsupervised Machine Learning Algorithms

Clustering  
Dimensionality Reduction  
Transfer Learning  
Natural Language Processing  
Word Embedding  
Apriori machine learning algorithm

Review Quiz

## Day 5: Data Pre-processing and Creation of Training Dataset

Overview Data Preprocessing

Steps of Data Pre-processing

## Day 6: Machine Learning Libraries

Prerequisites for application of Scikit-Learn library

Application of machine learning using Scikit-Learn library

Creating Training and Test subsets

Building the Machine Learning Model

## Day 7: Neural Network Training With Tensorflow

Fundamentals of Neural Network

Training a Neural Network using TensorFlow

## Extra content

Python Programming

Python Data Science

## Conclusion

# Introduction

Congratulations on purchasing *Python Machine Learning: An hands-on introduction to artificial intelligence coding, a project-based guide with practical exercises (Book 2)* , and thank you for doing so.

The following chapters will discuss the fundamental concepts of machine learning models that can be generated and advanced by utilizing Python based libraries. There are 7 chapters in this book, crafted specifically to help you master all the machine learning concepts in just a week.

The first chapter will introduce you to the core concepts of machine learning as well as various terminologies that are frequently used in this field. It will also provide you a thorough understanding of the significance of machine learning in our daily lives. Some of the most widely used learning models, such as Artificial Neural Networks (ANN) and Genetic Algorithms (GA) are explained in detail in the second chapter .

Chapter 3 will introduce you to the four fundamental machine learning algorithms with explicit details on the supervised machine learning algorithms. The subsequent chapter will include details on various unsupervised machine learning algorithms, such as clustering and dimensionality reduction among others.

You will also learn how the raw data can be processed to generate high quality training data set for the production of a successful machine learning model. The sixth chapter of this book will deep dive into the functioning of ML library called Scikit-Learn along with guidance on resolving nonlinear issues with k-nearest neighbor and kernel trick algorithms. The final chapter will explain the nuances of developing a neural network to generate predictions and build the desired machine learning model by utilizing the Tensorflow Python library.

We have also provided review exercises to help you test your understanding through this process. Every chapter of this book has real life

examples and applications included to solidify your understanding of each concept.

# Day 1: Introduction to Machine Learning

The modern concept of Artificial Intelligence technology is derived from the idea that machines are capable of human like intelligence and potentially mimic human thought processing and learning capabilities to adapt to fresh inputs and perform tasks with no human assistance. Machine learning is integral to the concept of artificial intelligence. Machine Learning can be defined as a concept of Artificial Intelligence technology that focuses primarily on the engineered capability of machines to explicitly learn and self-train, by identifying data patterns to improve upon the underlying algorithm and make independent decisions with no human intervention. In 1959, pioneering computer gaming and artificial intelligence expert, Arthur Samuel, coined the term “machine learning” during his tenure at IBM.

Machine learning stems from the hypothesis that modern day computers have an ability to be trained by utilizing targeted training data sets, that can be easily customized to develop desired functionalities. Machine learning is driven by the pattern recognition technique wherein the machine records and revisits past interactions and results that are deemed in alignment with its current situation. Given the fact that machines are required to process endless amounts of data, with new data always pouring in, they must be equipped to adapt to the new data without needing to be programmed by a human, which speaks to the iterative aspect of machine learning.

Now the topic of machine learning is so “hot” that the world of academia, business as well as the scientific community have their own take on its definition. Here are a few of the widely accepted definitions from select highly reputed sources:

- *“Machine learning is the science of getting computers to act without being explicitly programmed”. – Stanford University*
- *“The field of Machine Learning seeks to answer the question, “How can we build computer systems that*

*automatically improve with experience, and what are the fundamental laws that govern all learning processes?”* – Carnegie Mellon University

- *“Machine learning algorithms can figure out how to perform important tasks by generalizing from examples”.* – University of Washington
- *“Machine Learning, at its most basic, is the practice of using algorithms to parse data, learn from it, and then make a determination or prediction about something in the world”.* – Nvidia
- *“Machine learning is based on algorithms that can learn from data without relying on rules-based programming”.* – McKinsey & Co.

## **Core concepts of machine learning**

The biggest draw of this technology is its inherent ability of the system to automatically learn programs from the raw data in lieu of manually engineering the program for the machine. Over the last 10 years or so, the application of ML algorithms has expanded from computer science labs to the industrial world. Machine learning algorithms are capable of generalizing tasks so they can be executed iteratively. The process of developing specific programs for specific tasks is extremely taxing in terms of time and money, but occasionally, it is just impossible to achieve. On the other hand, machine learning programming is often feasible and tends to be much more cost effective. The use of machine learning in addressing ambitious issues of widespread importance such as global warming and depleting underground water levels, is promising with massive collection of relevant data.

*“A breakthrough in machine learning would be worth ten Microsofts”.*  
– Bill Gates

A number of different types of ML models exist today, but the concept of ML largely boils down to three core components “representation”, “evaluation”, and “optimization”. Here are some of the standard concepts that are applicable to all of them:

## **Representation**

Machine learning models are incapable of directly hearing, seeing, or sensing input examples. Therefore, data representation is required to supply the model with a useful vantage point into the key qualities of the data. To be able to successfully train a machine learning model selection of key features that best represent the data is very important. “Representation” simply refers to the act of representing data points to the computing system in a language that it understands with the use of a set of classifiers. A classifier can be defined as “a system that inputs a vector of discrete and/or continuous feature values and outputs a single discrete value called class”. For a model to learn from the represented data, the training data set or the “hypothesis space” must contain the desired classifier that you want the models to be trained on. Any classifiers that are external to the hypothesis space cannot be learned by the model. The data features used to represent the input are extremely crucial to the machine learning process. The data features are so critical to the development of the desired machine learning model that it could easily be the key distinction between a successful and failed machine learning project. A training data set consisting of multiple independent feature sets that are well correlated with the class can make the machine learning much smoother. On the other hand, class consisting of complex features may not be easy to learn from for the machine. This usually needs the raw data to be processed to allow the construction of desired features from it, which could then be utilized for the development of the ML model. The process of deriving features from raw data tends to be the most time consuming and laborious part of the ML projects. It is also considered the most creative and exciting part of the project where intuition and trial and error play just as important role as the technical requirements. The process of ML is not a single shot process of developing a training data set and executing it; instead, it’s an iterative process that requires analysis of the post run results followed by modification of the training data set and then repeating the whole process all over again. Another contributing factor to the extensive time and effort required in the engineering of the training

data set is domain specificity. Training data set for an e-commerce platform to generate predictions based on consumer behavior analysis will be very different from the training data set required to develop a self-driving car. However, the actual machine learning process largely holds true across the industrial spectrum. No wonder, a lot of research is being done to automate the feature engineering process.

## **Evaluation**

Essentially the process of judging multiple hypothesis or models to choose one model over another is referred to as an evaluation. To be able to differentiate between good classifiers from the not so good ones, an “evaluation function” must be used. The evaluation function is also called “objective”, “utility”, or “scoring” function. The machine learning algorithm has its own internal evaluation function, which tends to be different from the external evaluation function used by the researchers to optimize the classifier. Normally the evaluation function will be defined prior to the selection of the data representation tool and tends to be the first step of the project. For example, the machine learning model for self-driving cars has a feature that allows identification of pedestrians in the car’s vicinity at near zero false negatives and a low false positive, which are the evaluation functions and the pre-existing condition that needs to be “represented” using applicable data features.

## **Optimization**

The process of searching the space of presented models to achieve better evaluations or highest scoring classifier is called as “optimization”. For algorithms with multiple optimum classifiers, the selection of optimization technique is very important in the determination of the classifier produced as well as to achieve a more efficient learning model. A variety of off-the-shelf optimizers are available in the market that will help you kick start a new machine learning model before eventually replacing them with a custom designed optimizers.

**Table 1. The three components of learning algorithms.**

Representation	Evaluation	Optimization
Instances	Accuracy/Error rate	Combinatorial optimization
K-nearest neighbor	Precision and recall	Greedy search
Support vector machines	Squared error	Beam search
Hyperplanes	Likelihood	Branch-and-bound
Naive Bayes	Posterior probability	Continuous optimization
Logistic regression	Information gain	Unconstrained
Decision trees	K-L divergence	Gradient descent
Sets of rules	Cost/Utility	Conjugate gradient
Propositional rules	Margin	Quasi-Newton methods
Logic programs		Constrained
Neural networks		Linear programming
Graphical models		Quadratic programming
Bayesian networks		
Conditional random fields		

## Basic machine learning terminologies

**Agent** – In context of reinforcement learning, an agent refers to an entity that utilizes a policy to max out the expected return achieved with the transition of different environment states.

**Boosting** – Boosting can be defined as a ML technique that would sequentially combine set of simple and low accuracy classifiers (known as "weak" classifiers) into a classifier which is highly accurate (known as "strong" classifier) by increasing the weight of the samples that are being classified wrongly by the model.

**Candidate generation** – The phase of selecting the initial set of suggestions provided by a recommendation system is referred to as candidate generation. For example, a book library can offer 60,000 different books. Through this phase, a subset of few 100 titles meeting the needs of a particular user will be produced and can be refined further to an even smaller set as needed.

**Categorical Data** – Data features boasting a distinct set of potential values is called as categorical data. For example, a categorical feature named TV model can have a discrete set of multiple possible values, including Smart, Roku, Fire.

**Checkpoint** – Checkpoint can be defined as a data point that will capture the state of the variables at a specific moment in time of the ML model. With the use of checkpoints, training can be carried out across multiple sessions and model weights or scores can be exported .

**Class** – Class can be defined as “one of a set of listed target values for a given label”. For instance, a model designed to detect junk emails can have 2 different classes, namely, “spam” and “not spam”.

**Classification model** – The type of machine learning model used to differentiate between multiple distinct classes of the data is referred to as a classification model. For example, a classification model for identification of dog breeds could assess whether the dog picture used as input is Labrador, Schnauzer, German Shepherd, Beagle and so on.

**Collaborative filtering** – The process of generating predictions for a particular user based on the shared interests of a group of similar users is called collaborative filtering.

**Continuous feature** – It is defined as a “floating point feature with an infinite range of possible values”.

**Discrete feature** – It is defined as a feature that can be given only a finite set of potential values and has no flexibility .

**Discriminator** – A system used to determine whether the input samples are realistic or not is called as discriminator.

**Down-sampling** – The process of Down-sampling refers to the process used to reduce the amount of info comprised in a feature or use of an extremely low percentage of classes that are abundantly represented in order to train the ML model with higher efficiency.

**Dynamic model** – A learning model that is continuously receiving input data to be trained in a continuous manner is called a dynamic model.

**Ensemble** – A set of predictions created by combining predictions of more than one model is called an ensemble.

**Environment** – The term environment used in the context of reinforcement machine learning constitutes “the world that contains the agent and allows the agent to observe that world’s state” .

**Episode** – The term episode used in the context of reinforcement machine learning constitutes every sequential trial taken by the model to learn from its environment.

**Feature** – Any of the data variables that can be used as an input to generate predictions is called a feature.

**Feature engineering** – Feature engineering can be defined as “the process of determining which features might be useful in training a model, and then converting raw data from log files and other sources into said features”.

**Feature extraction** – Feature extraction can be defined as “the process of retrieving intermediate feature representations calculated by an unsupervised or pre-trained model for use in another model as input”.

**Few-shot learning** - Few-shot learning can be defined as “a machine learning approach, often used for object classification, designed to learn effective classifiers from only a small number of training examples” .

**Fine tuning** – The process of “performing a secondary optimization to adjust the parameters of an already trained model to fit a new problem” is called as fine tuning. It is widely used to refit the weight(s) of a “trained unsupervised model” to a “supervised model”.

**Generalization** – A machine learning model’s capability to produce accurate predictions from fresh and unknown input data instead of the data set utilized during the training phase of the model is called generalization.

**Inference** – In context of ML, inference pertains to the process of generating predictions and insight with the application of an already trained model to unorganized data sample.

**Label** – In context of machine learning (supervised), the "answer" or "result" part of an example is called a label. Each title in a labeled dataset will consist of single or multiple features along with a label. For example, in a house data set, the features could contain the year built, number of rooms and bathrooms, while the label can be the “house's price” .

**Linear model** – Linear model is defined as a model that can assign singular weight to each feature for generating predictions.

**Loss** – In context of ML, loss pertains to the measure of the extent by which the predictions produced by the model are not in line with its training labels.

**Matplotlib** – It is an “open source Python 2-D plotting library” that can be utilized to visualize various elements of ML.

**Model** – In context of ML, a model refers to a representation of the learning and training that has been acquired by the system from the training dataset.

**NumPy** – It is an open sourced data library that can provide effective operations to be used on Python arrays.

**One-shot learning** – In context of machine learning, one-shot learning can be defined as the machine learning approach that allows learning of effective classifiers from unique training sample and is frequently utilized classification of objects .

**Overfitting** - In context of machine learning, overfitting is referred to as production of a model that can match the training dataset extremely closely and renders the model inefficient in making accurate predictions on fresh input.

**Parameter** – Any variable of the ML model, which would allow the machine learning system to self-learn independently is called parameter.

**Pipeline** – In context of ML, pipeline pertains to the infrastructure that surrounds a learning algorithm and comprises of a collection of data, any data additions made to the training data files, training of single or multiple models, and releasing the models into live environment.

**Random forest** – In context of machine learning, the concept of random forest pertains to an ensemble technique to find a decision tree that would most accurately fit the training dataset by creating two or more decision trees with a random selection of features.

**Scaling** - In context of machine learning, scaling refers to “a common feature engineering practice to tame a feature's range of values to match the range of other features in the dataset”.

**Sequence model** - A sequence model simply refers to a model with sequential dependency on data inputs to generate a future prediction.

**Underfitting** – It is the process of generating a model with low efficacy to generate predictions due to the lack of proper understanding of the training data set.

**Validation** – The process of evaluating the quality of the machine learning model with the use of the validation set during its training phase is called validation. The main goal of this process is to make sure that the performance of the model can be applied beyond the training dataset.

## Machine Learning in Practice

The complete process of machine learning is much more extensive than just the development and application of machine learning algorithms and can be divided into steps below:

1.

Define the goals of the project, taking into careful consideration all the prior knowledge and domain expertise available. Goals can easily become ambiguous since there are always additional things you want to achieve than practically possible to implement.

2.

The data pre-processing and cleaning must result in a high quality data set. This is the most critical and time-consuming step of the whole project. The larger the volume of data, the more noise it brings to the training data set, which must be eradicated before feeding to the learner system.

3.

Selection of appropriate learning model to meet the requirements of your project. This process tends to be rather simple, given the various types of data models available in the market.

4.

Depending on the domain the machine learning model is applied to, the results may or may not require a clear understanding of the model by human experts as long as the model can successfully deliver desired results.

5.

The final step is to consolidate and deploy the knowledge or information gathered from the model to be used on an industrial level.

6.

The whole cycle from step 1 to 5 listed above is iteratively repeated until a result that can be used in practice is achieved.

## **Importance of machine learning**

To get a sense of how significant machine learning is in our everyday lives, it is simpler to state what part of our cutting edge way of life has not been touched by it. Each aspect of human life is being impacted by the "smart machines" intended to expand human capacities and improve efficiencies. Artificial Intelligence and machine learning technology is the focal precept of the "Fourth Industrial Revolution", that could possibly question our thoughts regarding being "human".

Here are a few reasons to help you understand the significance of machine learning in our daily lives:

•

Automation of repetitive learning and revelation from data. Not at all like hardware-driven robotic automation that simply automates manual assignments, machine learning

allows the performance of high volume, high volume, computer based tasks consistently and dependably.

- 

Machine learning algorithms are helping Artificial Intelligence to adapt to the evolving world by allowing the machine or system to learn, take note, and improve upon its prior errors. Machine learning algorithm functions as a classifier or a predictor to acquire new skills and identify data pattern and structure. For example, machine learning algorithm has generated a system that can teach itself how to play chess and even how to generate product recommendations based on customer activity and behavior data. The beauty of this model is that it adapts with every new set of data.

- 

Machine learning has made analysis of deeper and larger data set feasible with the use of neural networks containing multiple hidden layers. Think about it; a fraud detection system with numerous concealed layers would deemed a work of fantasy just a couple of years ago. With the advent of big data and unlikely to envision computer powers, a whole new world is on the horizon. Data to the machines resembles the gas to the vehicle; the more data you can add to them, faster and more accurate results will get. Deep learning models flourish with abundance of data because they gain straightforwardly from the data.

- 

The “deep neural networks” of the machine learning algorithms have resulted in unbelievable accuracy. For example, frequent and repeated use of smart tech like “Amazon Alexa” and “Google Search”, result in increased accuracy derived from deep learning. These “deep neural networks” are also empowering our medical field. Image classification and object recognition are now capable of finding cancer on MRIs with similar accuracy as that of a highly trained radiologist.

- 

Artificial Intelligence is allowing for enhanced and improved use of big data analytics in conjunction with machine

learning algorithms. Data has evolved as its own currency, and when algorithms are self-learning, it can easily become “intellectual property”. The raw data is similar to a gold mine in that the more and deeper you dig, the more “gold” or valuable insight you can dig out or extract. Application of machine learning algorithms to the data can enable you to find the correct solutions quicker and makes for an upper hand. Keep in mind the best information will consistently win, despite the fact that everyone is utilizing comparative techniques.

## Review Quiz

Answer the questions below to verify your understanding of the concepts explained in this chapter. The answer key can be found at the end of the quiz.

1. Name the 3 core concepts of the machine learning technology.
2. What is the selection procedure of the initial set of recommendations called?
3. \_\_\_\_ is a set of predefined target values for an indicated label.
4. Name two open source machine learning libraries.
5. What function is used to distinguish between the good and bad classifiers?
6. The process of searching the space of presented models to achieve better evaluations or highest scoring classifier is called \_\_\_\_.
- 7.

Name the system used to determine real input data from the fake input.

8.

Come up with 3 labels for a dog dataset.

9.

While the ML model is undergoing training, the quality of the model can be checked by utilizing the process called \_\_\_\_.

10.

The \_\_\_ revolution has resulted in development of Artificial Intelligence and machine learning technologies.

## Answer Key

1.

Representation, Evaluation and Optimization.

2.

Candidate Generation.

3.

Class.

4.

NumPy, Matplotlib.

5.

Evaluation Function.

6.

Optimization.

7.

Discriminator.

8.

Breed, Color, Age. (any other data feature that will help you identify the dataset contains info on dogs can be used as a label)

9.

Validation.

10.

Fourth Industrial.

# **Day 2: Machine Learning Models**

As we are marching towards a more digital lifestyle, the data has become the new gold. As a result, the need for the development of smart machines that can learn from this large volume of data and generate predictions with high accuracy is paramount. Some of the machine learning models that have already been developed and is widely used across the industrial spectrum are explained below.

## **Rule-based systems**

Rule-based systems (RBS) employ if and then statements to feed an inference engine with “working memory” of info pertaining to the problem. These condition and action parts of the rules are derived from expert knowledge. The inference engine has a pattern matcher, to make a decision on the rules which are relevant and a rule applicator to then selectively apply these rules. The actions (then) of the rule applied to create new information that can be incorporated into the working memory. The “match-select-act” cycle between the knowledge base and working memory will be repeated until all the pertinent rules are searched successfully. Since rules tend to be vague, uncertainty could be easily integrated in this system by multiple techniques like possibility theory, certainty factors, or Cohen’s theory of endorsements. These approaches simply assign uncertainty values such as: “probabilities, belief functions, membership values”, to rules and facts, by human experts. The forward and backward chaining comprise the two rule systems. Forward chaining is data driven and uses rules to draw a conclusion from the initial facts. On the other hand, backward chaining is goal driven and is based on its initial hypothesis which then needs to be verified by the applicable rules. Simply put, backward chaining works to justify its decisions while forward chaining will discover information that may be derived from the given data.

RBS involves no learning, and solutions are based completely on the conditional rules established based on human knowledge. But they are easy to understand, implement, and maintain for the same reason. RBS are unable to automatically add or modify the rules and can only be

implemented with comprehensive knowledge of the subject matter. They also tend to be more difficult to scale up. RBS are used in disease diagnosis and plant and animal identification as well as regional environment assessment tools.

## **Case-Based Reasoning**

“Case-Based Reasoning” (CBR) utilizes memory recollection to address problems that are similar in nature to a past problem the system encountered. CBR recognizes that similar problems may have similar solutions, and with accrued learning and multiple attempts, problems get easier to solve. CBR essentially has a four step process:

1. Search the database and retrieve similar past cases.
2. By utilizing the retrieved case and producing a solution of the new issue.
3. By employing simulations or test executions, revise the proposed solution.
4. Retain and store the new solution for future use.

CBR can draw inferences only from similar past cases of the problem. There is no insight into the system or the process as it is happening so it is a black box approach. This is beneficial for complex processes where greater understanding and insight is not necessary or even possible. CBR is used for diagnosis, control, and planning as well as prediction. Some examples of CBR being used in environmental applications are planning fire-fighting, managing waste-water treatment, weather prediction and monitoring air quality.

## **Artificial Neural Networks (ANN)**

This model has been inspired by the structure of the human brain and employs multiple processing units like human neurons or nodes working in unison. Each ANN has multiple hidden layers along with an input and an output layer. These neurons are intricately connected to each other by weighted links. A single neuron can accept input from multiple neurons. When a neuron is activated, it casts a weighted vote to guide the activation of the neuron that is gathering that input. An algorithm adjusts these weights based on the training data to optimize learning. A simple algorithm called “fire together, wire together” increases the weight between two connected neurons when the activation of either neuron leads to subsequent activation of the other neuron. “Concepts” are formed and distributed through the sub-network of shared neurons.

The most common ANN work on a unidirectional flow of information and is called “Feed forward ANN”. However, ANN are also capable of bidirectional and cyclic flow of information to achieve state equilibrium. ANNs learn from past cases by adjusting the connected weights and rely on fewer prior assumptions. This learning could be supervised or unsupervised. With supervised learning, every input pattern will result in the correct ANN output. To minimize the error between the given O/P and the O/P generated by ANN, the weights can be varied. For example, reinforced learning, which is a form of “supervised learning”, informs the ANN if the generated output is correct instead of generating the right output directly. On the other hand, unsupervised learning provides multiple input pattern to the ANN, and then the ANN itself will explore the relationship between these patterns and learn to categorize them accordingly. ANNs with a combination of supervised and unsupervised learning are also available. To solve data heavy problems wherein the algorithm or rules are unknown or difficult to comprehend, ANNs are highly useful owing to their non-linear computations and data structure. They are robust to multi-variable data errors and can easily process complex information in parallel. Though, the black box model of ANN is a major disadvantage, which makes them unsuitable for problems that require a deep understanding and insight into the actual process.

ANNs can be used to resolve problems that require:

-

**Pattern classification** by assigning input patterns to one of the classes that have been determined in advance. For example, classification of land based on satellite images.

- 

**Clustering** , which is essentially unsupervised pattern classification. For example, prediction of ecological status of streams by utilizing defined input patterns.

- 

**Function approximation or regression** , which can create a function out of the given set of training patterns. For example, prediction of ozone concentration in the atmosphere, estimation of amount of nitrate in groundwater, and modeling water supply.

- 

**Prediction** , which uses past samples in a time series to estimate the output. For example, prediction of air and water quality.

- 

**Optimization** , which tends to maximize or minimize a “cost function” subject to constraints. For example, calibration of infiltration equations.

- 

**Retrieval by content** is capable of recalling memory even if the input is incomplete or deformed. For example, by utilizing the satellite images to produce water quality proxies.

- 

**Process control** , which aims at keeping near constant speed under changing data load by altering the throttle angle. For example, engine speed control.

## Genetic Algorithms

As the name suggests, Genetic Algorithm (GA) mimics natural selection theory by passing the traits of the fitter solutions to the offspring and replacing the poorer solutions. The GA continues to evolve until the satisfactory solution of the problem is achieved. Just like human chromosomes, every possible solution will be encoded as a “binary string”

of characters, and each set of subsequent populations are called “generations”. The starting or original population will be randomly produced and all subsequent populations are generated via the process of selection and reproduction. A selective subset of the population is then bred to generate new chromosomes. The process of selection is based on the fitness of the individual solutions that include closeness to a perfect solution and deterministic sampling. The closeness to a perfect selection is often done by “roulette selection”, which leads to random selection of a parent with calculated probability based on its fitness. Then the deterministic sampling assigns a value to a subset of a selected organism.

Traditional reproduction is carried out by genetic crossover, which creates the off-spring by exchanging chromosomes from two parents and mutating the chromosome, which randomly alters part of the parent chromosome. The reproduction doesn't occur frequently and introduces new genetic material in the gene pool. Mutation is deemed less significant than crossover in the advancement of the search but is paramount in the maintenance of genetic diversity, which is the key to continued evolution. In steady-state GAs, less fit members are replaced by the new offspring, resulting in higher average fitness. The cycle of reproduction and selection is repeated until a stopping criteria is met; such as maximum fitness is reached or all organisms are identical or evolution is not yielding new results.

By focusing solely on examining fitness and ignoring other derivatives, GAs can balance load and efficacy very well and are computationally highly robust. Another important characteristic of GAs is their ability to indirectly sample a large number of code sequences that are actually tested. GA stores a population of solution instead of adjusting a single solution like most of the stochastic search techniques, which enables them to search through noisy and multimodal relations. Some application examples of GAs include: forecasting air quality, calibration of the water-quality models, estimation of soil bulk density, and water management systems.

## **Cellular Automata**

The Cellular Automata (CA) are dynamic models that are disconnected in space, time, and state. CAs are comprised of regular cell lattices that are capable of interacting with their neighbor lattices. The local rules synchronously update the cell states in time and compute the new state of a cell at “T+1” by utilizing the cell’s own state and the states of the neighboring cells. A major limitation of the CAs pertains to their boundary conditions. A torus is produced when the cells on an edge are made neighbors with cells on the opposite edge in a periodic condition. Alternatively, the boundary conditions can also be reflected and absorbed. This allows the CA to not take off-lattice neighbors in consideration and assign unique values to the border states. These boundary conditions fail to depict the real life scenarios. To avoid this problem, the lattice can be made much bigger than the area under study but it increases the resources and computation expenses.

CAs can be used to simulate complicated physical systems and integrate interactions and spatial variations. These simple mathematical models are also capable of spatial expansion with time. They have high sensitivity to the initial state and transition functions and are often used to understand real environment but have limited capacity to make precise predictions. Some examples of CA applications are: animal migration models, earthquake activity models, fire spread models, and landscape changes .

## Fuzzy Systems

As the name suggests, Fuzzy Systems (FS) work with imprecise and incomplete data by utilizing “fuzzy sets”, where the member objects take value between 0 and 1. This is unlike the “conventional set theory” where the object can or cannot be a member of the set. Therefore, fuzzy sets are capable of describing vague statements into cohesive natural language. For example, a “vegetation model” assigns plant varieties to their kinds, while the “fuzzy membership” indicates the extent to which the vegetation will meet the definition of each kind. The fuzzy inference machine generates the fuzzified data, which is then combined with prior rules to create Fuzzy models. The process called “fuzzification” transforms firm inputs into fuzzy memberships while the fuzzy O/P that has been generated is transformed to a firm numerical value. The fuzzy systems are not the easiest to understand,

but their ability to handle vague and imprecise information like human reasoning makes them stronger than all the other AI techniques. Determination of good membership functions is one of the biggest challenge in the development of a fuzzy system. The biggest limitation of fuzzy systems is their inability to learn and memorize information. Therefore, fuzzy systems are often used in combination with other AI techniques to develop “hybrid systems”. For example, neural networks can be combined with fuzzy systems to generate neuro-fuzzy systems. The incomplete and distorted data in applications like “function approximation, classification/clustering, control, and prediction” can be easily managed by fuzzy systems. Some examples of fuzzy system applications are: estimation of soil hydraulic properties, identification of ecological conditions in a lagoon, by utilizing satellite images to identify oil spills in the oceans, control of waste water treatment and assessment of environmental risks caused by drilling waste discharge.

## **Multi-Agent Systems**

A “Multi-Agent System” is composed of a network of agent that can interact with one another and directed at achieving predefined goals. A software component that contains the data, as well as its code, is called an “agent”. A single agent cannot solve the problem assigned to the multi-agent system on its own. Agents are capable of sharing info, requesting services, and negotiating with one another by utilizing a high level language called “Agent Communication Language” (ACL). “Knowledge Query and Manipulation Language” (KQML) are the most commonly used “Agent Communication Language”. KQML contains a “communication layer” with parameters like senders, recipients and communication identifiers. It also has a “message layer” that can specify the performance and interpretative procedure, as well as a “content layer” containing additional info on the performance procedure.

Message relay and response processing occurs asynchronously between agents. Therefore smooth coordination between agents is critical. System infrastructure dictates the efficiency of agent coordination by governing the flow of information, resource use, the extent of concurrency, and the nature of interactions between agents. A “peer to peer” network where all agents

are capable of communicating directly, makes the simplest infrastructure. Problems that require fast and concurrent processing to find the solution without resolving the conflicts are well suited for the peer to peer infrastructure. Other common multi-agent systems include: “multi-agent blackboard”, which contains a central controller for coordination of agent activity, “public blackboard space”, wherein the data is shared by all agents and “federated agent”, which contains mediators between agents called facilitator agents. Complicated systems with a number of interactions between autonomous and dynamic entities can be modelled by multi-agent systems. Agent organization drives the effectiveness of these systems. The network maintenance is rather cumbersome in peer to peer infrastructure, as every time a new network is added, all the agents are required to be updated. On the other hand, infrastructures with central coordination require the new addition to be updated only in the facilitator’s directory. Another challenge in developing multi-agent system arises from highly complex interactions between the agents and an inherent dynamic tendency, which can lead to conflict between allocation of tasks and resources. Some examples of multi-agent system applications are: management of rangelands, management of irrigated farming systems, management of forest ecosystems undergoing change in land usage, and management of fishing and dairy industries.

## **Swarm Intelligence**

A form of agent based model which draws inspiration from animal colonies like that of the ants and the bees is called Swarm intelligence. Singular agents tend to be simple, but as a swarm, these agents display high intelligence. These models lack centralized control or a global model, so the local interactions create the global patterns making self-organization very critical. Agent to agent or direct communication as well as communication through the environment or indirect communication, can lead to these interactions. An important feature of the Swarm intelligence model is its ability to communicate by altering the environment called “stigmergy”. For example, ants depositing pheromone on their way to the food source so other ants can easily recognize and follow their trail. One of the types of Swarm intelligence algorithm called “Ant Colony Optimization” is based on the behavior of ants. Individually ants move in random directions until

they come across a “pheromone trail” that they are inclined to follow and even reinforce by depositing their pheromones. The trail with the strongest pheromone concentration is more likely to be chosen which results in autocatalytic behavior. High pheromone concentration depicts higher traffic and in turn, becomes more appealing. Shorter paths to the destination also result in the high concentration of pheromone and reinforced as dominant. With the display of autocatalytic behavior, eventually, all ants end up choosing the shortest trail. The Ant Colony Optimization can handle discrete combinatorial optimization. A variety of potential solutions are produced, each representing an individual route with multiple decision nodes. The other frequently used Swarm intelligence algorithm is “Particle Swarm Optimization” (PSO), which draws inspiration from schools of fish and flocks of birds. This system has multiple points each represented by a solution. To initialize the system, a large variety of potential solutions are ‘flown’ through the multidimensional “problem-parameter” space. The fitness that is the positions relative to the goal is evaluated by the solutions at each iteration, and the information of their best position is shared with their swarm. This allows the subsequent solutions to update their speed and location, according to their own best prior location and the location of the best solution within the swarm. Some examples of Swarm Intelligence applications are: project scheduling, pattern recognition, predictive temperature control and optimization of parameters in a rainfall-runoff model.

## **Reinforcement Learning**

Reinforcement Learning can be defined as a type of machine learning that is based on interactions between a “learning agent” and the environment. The agents use the trial and error method to learn how to achieve a goal. The environment, “reinforcement function”, and “value function” are the constituents of a “Reinforcement Learning” problem. The environment has a set of possible states and is dynamic in nature. The reinforcement function has three classes: “pure delayed reward” (reward is given after achieving the terminal state), “minimum time to goal” (agent must find the shortest path to the goal) and “games” (the search for max, min or stationary points of the “reinforcement function”). The game’s “reinforcement function” is utilized when 2 or more agents with counter

goals and actions are selected freely but execution occurs concurrently. Unlike the “reinforcement function” that indicates immediate desirability, the “value function” will determine long term desirability by considering states that are likely to follow an action. Reinforcement learning creates new behavior instead of modeling the existing behavior, which makes it highly suitable for robotics and game playing. With Reinforcement learning, computer programs can progressively improve outcomes and efficiency. Reinforcement learning on its own has limited use in solving the problems of the environment but is being increasingly used in combination with other AI techniques to address other problems. For example, ANN in combination with Reinforcement learning is being used to model the behavior of fishermen and bees.

## **Hybrid systems**

A combination of two or more systems or ‘paradigms’ to overcome individual weaknesses and gain strengths is called the Hybrid system. Based on the combination of the techniques, three main types of hybrid systems are: “sequential, auxiliary and embedded”. In a “Sequential Hybrid System”, output is passed from the first paradigm to the next that generates its own output and the process keeps moving in a sequence. In “Auxiliary Hybrid Systems”, the initial paradigm will obtain some input from the adjacent paradigm to produce the O/P. On the other hand, in an “Embedded Hybrid System”, the 2 paradigms are simply contained within each other. The most common hybrid systems are generated by combining Artificial Neural Networks and Fuzzy Systems, creating neuro-fuzzy systems. The lack of learning capability of the fuzzy systems is compensated by the ANNs and ANN’s inability to attenuate noise in the data is supported by the fuzzy systems. This combination is highly effective, fast and can be easily understood and implemented. Another example of successful hybrid systems is the combination of multi-agent systems with cellular automata. The cellular automata have been used successfully in modeling spatial landscape changes but in combination with the multi-agent systems, the hybrid system is capable of incorporating human decisions that drive these landscape changes. Some other examples of successful hybrid systems are the combination of Artificial Neural Network and Genetic algorithm in water quality management and modeling rainfall-runoff; the combination of

Genetic Algorithm and Particle Swarm Optimization in development of pollution prevention and control strategies; the combination of Artificial Neural Network, Reinforcement learning and Fuzzy systems in optimization of water allocation.

## Statistics

Statistics can be defined as “the science of collecting data, analyzing and drawing of inferences from the data”. Statistics focus primarily on describing the properties of a dataset and establishing relationships between existing data points. Therefore, it's not considered part of Artificial Intelligence. However, many statistical techniques are used as a foundation in advanced machine learning techniques. The 2 main kinds of statistics are “Descriptive Statistics” and “Inferential Statistics”.

- 

“Descriptive Statistics” is utilized to describe the basic features of the data under study. For example, finding the best-selling item in a retail store over a defined period of time.

- 

“Inferential Statistics” draws conclusions that can be applied to more than just the date under study. This is essential when the true population is too large and difficult to study, and analysis can be performed only on a subset of the population. The conclusions reached with this technique are only probabilistic and never quite accurate. For example, Election exit polls rely on surveying a small percentage of citizens to gauge the decisions made by the entire population.

## Probabilistic Programming

Probabilistic programming is a high level programming language that enables the creation of probability models, with the ability to draw values from distributions and condition these values into a program. The Probabilistic programming based learning systems can make inferences from prior knowledge allowing decision making even in the face of

uncertainty. The knowledge of the target system is captured in quantitative and probabilistic terms. With sufficient training, the model can be applied to a specific query to generate an answer via a process called inference. This programming language also solves the probability model automatically, with no external assistance. The Probabilistic programming language allows for unfettered access and reuse of model libraries. It also provides support for interactive modeling and formal verification. The abstraction layer within the probabilistic algorithm is essential to foster generic and efficient inferences since a lot of problems in AI require the agent to work with incomplete and distorted data set. Probabilistic algorithms are increasingly used to filter streams of data to generate predictions and help perception systems to analyze the underlying processes. Some examples of Probabilistic programming applications are: medical imaging, financial predictions, machine perception and atmospheric forecasting. The most widely used probabilistic model is called Bayesian network, which uses Bayesian inference to compute probability. There are two forms in which inference can be made over Bayesian network. The first is the evaluation of the joint probability of a particular assignment of values for each variable in the network. And the second form is finding probability of a subset of variables that have been given assignments of a different subset of variables. Bayesian networks have a wide range of application in solving problems concerning reasoning, learning, planning and perception of the program. The Bayesian inference algorithm can be used to solve reasoning problems while the expectation and maximization algorithm can be used to address learning problems. The dynamic Bayesian networks allow for seamless perception and decision networks can be used to solve planning issues.

## **Review Quiz**

Answer the questions below to verify your understanding of the concepts explained in this chapter. The answer key can be found at the end of the quiz.

1.

Name the machine learning model that uses if and then statements to generate a working memory of the information

to resolve a problem.

2.

\_\_\_\_\_ uses memory recollection to address problems that are similar in nature to a past problem encountered by the system.

3.

The model that has a similar structure as the human brain and utilizes processing units resembling human neurons is called \_\_\_\_\_.

4.

Name the machine learning model that mimics natural selection theory by passing the traits of the fitter solutions to the offspring and replacing the poorer solutions.

5.

The model that results in a torus being produced when the cells on a corner become neighbors with cells on the other corner in a periodic condition.

6.

A vegetation model that can assign plant communities to their kinds and indicate the extent to which the vegetation suits the definition of each kind is an example of \_\_\_\_\_ machine learning model.

7.

The model that is comprised of a network of agent that can interact with one another and achieve a predefined goal is called as \_\_\_\_\_.

8.

Name the machine learning model that has been inspired by animal colonies like that of the ants and the bees.

9.

Name the two main types of statistics methods used on the data being studied for a machine learning model.

10.

The most widely used probabilistic model is called \_\_\_\_\_, which uses \_\_\_\_\_ to compute probability.

## Answer Key

1. Rule-based system
2. Case-based reasoning
3. Artificial Neural Network
4. Genetic Algorithms
5. Cellular Automata
6. Fuzzy systems
7. Multi agent systems
8. Swarm Intelligence
9. Statistics
10. Bayesian network, Bayesian inference

# **Day 3: Supervised Machine Learning Algorithms**

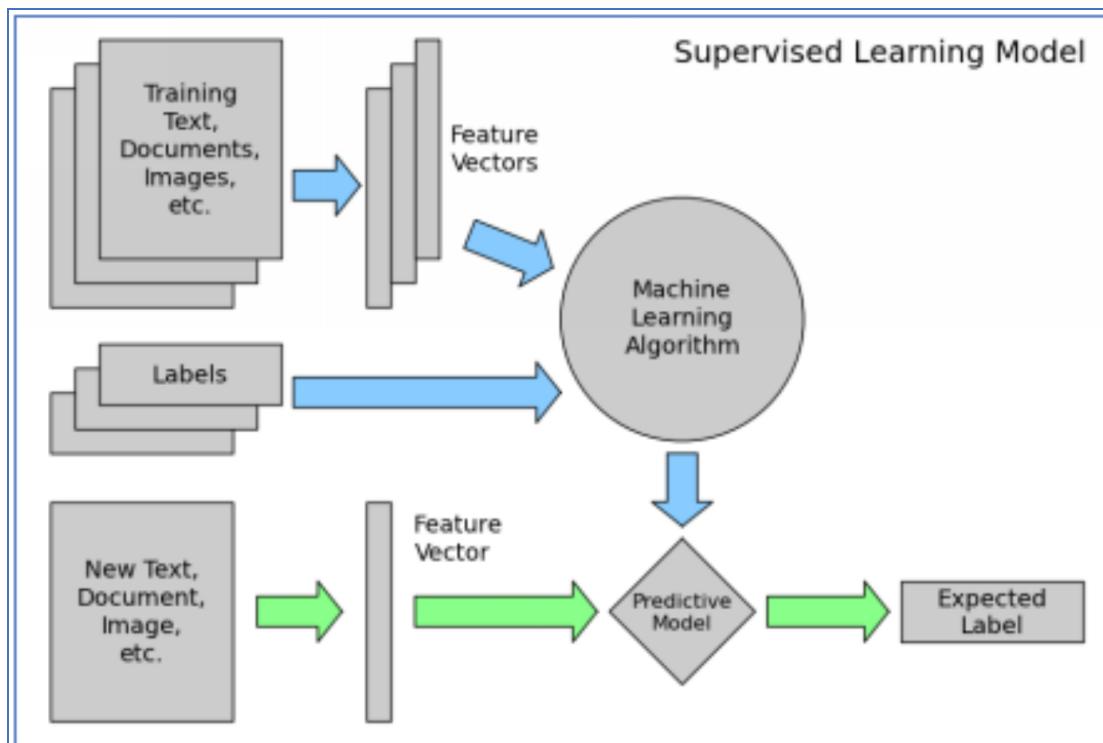
By utilizing prior computations and underlying algorithms, machines are now capable of learning from and training on their own to generate high-quality, readily reproducible decisions and results. The notion of machine learning has been around for a long time now, but latest advances in machine learning algorithms have made large data processing and analysis feasible for computers. This is achieved by applying sophisticated and complicated mathematical calculations by utilizing high speed and frequency automation. Today's advanced computing machines are able to analyze humongous information quantities quickly and deliver quicker and more precise outcomes. Companies utilizing machine learning algorithms have increased flexibility to change the training data set to satisfy their company needs and train the machines accordingly. These tailored algorithms of machine learning enable companies to define potential hazards and possibilities for development. Typically, machine learning algorithms are used in cooperation with artificial intelligence technology and cognitive techniques to create computers extremely efficient and extremely effective in processing large quantities of information or big data and to generate extremely precise outcomes.

There are four fundamental kinds of ML algorithms available today:

## **Supervised machine learning algorithms**

Due to their ability to evaluate and apply the lessons learned from prior iterations and interactions to fresh input data set, the supervised learning algorithms are commonly used in predictive big data analysis. Based on the instructions given to effectively predict and forecast future occurrences, these algorithms can label all their ongoing runs. For instance, people can program the machine as "R" (Run), "N" (Negative), or "P" (Positive) to label its data points. The algorithm for machine learning will then label the input data as programmed and obtain data inputs with the right outputs. The algorithm will compare its own produced output to the "anticipated or correct" output, identifying future changes that can be created and fixing mistakes to make the model more precise and smarter. By using methods

such as "regression", "prediction", "classification" and "gradient boosting" to train the machine learning algorithms well, any new input data can be fed into the machine as a set of "target" data to orchestrate the learning program as desired. This "known training data set" jump starts the analytical process followed by the learning algorithm to produce an "inferred feature" that can be used to generate forecasts and predictions based on output values for future occurrences. For instance, financial institutions and banks rely strongly on monitoring machine learning algorithms to detect fraudulent credit card transactions and predict the probability of a prospective credit card client failing to make their credit payments on time.



### Unsupervised machine learning algorithms

These algorithms are widely used to define how the machine can generate "inferred features" to elucidate a concealed construct from the stack of unlabeled and unclassified data collection. (Details on this will follow in the next chapter)

### Semi-supervised machine learning algorithms

Highly versatile, the "semi-supervised machine learning algorithms" are capable of using both labeled and unlabeled information set to learn from

and train themselves. These algorithms are a "hybrid" of algorithms that are supervised and unsupervised. Typically, with a small volume of labeled data, the training data set is comprised of predominantly unlabeled data. The use of analytical methods, including "forecast", "regression", and "classification", in conjunction with semi-supervised learning algorithms enables the machine to considerably enhance its learning precision and training capabilities. These algorithms are commonly used in instances where it is highly resource intensive and less cost-effective for the business to generate labeled training data set from raw unlabeled data. Companies use semi-supervised learning algorithms on their systems to avoid incurring extra costs of staff and equipment. For instance, implementation for "facial recognition" technology needs a huge amount of facial data distributed across various sources of input. The raw data pre-processing, processing, classification and labeling, acquired from sources such as internet cameras, needs a lot of resources and thousands of hours of job to be used as a training data set.

### **Reinforcement machine learning algorithms**

The "reinforcement machine learning algorithms" are much more distinctive in that they learn from the environment. These algorithms conduct activities and record the outcomes of each action diligently, which would have been either a failure resulting in mistake or reward for good performance. The two primary features that differentiate the reinforcement learning algorithms are: the research method of "trial and error" and feedback loop of "delayed reward". By utilizing a range of calculations, the computer constantly analyzes input data and sends a reinforcement signal for each right or anticipated output to ultimately optimize the end result. The algorithm develops a straightforward action and reward feedback loop to evaluate, record, and learn which actions have been effective and, in a shorter period of time, have resulted in correct or expected output. The use of these algorithms allows the system to automatically determine optimal behaviors and maximize its efficiency within the constraints of a particular context. The reinforcement machine learning algorithms are therefore strongly used in gaming, robotics engineering and navigation systems.

Now, let's look at a few of the most widely used supervised learning algorithms in detail.

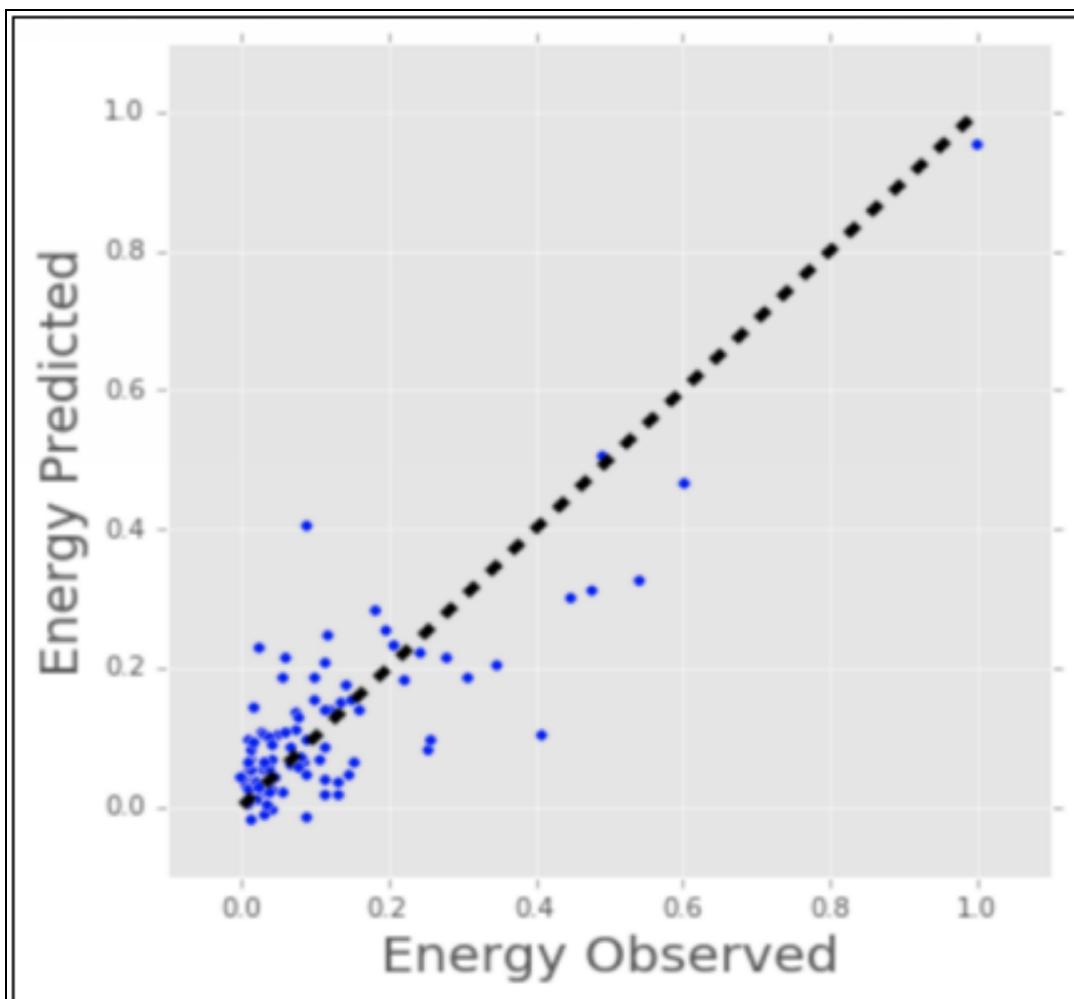
## Regression

The "regression" techniques fall under the category of supervised machine learning. They help predict or describe a particular numerical value based on the set of prior information, such as anticipating the cost of a property based on previous cost information for similar characteristics. Regression techniques vary from simple (such as "linear regression") to complex (such as "regular linear regression", "polynomial regression", "decision trees", "random forest regression" and "neural networks", among others).

The simplest method of all is "**linear regression**", where the line's mathematical equation ( $y = m \cdot x + b$ ) is utilized for modeling the data collection. Multiple "data pairs ( $x, y$ )" can train this model by calculating the position and slope of a line that can decrease the total distance between the data points and the line. In other words, the calculation of the "slope ( $m$ )" and "y-intercept ( $b$ )" is used for a line that produces the highest approximation for data observations.

For example, by utilizing this technique to generate predictions for the energy consumption (in kWh) of houses by collecting the age of the house, no. of bedrooms, square footage area, and the number of installed electronic equipment. Now, we have more than one input (year built, square footage) it is possible to use "linear multi-variable regression". The underlying process is the same as "one-to-one linear regression", however, the line created was based on the number of variables in multi-dimensional space.

The plot below demonstrates how well the model of linear regression fits real construction energy consumption. In case where you could gather house characteristics such as year built and square footage, but you don't understand the house's energy consumption, then you are better off using the fitted line to generate approximations for the house's energy consumption.



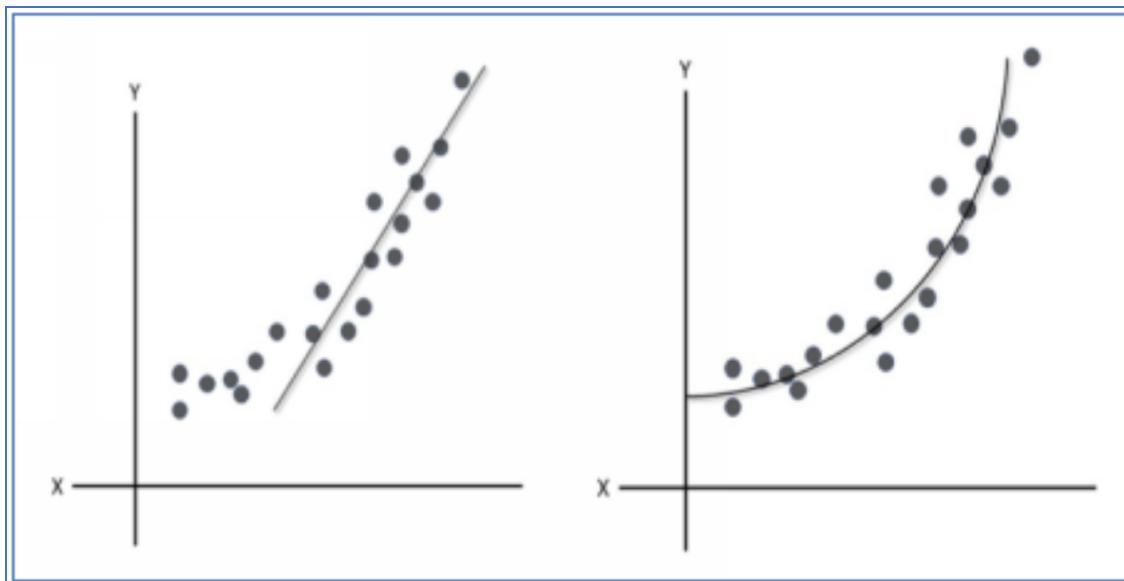
**"Multiple Linear Regression"** tends to be the most common form of regression technique used in data science and the majority of statistical tasks. Just like the "linear regression" technique, there will be an output variable "Y" in "multiple linear regression". However, the distinction now is that we're going to have numerous "X" or independent variables generating predictions for "Y". For instance, a model developed for predicting the cost of housing in Los Angeles will be driven by "multiple linear regression" technique. The cost of housing in Los Angeles will be the "Y" or dependent variable for the model. "X" or the independent variables for this model will include data points such as vicinity to public transport, schooling district, square footage, number of rooms, which will eventually determine the market price of the housing. The mathematical equation for this model can be written as below:

*“housing\_price =  $\beta_0 + \beta_1 sq\_foot + \beta_2 dist\_transport + \beta_3 num\_rooms$ ”*

“Polynomial regression” - Our models developed a straight line in the last two types of regression techniques. This straight line is a result of the connection between “X” and “Y”, which is “linear” and does not alter the influence “X” has on “Y” as the changing values of “X”. Our model will lead in a row with a curve in “polynomial regression” .

If we attempted to fit a graph with non-linear features using “linear regression”, it would not yield the best fit line for the non-linear features. For instance, the graph on the left shown in the picture below has the scatter plot depicting upward trend, but with a curve. A straight line does not operate in this situation. Instead, we will generate a line with a curve to match the curve in our data with a polynomial regression, like the chart on the right shown in the picture below. The equation of a polynomial will appear like the linear equation, the distinction being that one or more of the “X” variables will be linked to some polynomial expression. For instance,

$$\text{“}Y = mX^2 + b\text{”}$$



Another technique of reduction is called “**LASSO regression**” . A very complementary technique to the “ridge regression”, “lasso regression”, promotes the use of simpler and leaner models to generate predictions. In

lasso regression, the model reduces the value of coefficients relatively more rigidly. LASSO stands for the "least absolute shrinkage and selection operator". Data on our scatterplot, like the mean or median values of the data are reduced to a more compact level. We use this when the model is experiencing high multicollinearity similar to the "ridge regression" model.

A hybrid of "LASSO" and "ridge regression" methods is known as "**ElasticNet Regression**". Its primary objective is to further enhance the accuracy of the predictions generated by the "LASSO regression" technique. "ElasticNet Regression" is a confluence of both "LASSO" and "ridge regression" techniques of rewarding smaller coefficient values. All three of these designs are available in the R and Python "Glmnet suite".

"**Bayesian regression**" models are useful when there is a lack of sufficient data, or available data has poor distribution. These regression models are developed based on probability distributions rather than data points, meaning the resulting chart will appear as a bell curve depicting the variance with the most frequently occurring values in the center of the curve. The dependent variable "Y" in "Bayesian regression" is not a valuation but a probability. Instead of predicting a value, we try to estimate the probability of an occurrence. This is regarded as "frequentist statistics", and this sort of statistic is built on the "Bayes theorem". "Frequentist statistics" hypothesize if an event is going to occur and the probability of it occurring again in the future.

"Conditional probability" is integral to the concept of "frequentist statistics". Conditional probability pertains to the events whose results are dependent on one another. Events can also be conditional, which means the preceding event can potentially alter the probability of the next event. Assume you have a box of candies, and you want to understand the probability of withdrawing distinct colors of the candy from the bag. If you have a set of 3 yellow candy and 3 blue candy and on your first draw, you get a blue candy, then with your next draw from the box, the probability of taking out a blue candy will be lower than the first draw. This is a classic example of "conditional probability". On the other hand, an independent event is flipping of a coin, meaning the preceding coin flip doesn't alter the probability of the next flip of the coin. Therefore, a coin flip is not an example of "conditional probability".

## Classification

The method of “classification” is another class of "supervised machine learning", which can generate predictions or explanations for a “class value”. For example, this method can be used to predict if an online customer will actually purchase a particular product. The result generated will be reported as a yes or no response i.e. “buyer” or “not a buyer”. But techniques of classification are not restricted to two classes. A classification technique, for instance, could assist to evaluate whether a specified picture includes a truck or a coupe. The output will be three different values in this case: 1) the picture contains a truck, 2) the picture contains an coupe, or 3) the picture does not contain either a truck or a coupe.

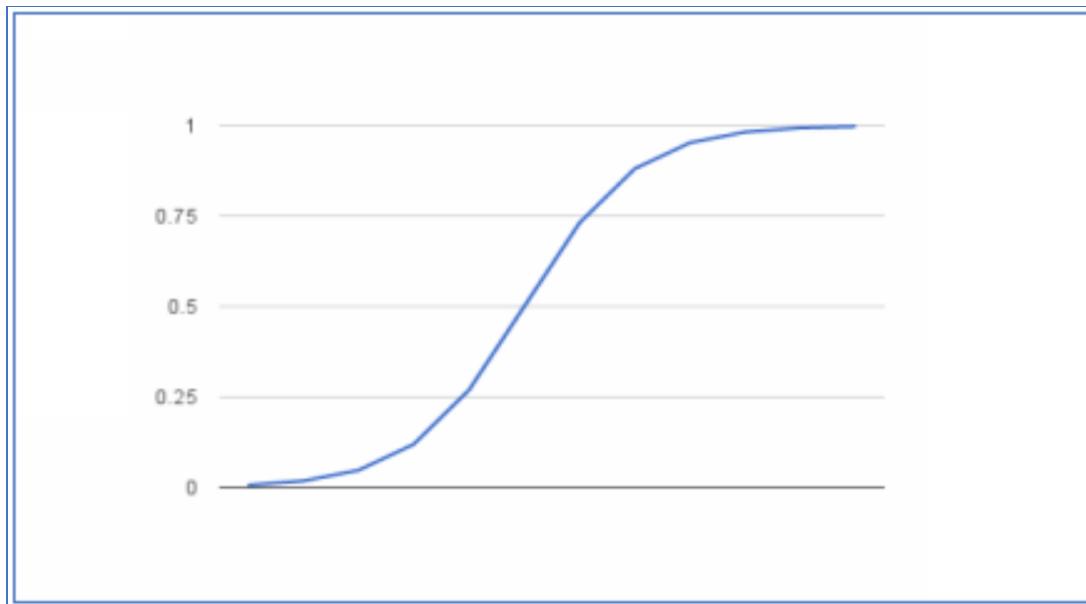
“**Logistic regression**” is considered the easiest classification algorithm, though the term comes across as a “regression” technique that is far from reality. “Logistic regression” generates estimations for the likelihood of an event taking place based on single or multiple input values. For example, to generate estimation for the likelihood of a student being accepted to a specific university, a “logistic regression” will use the standardized testing scores and university testing score for a student as inputs. The generated prediction is a probability, ranging between ‘0’ and ‘1’, where ‘1’ is complete assurance. For the student, if the estimated likelihood is greater than 0.5, then the prediction would be that they will be accepted. If the projected probability is less than 0.5, the prediction would be that they will be denied admission. Logistic regression enables the creation of a line graph that can represent the “decision boundary”.

The “Logistic regression” technique has been borrowed by ML technology from the world of statistical analysis. It is widely used for binary classification tasks that involve two different class values. Logistic regression is so named owing to the fundamental statistical function at the root of this technique called the "logistic function". Statisticians created the "logistic function", also called the "sigmoid function", to define the attributes of population growth in ecosystems which continues to grow rapidly and nearing the maximum carrying capacity of the environment. The logistic function is “an S-shaped curve capable of taking any real-valued integer and mapping it to a value between '0' and '1',

but never precisely at those boundaries, where ‘e’ is the base of the natural log (Euler’s number or the EXP)” and the numerical value that you are actually going to transform is called the ‘value.’

$$“1 / (1 + e^{-value})”$$

Here is a graph of figures ranging from “-5 and 5”, which has been transformed by the logistic function into a range between 0 and 1.



Similar to the "linear regression" technique, "logistic regression" utilizes an equation for data representation.

Input values (X) are grouped linearly to forecast an output value (Y), with the use of weights or coefficient values (presented as the symbol "Beta"). It is mainly different from the "linear regression" because the modeled output value tends to be binary (0 or 1) instead of a range of values.

Below is an example of the "logistic regression" equation, where "the single input value coefficient (X) is represented by 'b1', the "intercept or bias term' is the 'bo,' and the "expected result" is 'Y.' Every column in the input data set has a connected coefficient "b" with it, which should be understood by learning the training data set. The actual model representation, which is stored in a file or in the system memory, would be "the coefficients in the equation (the beta values)".

$$y = e^{(b_0 + b_1 * x)} / (1 + e^{(b_0 + b_1 * x)})$$

The "logistic regression" algorithm's coefficients (the beta values) must be estimated on the basis of the training data. This can be accomplished by utilizing another statistical technique called "maximum-likelihood estimation", which is a popular ML algorithm using a multitude of other ML algorithms. "Maximum-likelihood estimation" works by making certain assumptions about the distribution of the input data set.

An ML model that can predict a value nearer to "0" for the "other class" and a value nearer to "1" for the "default class" can be obtained by employing the best coefficients of the model. The underlying assumption for most likelihood of the "logistic regression" technique is that "a search procedure attempts to find values for the coefficients that will reduce the error in the probabilities estimated by the model pertaining to the input data set (e.g. probability of '0' if the input data is not the default class)".

Without going into mathematical details, it is sufficient to state that you will be using a minimization algorithm for optimization of the values of the most relevant coefficients from the training dataset. In practice, this can be achieved with the use of an effective "numerical optimization algorithm", for example, the "Quasi-newton" technique.

### **Generating predictions using logistic regression**

In here, you can simply plug in the measurements into the "logistic regression" equation and calculate the outcome to generate predictions with the "logistic regression" model. Let's take a look at an example to solidify this concept. Let's assume there is a model that is capable of generating predictions if an individual is masculine or woman depending on with fictitious values of their height. If the value of the height for an individual is set as 150 cm, would the individual be predicted as a male or female? Assuming we have already discovered the values of coefficients "b0= -100" and "b1= 0.6". By leveraging the above equation, the probability of male with a height of 150 cm or "P(male|height=150)" can be easily calculated. The function EXP() will be used for "e" because if you log this instance into your spreadsheet, this is what you can use:

$$y = e^{(b_0 + b_1 * X)} / (1 + e^{(b_0 + b_1 * X)})$$

$y = \exp(-100 + 0.6*150) / (1 + \exp(-100 + 0.6*X))$

$y = 0.0000453978687$

Or a near "0" probability male is the gender of that specific person.

In theory, probability can simply be used. But since this is a "classification" algorithm and we want a sharp outcome, the probabilities can be tagged on to a binary class value. For instance, the model can predict "0" if "p (male) < 0.51" and predict "1" if "p (male) >= 0.5". Now that you know how predictions can be generated by utilizing "logistic regression", you can easily pre-process the training data set to get the most out of this technique. The assumptions made pertaining to the distribution and relations within the data set by the "logistic regression" technique are nearly identical to the assumptions made in the "linear regression" technique.

A lot of research has been done to define these hypotheses and to use accurate probabilistic and statistical language. It is recommended to use these as thumb rules or directives and try with various processes for data preparation.

The ultimate goal in "predictive modeling" machine learning initiatives is the generation of highly accurate predictions rather than analysis of the outcomes. Considering everything, some assumptions could be broken if the designed model is stable and has high performance.

•

**"Binary Output Variable":** This may be evident as we have already discussed it earlier, but "logistic regression" is designed specifically for issues with "binary (two-class) classification". This will generate predictions for the probability of a default class instance that can be tagged into a classification of "0" or "1".

•

**"Remove Noise":** Logistic regression does not assume errors in the "output variable ('y')", therefore, the "outliers and potentially misclassified" cases should be removed from the training data set.

- 

**“Gaussian Distribution”:** Logistic regression can be considered as a type of “linear algorithm but with a non-linear transform on the output”. A liner connection between the output and input variables is also assumed. Data transformations of the input variables may lead to a more accurate model with a higher capability of revealing the linear relationships of the data set. For instance, to better reveal these relationships, we could utilize "log", "root", "Box-Cox" and other single variable transformations.

- 

**“Remove Correlated Inputs”:** If you have various highly correlated inputs, the model could potentially be “over-fit” similar the "linear regression" technique. To address this issue, you can “calculate the pairwise correlations between all input data points and remove the highly correlated inputs”.

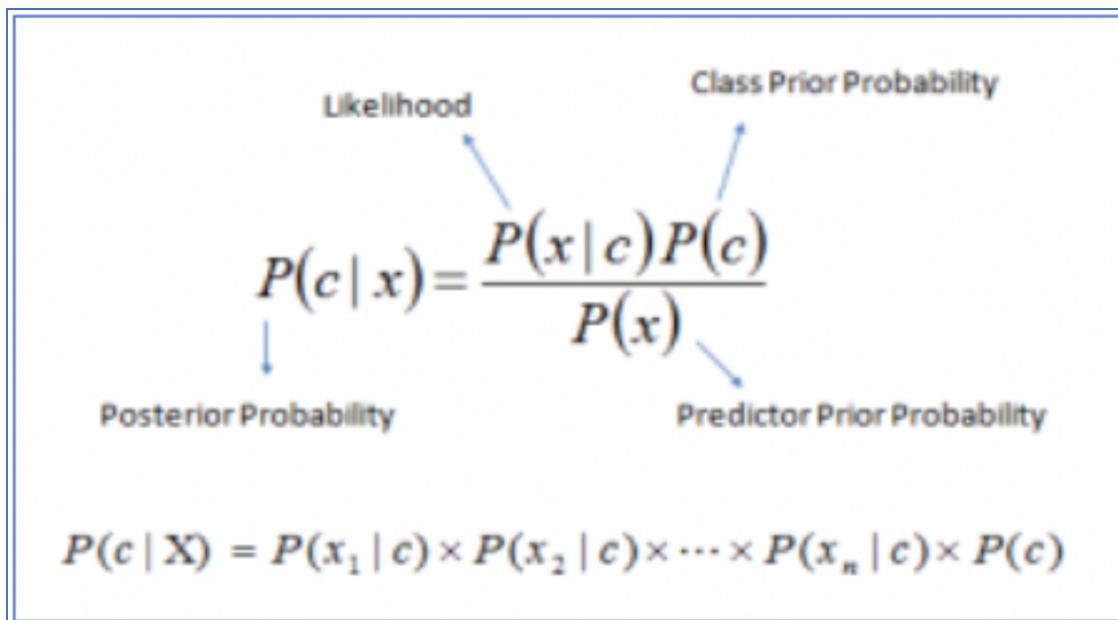
- 

**“Failure to converge”:** It is likely for the "expected likelihood estimation" method that is trained on the coefficients that have failed to converge. It could occur if the data set contains several highly correlated inputs, or there is a very limited data (e.g. loads of "0" in the input).

**“Naïve Bayes classifier algorithm”** is another “classification” learning algorithm with a wide variety of applications. It is a method of classification derived from the "Bayes theorem", which assumes predictors are independent of one another. The "Naïve Bayes classifier" will assume that “all the features in a class are unrelated to the existence of any other feature in that class”. For instance, if input data has an image of a fruit that is green, round, and about 10 inches in diameter, the model can consider the input to be a watermelon. Although these attributes rely on one another or on the presence of a specific feature, all of the characteristics contribute freely to the probability that the image of the fruit is that of a watermelon, hence it is referred to as "Naive". “Naïve Bayes model” for large volumes of datasets is relatively easy to construct and extremely effective.

“Naïve Bayes” has reportedly outperformed even the most sophisticated techniques of classification, even considering the ease of its development. “Bayes theorem” can also provide the means to calculate the posterior probability “ $P(c|x)$ ” using “ $P(c)$ ,  $P(x)$ , and  $P(x)$ ”. On the basis of the equation shown in the picture below, where the probability of “c” can be calculated if “x” has already occurred.

“ $P(c|x)$ ” is the posterior probability of “class (c, target)” provided by the “predictor (x, attributes)”. “ $P(c)$ ” is the class's previous probability. “ $P(x|c)$ ” is the probability of the class provided by the predictor. “ $P(x)$ ” is the predictor's prior probability.



Here is an example to better explain the application of the “Bayes Theorem”. The picture below represents the data set pertaining to the problem of identifying suitable weather days to play golf. The columns depict the weather features of the day and the rows contain individual entries. Considering the first row of the data set, it can be concluded that the weather will be too hot and humid with rain so the day is not suitable to play golf. Now, the primary assumption here is that all these features or predictors are independent of one another. The other assumption being made here is that all the predictors have potentially the same effect on the end result. Meaning, if the day was windy, it would have some relevance to the decision of playing golf as the rain. In this example, the variable (c) is

the class (playing golf) representing the decision if the weather is suitable for golf and variable (x) represents the features or predictors.

	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY GOLF
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

### Types of "Naïve Bayes classifier".



**"Multinomial Naïve Bayes"** - This is widely used to classify documents, for example, which category does a document belong: beauty, technology, politics, and so on. The frequency of the phrases in the document is considered as the features or predictors of the classifier.



**"Bernoulli Naive Bayes"** - This is nearly identical to the "Multinomial Naïve Bayes", however, the predictors used here are the "boolean variables". For example, depending on

whether a select phrase occurs in the text or not, the parameters used to predict the class variable can either be a yes or no value.

- 

**“Gaussian Naive Bayes”** - When the predictors are not distinct and have very similar or continuous values, it can be assumed that these values are obtained from a Gaussian distribution.

## **Applications of “Naïve Bayes”**

- 

**“Real-Time Prediction”:** Naive Bayes is extremely quick in learning from the input data and can be seamlessly used to generate real-time predictions.

- 

**“Multi-class Prediction”:** This algorithm is widely used to generate predictions for multiple classes at the same time. It allows the prediction of the probability of various classes of the target variable.

- 

**“Text classification / Spam Filtering / Sentiment Analysis”:** The “Naive Bayes classifiers” is heavily utilized in text classification models owing to its ability to address problems with multiple classes of the target variable and the rule of autonomy. This algorithm has reported higher success rates than any other algorithm. As a consequence, it is commonly used for the identification of spam emails and sentiment analysis by identifying favorable and negative consumer feelings on the social media platforms.

- 

**“Recommendation System”:** "Naive Bayes Classifier" and "Collaborative Filtering" can be combined together to generate a this system that utilizes ML and data mining methods to filter hidden data and generate insight as to whether the customer would prefer a particular item or product.

## **Ensemble Methods**

Imagine that you've chosen to construct a car because you're not pleased with the variety of cars available in stores and online. You may start by discovering the best option for each component that you need. The resulting car will outshine all the other alternatives with the assembly of all these excellent components.

Ensemble methods use the same concept of mixing several predictive models (controlled ML) to obtain results of greater quality than any of the models could provide on their own. The “Random Forest” algorithms, for instance, is an ensemble technique that combines many trained “Decision Trees” with various data sets samples. As a result, the quality of predictions generated by “Random Forest” method is higher than the quality of the estimated predictions with a single “Decision Tree”.

Think of ensemble methods as an approach for reducing a single machine learning model's variance and bias. This is essential because, under certain circumstances, any specified model may be accurate but completely incorrect under other circumstances. The relative accuracy could be overturned with another model. The quality of the predictions is balanced by merging the two models.

### **Decision Trees**

To refresh your memory, a machine learning decision tree can be defined as “a tree like graphical representation of the decision making process by taking into consideration all the conditions or factors that can influence the decision and the consequences of those decisions”. Decision trees are considered one of the simplest “supervised machine learning algorithms” and have three main elements: “branch nodes” representing conditions of the data set, “edges” representing ongoing decision process and “leaf nodes” representing the end of the decision.

There are two types of decision trees: “Classification tree” that is used to classify Data for information on the basis of existing system data available in the system; “Regression tree”, which is used to make a forecast for predictions for future events on the basis of existing system data. Both of these trees are heavily used in machine learning algorithms. A widely used

terminology for decision trees is “Classification and Regression trees” or “CART”.

### **Support vector machine learning algorithm (SVM)**

This is a type of “supervised machine learning algorithm”, used for “classification” or “regression”, wherein the dataset teaches SVM about classes to allow classification of any new data. It operates by classifying the data into various classes by discovering a line (hyper-plane) that divides the collection of training data into classes. Due to the availability of various linear hyper-planes, this algorithm attempts to maximize the distance between the different classes involved, which is known as “margin maximization”. By identifying the line that maximizes the class distance, the likelihood of generalizing apparent to unseen data can be improved.

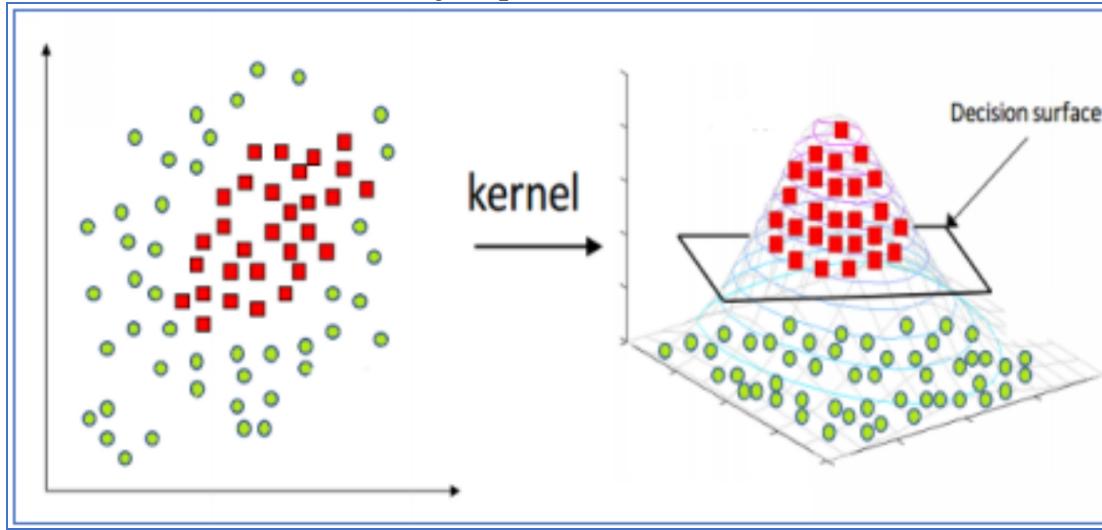
SVM's can be categorized into two as follows:

- “Linear SVM's” – In linear SVM's, the training data or classifiers can be divided by a hyper-plane.
- “Non-Linear SVM's” – Unlike linear SVMs, in non-linear SVM's the possibility to separate the training data with a hyper-plane does not exist. For example, the Face Detection training data consists of a group of facial images and another group of non-facial images. The training data is highly complicated under such circumstances that it is difficult to discover a representation for each feature vector. It is extremely complex to separate the set of faces linearly from the set of non-facial data.

SVM is widely used by different economic organizations for stock market forecasting. For example, SVM may be utilized for comparison of relative stock performances, in comparison to the performances of other stocks within an industrial sector. Relative stock comparisons help in the investments related decision making process, on the basis of the classifications generated by SVM algorithms.

### **The Kernel Trick**

The data collected in the real world is randomly distributed and making it too difficult to separate different classes linearly. However, if one can potentially figure out a way for mapping the data from 2-D space to 3-D space, as shown in the picture below, they would be able to discover a decision surface that obviously separates distinct classes.



One approach to transforming data like this is mapping all data points to a higher dimension (for this example, it is 3 dimensions), finding the limit, and making the classifications. That works for a limited number of dimensions but computations within a given space becomes increasingly costly when there are a lot of dimensions to deal with. And so the kernel trick comes to the rescue! The “kernel trick” enables us to function in the original feature space without needing to calculate the data coordinates in a higher dimensional space. For example, the equation in the picture below has a couple of 3-D data points as ‘x’ and ‘y.’

$$\mathbf{x} = (x_1, x_2, x_3)^T$$

$$\mathbf{y} = (y_1, y_2, y_3)^T$$

Suppose we want to map “x” and “y” to 9-D space. To get the final outcome, which would be just scalar, we have to do the calculations shown in the picture below. In this case, the computational complexity will be  $O(n^2)$ .

$$\phi(\mathbf{x}) = (x_1^2, x_1x_2, x_1x_3, x_2x_1, x_2^2, x_2x_3, x_3x_1, x_3x_2, x_3^2)^T$$

$$\phi(\mathbf{y}) = (y_1^2, y_1y_2, y_1y_3, y_2y_1, y_2^2, y_2y_3, y_3y_1, y_3y_2, y_3^2)^T$$

$$\phi(\mathbf{x})^T \phi(\mathbf{y}) = \sum_{i,j=1}^3 x_i x_j y_i y_j$$

However, by using the “kernel function”, indicated as “ $k(\mathbf{x}, \mathbf{y})$ ”, in lieu of performing the complex calculations in the 9-D space, the same outcome can be achieved in the 3-D space with calculation of the “dot product” of ‘ $\mathbf{x}$ -transpose’ and ‘ $\mathbf{y}$ .’ In this case, the complex computations will be “ $O(n)$ ”.

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y})^2$$

$$= (x_1y_1 + x_2y_2 + x_3y_3)^2$$

$$= \sum_{i,j=1}^3 x_i x_j y_i y_j$$

In principle, this trick is used to make the transformation of data into higher dimensions much more effective and less costly. The use of this trick is not restricted to the SVM algorithm and can be utilized with any computations that involve the “dot products ( $\mathbf{x}, \mathbf{y}$ )”.

## Review Quiz

Answer the questions below to verify your understanding of the concepts explained in this chapter. The answer key can be found at the end of the quiz.

1.

What are the different types of machine learning algorithms?

2. Name the type of algorithm that is widely used for predictive data analysis.
3. Name the type of algorithm that are capable of utilizing raw data just as well as the labeled data.
4. What are the 2 features that are unique to Reinforcement algorithms?
5. \_\_\_\_\_ is one of the supervised algorithms that can be trained with multiple by calculating a line's position and slope.
6. \_\_\_\_\_ is another supervised algorithm that can generate predictions or explanations for a class value.
7. Define Naïve Bayes classifier algorithm.
8. The most popular algorithm used by different economic organizations for stock market forecasting.

## Answer Key

1. Supervised, Unsupervised, Semi-supervised, Reinforcement
2. Supervised machine learning algorithms
3. Semi-supervised machine learning algorithms
4. Trial and error research method, Delayed reward feedback loop
5. Linear regression
6. Classification

7.

It is a method of classification derived from the Bayes theorem, which assumes predictors are independent of one another.

8.

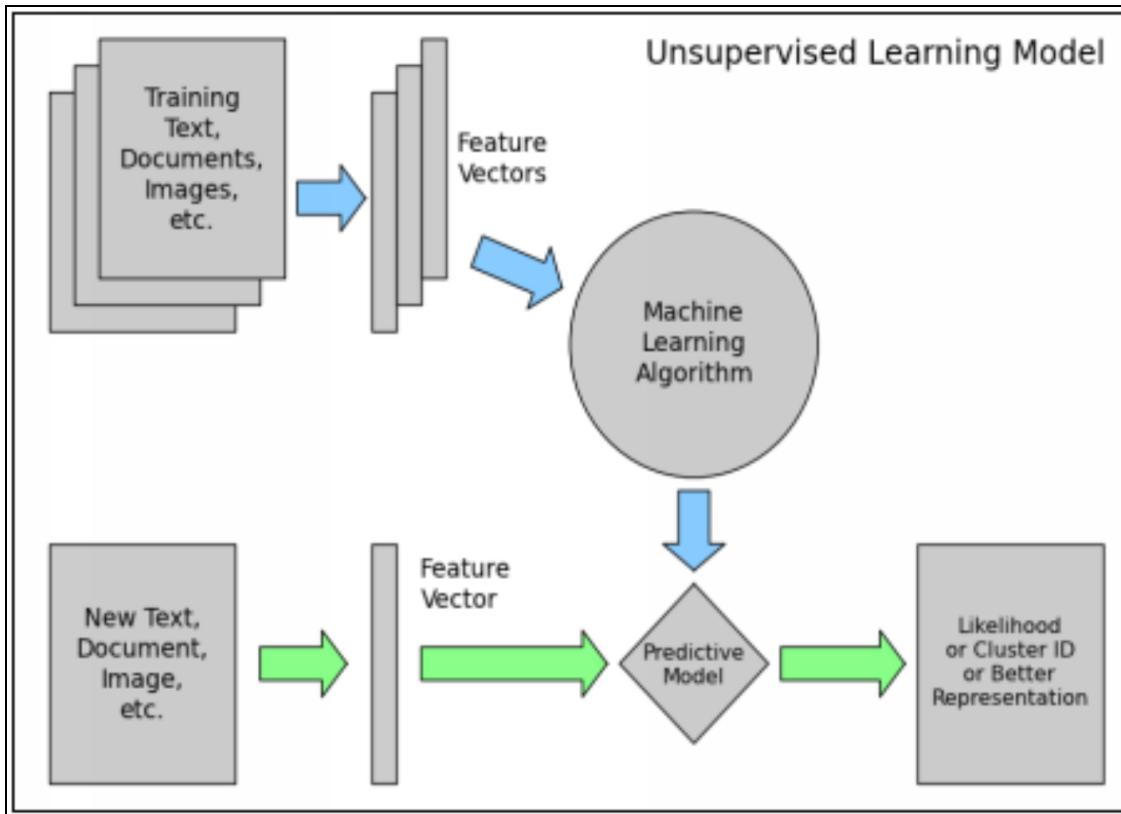
Support vector machine learning algorithm

## **Day 4: Unsupervised Machine Learning Algorithms**

Working with a large volume of raw and unstructured data can be very challenging. Companies frequently run into issues of not being able to access the required information from the source, which prevents them from generating a labeled and classified training data set to develop the desired machine learning model. This is where the unsupervised machine learning algorithms come to the rescue.

Unsupervised ML algorithms have a wide array of usage in defining how the machines can generate "inferred features" for the elucidation of a concealed construct from the stack of unlabeled and unclassified data collection. These algorithms are capable of exploring the data in order to define a structure within the data mass. Unlike the supervised machine learning algorithms, unsupervised algorithms are unable to identify the correct output, even though they have similar efficacy as the supervised learning algorithms in investigating input data and drawing inferences. These algorithms can be used to identify information outliers, generate tailored and custom product recommendations, classify text subjects by utilizing methods such as "self-organizing maps", "singular value decomposition" and "k-means clustering".

For instance, these algorithms are being utilized to identify customers with shared shopping characteristics which are then segmented into specific groups with a singular focus based marketing strategies and campaigns. As a result, the online marketing world is heavily dependent on unsupervised learning algorithms.



Some of the most widely used unsupervised algorithms are described below:

## Clustering

We enter the category of unsupervised machine learning, with “clustering methods” because its objective is to “group or cluster observations with comparable features”. Clustering methods do not use output data to train but allow the output to be defined by the algorithm. Only data visualizations can be used in clustering techniques to check the solution's quality.

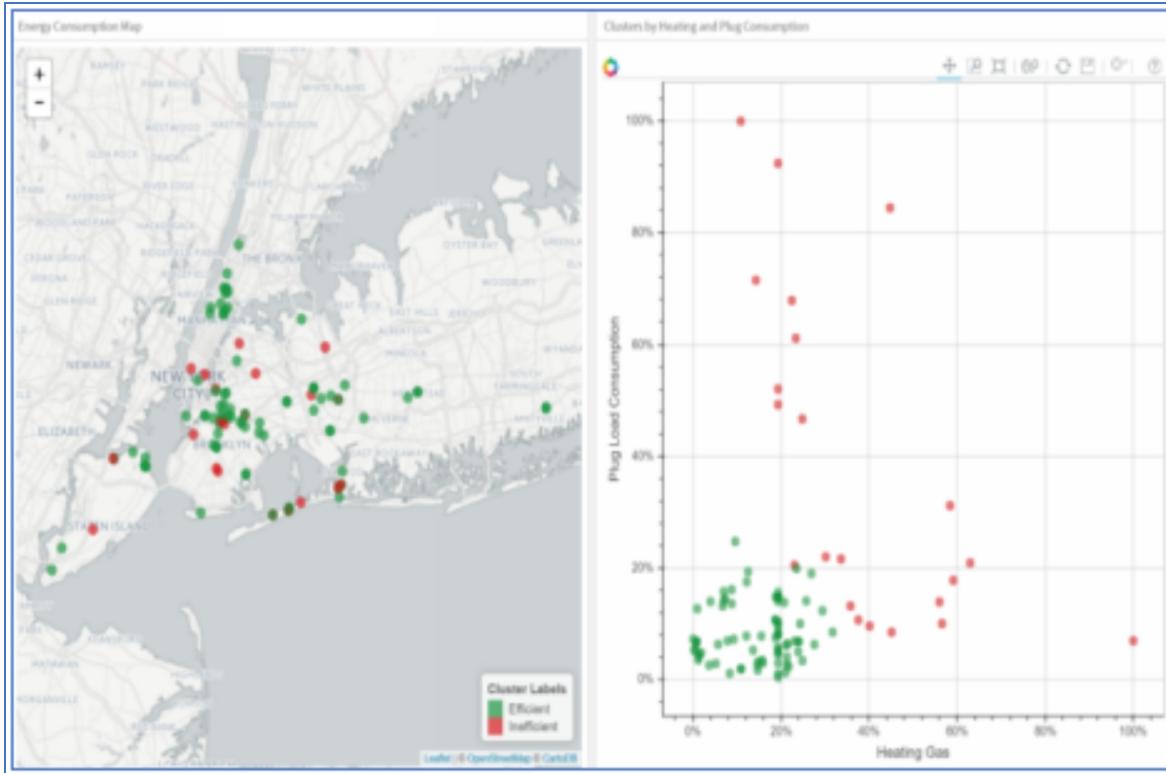
“K-Means clustering”, where ‘K’ is used to represent the no. of “clusters” that the customer elects to generate a machine learning model and is the most common clustering method. (Note that different methods for selecting K value, such as the “elbow technique” are also available.)

Steps used by K-Means clustering to process the data points:

1. The data centers are selected randomly by ‘K.’
2. Assigns each data point to the nearest centers that have been randomly generated.
3. Re-calculates each cluster's center.
4. If centers do not change (or have minor change), the process will be completed.

Otherwise, we'll go back to step 2. (Set a maximum amount of iterations in advance to avoid getting stuck in an infinite loop, if the center of the cluster continues to alter.)

The following plot applies “K-Means” to a building data set. Each column in the plot shows each building's efficiency. The four measurements relate to air conditioning, heating, installed electronic appliances (refrigerators, TV), and cooking gas. For simplicity of interpretation of the results, ‘K’ can be set to value ‘2’ for clustering, wherein one cluster will be selected as an efficient building group and the other cluster as an inefficient building group. You see the place of the structures on the left as well as a couple of the building characteristics used as inputs on the right: installed electronic appliances and heating.



## Dimensionality Reduction

As the name indicates, to extract the least significant information (sometimes redundant columns) from a data set, we use “dimensionality reduction”. In practice, data sets tend to contain hundreds and thousands of rows (also known as characteristics), which makes it essential to decrease the total number of rows. For example, pictures may contain thousands of pixels; all those pixels are not important for the analysis. Or a large number of measurements or experiments can be applied to every single chip while testing microchips within the manufacturing process, the majority of which produce redundant data. In such scenarios, “dimensionality reduction” algorithms are leveraged to manage the data set.

### Principal Component Analysis (PCA)

It is the most common “dimension reduction technique”, which reduces the size of the “feature space” by discovering new vectors that are capable of maximizing the linear variety of the data. When the linear correlations of the data are powerful, PCA can dramatically reduce the data dimension while maintaining most of the information. PCA is one of the fundamental

ML algorithms. It enables you to reduce the data dimension, losing as little info as possible. It is used in many fields such as object recognition, the vision of computers, compression of information, etc.

The calculation of the main parts is limited to the calculation of the initial data's own vectors and covariance matrix values or to the data matrix's unique decomposition. Through one, we can convey several indications, merge, so to speak, and operate with a simpler model already. Of course, most probably, data loss will not be avoided, but the PCA technique will assist us to minimize any losses.

### **t-Stochastic Neighbor Embedding (t-SNE)**

Another common technique is “t-Stochastic Neighbor Embedding (t-SNE)”, which results in a decrease of non-linear dimensionality. This technique is primarily used for data visualization, with potential use for machine learning functions such as space reduction and clustering.

The next plot demonstrates “MNIST database” analysis of handwritten digits. “MNIST” includes a large number of digit pictures from 0 to 9, used by scientists to test “clustering” and “classification” algorithms. Individual row of the data set represents “vectorized version” of the original picture (size 28x28 = 784 pixels) and a label (0, 1, 2, and so on) for each picture. Note that the dimensionality is therefore reduced from 784 pixels to 2-D in the plot below. Two-dimensional projecting enables visualization of the initial high-dimensional data set.



## Transfer Learning

Imagine you are a data scientist focusing on the clothing industry. You have been training a high-quality learning model for months to be able to classify pictures of “women’s tops” as tops, tank tops, and blouses. You have been tasked to create a comparable model for classification of pants pictures such as jeans, trousers and chinos. With the use of the “Transfer Learning” method, the understanding incorporated into the first model be seamlessly transferred and applied to the second model .

Transfer Learning pertains to the re-use and adaptation of a portion of a previously trained neural network to a fresh but comparable assignment. Specifically, once a neural network has been successfully trained for a particular task, a proportion of the trained layers can be easily transferred and combined with new layers that are then trained on pertinent data for the new task. This new “neural network” can learn and adapt rapidly to the new assignment by incorporating a few layers.

The primary benefit of transferring learning is a decrease in the volume of data required to train the neural network resulting in cost savings for the development of “deep learning algorithms”. Not to forget how hard it can be to even procure a sufficient amount of labeled data required for training the model.

Suppose in this example; you are utilizing a neural network with 20 hidden layers for the “women’s top” model. You understand after running a few tests that 16 of the women’s top model layers can be transferred and combined them with new set of data to train on pants pictures. Therefore, the new pants model will have 17 concealed layers. The input and output of both the tasks are distinct, but the reusable layers are capable of summarizing the data appropriate to both, e.g. clothing, zippers, and shape of the garment.

Transfer learning is getting increasingly popular, so much so that for basic “deep learning tasks” such as picture and text classification, a variety of high quality pre-trained models are already available in the market.

## **Natural Language Processing**

A majority of the knowledge and information pertaining to our world is in some type of human language. Once deemed as impossible to achieve, today, computers are capable of reading large volumes of books and blogs within minutes. Although computers are still unable to fully comprehend “human text”, but they can be trained to perform specific tasks. Mobile devices, for instance, can be trained to auto-complete text messages or fix spelling mistakes. Machines have been trained enough to hold straightforward conversations like humans.

“Natural Language Processing” (NLP) is not exactly a method of ML, instead it is a commonly used technique to produce texts for machine learning. Consider multitude of formats of tons of text files (words, internet blogs etc.). Most of these text files are usually flooded with typing errors, grammatically incorrect characters and phrases that need to be filtered out. The most popular text processing model available in the market today is

“NLTK (Natural Language ToolKit)”, developed by “Stanford University” researchers.

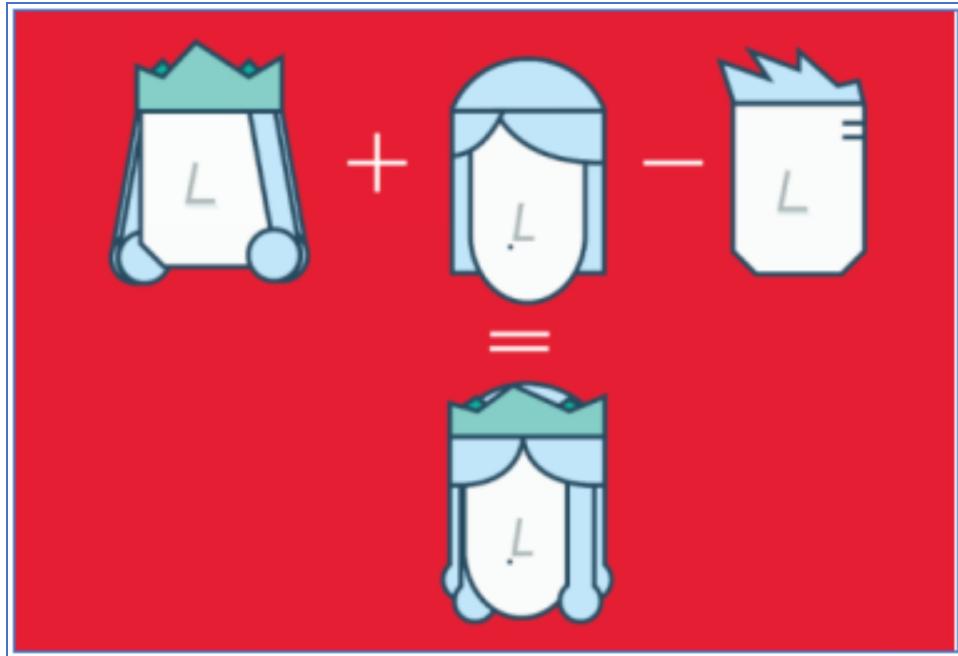
The easiest approach to map texts into numerical representations is calculation of the frequency of each word contained in every text document. For example, an integer matrix where individual rows represent one text document and every column represents single word. This word frequency representation matrix is frequently referred to as the “Term Frequency Matrix” (TFM). From there, individual matrix entries can be separated by a weight of how essential every single term is within the whole stack of papers. This form of matrix representations of text documents is called as “Term Frequency Inverse Document Frequency” (TFIDF), which usually yields better performance for machine learning tasks .

## Word Embedding

“Term Frequency Matrix” and “Term Frequency Inverse Document Frequency” are numerical representations of text papers which only take into account frequency and weighted frequencies to represent text files. On the other hand, “Word Embedding” in a document is capable of capturing the actual context of a word. Embedding can quantify the similarity between phrases within the context of the word, which subsequently allows execution of arithmetic operations with words.

“Word2Vec” is a neural network based technique that can map phrases to a numerical vector in a corpus. These vectors are then used to discover synonyms, do arithmetic with words or phrases, or to represent text files. Let's suppose, for instance, a large enough body of text files was used to estimate word embedding. Suppose the words “king, queen, man and female” are found in the corpus and vector (“word”) is the number vector representing the word “word”. We can conduct arithmetic procedure with numbers to estimate vector('woman'):

$$\text{vector}(\text{'king'}) + \text{vector}(\text{'woman'}) - \text{vector}(\text{'man'}) \sim \text{vector}(\text{'queen'})$$



Word depictions enable similarities to be found between phrases by calculating the “cosine similarity” between the vector representation of the two words. The “cosine similarity” gives a measure of the angle between two vectors.

We use machine learning techniques to calculate word embedding, but this is often a preliminary step in implementing a machine learning algorithm on top of the word embedding method. For example, the “Twitter” user database containing large volume of “tweets” can be leveraged to understand which of these customers purchased a house recently. We can merge “Word2Vec” with a logistic regression to generate predictions on the likelihood of a new “Twitter” user purchasing a home.

## **Apriori machine learning algorithm**

“Apriori algorithm” is another unsupervised ML algorithm that can produce rules of association from a specified set of data. “Association rule” simply means if an item X exists then the item Y has a predefined probability of existence. Most rules of association are produced in the format of “IF-THEN” statements. For instance, “IF” someone purchases an iPhone, “THEN” they have most likely purchased an iPhone case as well. The Apriori algorithm is able to draw these findings by initially observing

the number of individuals who purchased an iPhone case while making an iPhone purchase and generating a ratio obtained by dividing the number individuals who bought a new iPhone (1000) with individuals who also bought an iPhone case (800) with their new iPhones. The fundamental principles of Apriori ML Algorithm are:

- 

- If a set of events have high frequency of occurrence, then all subsets of that event set will also have high frequency of occurrence.
- If a set of events occur occasionally, then all supersets of the event set of will occur occasionally as well.

Apriori algorithm has wide applicability in following areas:

#### **“Detecting Adverse Drug Reactions”**

“Apriori algorithm” is used to analyze healthcare data such as the drugs administered to the patient, characteristics of each patient, harmful side effects experienced by the patient, the original diagnosis, among others. This analysis generates rules of association that provide insight on the characteristic of the patient and the administered drug that potentially contributed to harmful side-effects of the drug.

#### **“Market Basket Analysis”**

Some of the leading online e-commerce businesses including “Amazon”, use Apriori algorithm to gather insights on products that have high likelihood of being bought together and products that can have an upsell with product promotions and discount offers. For instance, Apriori could be used by a retailer to generate prediction such as: customers purchasing sugar and flour have high likelihood of purchasing eggs to bake cookies and cakes.

#### **“Auto-Complete Applications”**

The highly cherished auto-complete feature on “Google” is another common Apriori application. When the user starts typing in their keywords for a search, the search engine searches its database, for other related phrases that are usually typed in after a particular word.

## **Review Quiz**

Answer the questions below to verify your understanding of the concepts explained in this chapter. The answer key can be found at the end of the quiz.

1. \_\_\_\_\_ algorithms can be used to define how the machines can generate inferred from raw data.
2. How are the machines trained with the use of the clustering method?
3. Name the method used to represent the number of clusters elected by the customer to generate a machine learning model.
4. What algorithm is used for extraction of the least significant information or redundant columns from a data set.
5. Which dimensionality reduction technique may be utilized to reduce the size of the feature space by maximizing the linear variety of the data?
6. Which dimensionality reduction technique may be utilized to reduce the non-linearity of the data?
7. Define the transfer learning method.
8. Name the technique used to produce text based data for machine learning models such as auto-complete feature on mobile messaging applications.

## Answer Key

1. Unsupervised
- 2.

Clustering methods use output data to train but allow the output to be defined by the algorithm on the basis of group observations generated on comparable features.

3. K-means clustering
4. Dimensionality reduction
5. Principal Component Analysis
6. t-Stochastic Neighbor Embedding
7. The reuse and adaptation of portion of a previously trained neural network to a fresh but comparable assignment is defined as transfer learning.
8. Natural Language Processing

# Day 5: Data Pre-processing and Creation of Training Dataset

Data Preprocessing is a “data mining technique, which is used to transform raw data into a comprehensible and effective format”. Real-world data tends to lack certain behaviors or trends and is almost always incomplete, inconsistent, and/or missing attribute values, flooded with errors or outliers. Preprocessing data is a proven way to solve such problems. This raw data or real data from the world cannot be readily transmitted through a machine learning model. Therefore, before feeding real world data to a machine learning model, we need to clean and pre-process it.

## Overview Data Preprocessing

### Data Cleaning:

Many meaningless and missing sections can be found in the data. “Data cleaning” is performed in order to manage these inadequacies and constitutes handling data set that is missing values and consists of noisy data .

### Missing Data

In this scenario, certain significant information in the data set is missing. It can be dealt with in different respects, such as:

- 

**Ignoring the tuples:** This strategy is appropriate only if the dataset is big, and numerous values within a tuple are lacking.

- 

**Fill the missing values:** This assignment can be done in different ways, such as: manually completing the missing values by utilizing the mean attribute or the most relevant value.

## Noisy Data

“Noisy data” is useless data that machines or the machine learning model is unable to interpret. It can be generated as a result of defective data collection or mistakes in data entry, among others. It can be addressed by utilizing the methods below:

- 

**Binning Method:** This technique operates on sorted data to smoothen it out. The entire data set is split into equivalent size sections, and then different techniques are used to finish the job. Each section is fixed individually. To fix the entire data set in ago, all data points in a section can be substituted with its “mean” or most probable values.

- 

**Regression:** In this case, data may be smoothed by fitting into a “regression function”, which can be either “linear” (with one autonomous variable) or “multiple” (with various autonomous variables).

- 

**Clustering:** This technique is used to group comparable data points into a cluster. The outliers could be obtained with data points falling outside of the clusters or could not be detected.

## Data Transformation

This technique is used to convert the data into format, which is suitable for the data mining method. This includes the following ways:

- 

**Normalization:** This technique is used to scale data values within a defined range, for example, “-1.0 to 1.0” or “0.0 to 1.0”.

- 

**Attribute Selection:** New data attributes can be generated from the existing data set of characteristics by utilizing this technique to assist in the data mining process.

-

**Discretization:** This technique is used to “replace raw values of numerical attributes with interval or conceptual levels”.

- 

**Generation of the concept of hierarchy:** This technique is used to transform lower level data attributes to higher level in the hierarchical set up. For example, you can convert the attribute "city" to "country".

## Data Reduction

“Data mining” is a method used for analysis and extraction of insights from Big data. In such instances, analysis becomes more and more difficult to work with, given the enormity of data. We use data reduction method to decrease the volume of data set to an optimal and manageable volume. With this method the cost of data storage and analysis can be significantly lowered while improving the effectiveness of the data storage. It can be dealt with in different respects such as:

- 

**Data Cube Aggregation:** This technique is used to apply “aggregation operation” to data to effectively build data cubes.

- 

**Selection of attribute subset:** This technique is used to ensure that only necessary data attributes are used, and the not so relevant attributes can be discarded. To perform attribute selection, the “level of significance” and “p-value of the attribute” can be leveraged. The attribute with “p-value” higher than the “significance level” can be removed to obtain optimal volume of the data set.

- 

**Numerosity Reduction:** This technique allows the data model to be stored instead of the whole data set or raw data collected from various input sources, for instance, “Regression Models”.

**Dimensionality Reduction:** This technique uses encoding mechanisms to reduce the volume of the data set. If initial data set can be recovered after reconstruction from compressed data set, this reduction in dimensions of the dataset is called as “lossless reduction”, otherwise, it is referred to as “loss reduction”.

The two efficient techniques of reducing data set “dimensionality” are: “Wavelet transforms” and “PCA (Principal Component Analysis)”.

## Steps of Data Pre-processing

### I – Import the data library

There is a wide variety of data libraries available that you can choose to meet your data requirements, such as:

**“Pandas”:** Widely used for data visualization and data manipulation processes.

**“NumPy”:** A basic package to perform scientific computations with the use of Python programming language.

**“Matplotlib”:** A standard Python Library used by data scientists to create 2-D plots and graphs.

**“Seaborn”:** Seaborn is derived from the “Matplotlib” library and an extremely popular visualization library.

For example, you can use the main libraries from “Pandas”, “NumPy” and “time”; data visualization libraries from “Matplotlib” and “Seaborn”; and Scikit-Learn libraries for the data preprocessing techniques and algorithms. To import the libraries mentioned above, use the code below:

**For main libraries**    *“import pandas as pd import numpy as np import time”*

**For visualization libraries**    *“from matplotlib import pyplot as plt import seaborn as sns from mpl\_toolkits.mplot3d import Axes3D plt.style.use('ggplot')”*

**For Scikit-Learn libraries**    *“from sklearn.neighbors import KNeighborsClassifier from sklearn.model\_selection import train\_test\_split from sklearn.preprocessing import normalize from sklearn.metrics import*

```
confusion_matrix,accuracy_score,precision_score,recall_score,f1_score,ma  
tthews_corrcoef,classification_report,roc_curve from sklearn.externals  
import joblib from sklearn.preprocessing import StandardScaler from  
sklearn.decomposition import PCA"
```

## **II – Data exploration**

To get some sense of the imported dataset in Pandas, use the code below:

```
"# Read the data in the CSV file using pandas df =  
pd.read_csv('../input/creditcard.csv') df.head()"
```

## **III – Check for missing values**

It is essential to comprehend the concept of missing values to be able to effectively manage data. If the researcher does not handle the missing values correctly, they may end up drawing incorrect data inferences. Because of improper handling, the results produced will be different from those with missing values. You can apply any of the techniques below to deal with missing data values in your data set:

**1.**

### **Ignoring the data row**

It's generally performed when the "class label" is missing or if multiple data attribute are missing in the row, assuming the data mining objective is classification. However, if the proportion of such rows with missing class labels is high, you will obviously get bad output.

For instance, database with enrolment data for the student (age, SAT score, address, etc.) containing a column with "Low", "Medium" and "High" to classify their achievement in college. Assuming the objective is to construct a model that predicts the college achievement of a student. Data rows that do not include the achievement column are not helpful to generate predictions regarding the success of the student, so they can be overlooked and deleted before the algorithm is executed.

**2.**

## **Using a global constant for filling in the missing values**

In this technique, an appropriate and new global constant value is selected, such as "unknown", "N / A" or "minus infinity", which is then utilized for filling all the missing values. This method is employed when the concerted effort to predict the missing value just doesn't make sense. Let's consider the student enrollment database example again, assume that some students lack information on the 'state of residence' attribute. It doesn't really make sense to fill it with some random state instead of using "N/A".

### **3.**

#### **Using attribute mean**

This technique is used to replace an attribute's missing values with the "mean or median value (if its discrete)" for specific attribute in the database. For instance, in a US family expenditure database, if the average expenses of a family are A then we may be able to utilize A to replace missing values in the other family records.

### **4.**

#### **Using attribute mean for all samples belonging to the same class**

This technique is used to restrict the calculations to a particular class for obtaining a value that is applicable to the row that we are searching for in lieu of utilizing the "mean (or medians)" of a particular attribute calculated by searching all the rows of the database. For example, if you have an automobile price database that classifies vehicles, among other things, into "Luxury" and "Low Budget". It is likely more precise to replace the missing price of a "luxury car" with the average price of all luxury vehicles instead of the value obtained after factoring in "low budget cars".

### **5.**

## **Using data mining algorithm for prediction of the most probable value**

Data mining algorithms such as “Regression”, inference based tools utilizing the Bayesian formula, “decision trees”, clustering algorithms (K-Mean or Median) among others, can be used to determine the probable value of the data attribute. For instance, “clustering algorithms” could be employed to produce a cluster of rows that are then used to calculate the mean or median of the attribute, as indicated earlier in Technique 3. Another instance might be to use a decision tree to generate a prediction for the most probable value of the missing attributes by taking into account all other attributes in the data set.

### **IV – Dealing with Categorical Data values**

Categorical attributes can only take on a restricted amount of feasible values, which are generally fixed. For instance, if a dataset is about user-related data, then characteristics such as ‘nation,’ ‘gender,’ ‘age group,’ etc. will constitute the data set. Alternatively, you will find attributes such as ‘product type,’ ‘manufacturer,’ ‘vendor,’ and so on if your data set pertains to a commodity or product.

In context of the data set, these are all categorical attributes. Typically, these attributes are stored as text values representing different characteristics of the observations. Gender is defined, for instance, as “male” or “female”, and product type could be defined as “electronics”, “apparel”, “food”, and so on.

There are three types of categorical data:

- 

**Nominal** – The types of attributes where categories are only labeled and have no order of succession are called “nominal features”. For example, gender could be ‘male’ (M) or ‘female’ (F) and have no order of precedence.

- 

**Ordinal** – The types of attributes where categories are labeled with an order of precedence are called “ordinal features”. For example, an economic status feature can contain

three categories: “low”, “medium”, and “high”, which have an inherent order associated with them.

- 

**Continuous** – The types of attributes where categories are numerical variables with infinite values ranging between two defined values are called “continuous features”.

## Challenges of Categorical Data:

- 

Categorical attributes can contain multiple levels, called as “high cardinality” (e.g. states, towns or URLs), where most levels appear in relatively smaller number of instances.

- 

Various ML models are algebraic, such as “regression” and “SVM”, which require numerical input. Categories have to be changed first to numbers in order to use these models before the machine learning algorithm can be applied.

- 

While some machine learning packages or libraries are capable of automatically transforming the categorical data into numeric, depending on the default embedded technique, a variety of machine learning libraries don't support categorical data inputs.

- 

Categorical data for the computer does not translate the context or background, that people can readily associate with and comprehend. For instance, consider a function called “City” with different city names like “New York”, “New Jersey”, and “New Delhi”. People know that “New York” is strongly linked to “New Jersey” being two neighboring states of America, while “New York” and “New Delhi” are very distinct. On the other hand, for the machine, all three cities just denote three distinct levels of the same “City” function. Without specifying adequate context through data for the model, differentiating between extremely distinct levels will be difficult for it.

## **Encoding Categorical Data:**

ML models are built on math equations, so it is easy to comprehend that maintaining the categorical data in equations would cause issues since equations are primarily driven by numbers alone. To cross this hurdle, the categorical features can be encoded to numeric quantities. The encoding techniques below will be described with the use of the example of an “airline carrier” column from a make believe airline database, for ease of understanding. However, it is possible to extend the same techniques to any desired column.

### **1.**

#### **Replacing the categorical values**

This is a fundamental technique of replacing the categorical data values with required integers. The “*replace()*” function in Pandas, can be used for this technique. Depending on your business requirements, desired numbers can be easily assigned to the categorical values.

### **2.**

#### **Encoding Labels**

The technique of converting categorical values in a column to a number is called as "label encoding". Numerical labels always range from “0” to “n categories-1”. Encoding a group of category to a certain numerical value and then encoding all other categories to another numerical value can be done by utilizing the “*where()*” function in NumPy. For example, one could encode all the “US airline carriers” to value “1” and all other carriers can be given a value “0”. You can perform similar label encoding by utilizing the Scikit-Learn Label Encoder.

Label encoding is fairly intuitive and simple and produce satisfactory performance from your learning algorithm. However, the algorithm is at a disadvantage and may misinterpret numerical values. For example, algorithm may confuse whether the “U.S. airline carrier” (encoded to 6) should be given 6 times more weight “U.S. airline carrier” (encoded to 1).

### **3.**

#### **One-Hot encoding**

To resolve the misinterpretation issue of the machine learning algorithm generated by the “label encoding” technique, each categorical data value can be transformed into a new column, and that new column can be allocated a ‘1’ or ‘0’ (True/False) value, and is called as “one-hot encoding”.

Of all the machine learning libraries in the market that offer “one-hot encoding”, the easiest one is “*get\_dummies()*” technique in “Pandas”, which is appropriately titled given the fact that dummy/indicator data variables such as “1” or “0” are created. In its preprocessing module, Scikit-Learn also supports “one-hot encoding” in it’s pre-processing module via “LabelBinarizer” and “OneHotEncoder” techniques.

While “one-hot encoding” addresses the issue of misinterpreted category weights, it gives rise to another issue. Creation of multiple new columns to solve this category weight problem for numerous categories can lead to a “curse of dimensionality”. The logic behind “curse of dimensionality” is that some equations simply stop functioning correctly in high-dimensional spaces.

### **4.**

#### **Binary encoding**

This method initially encodes the categories as “ordinal”, then converts these integers into a binary string, and then divides digits of that binary code into distinct columns. Therefore, the data is encoded in only a few dimensions unlike the “one-hot encoding” method.

There are several options to implement binary encoding in your machine learning model, but the easiest option is to install “category\_encoders” library. This can be done with the use of the “`pip install category_encoders`” file on cmd.

### **5.**

## **Backward difference encoding**

This “backward difference encoding” method falls within the “contrast coding scheme” for categorical attributes. A “K” category or level characteristic typically enters a “regression” as a series of dummy “K-1” variables. This technique works by drawing a comparison between the “mean” of the dependent variable for a level with the “mean” of the dependent variable in the preceding stage. This kind of encoding is widely used for a “nominal” or an “ordinal “variable”.

The code structure for this technique is quite similar to any other technique in the “category\_encoders” library, except the run command for this technique is “BackwardDifferenceEncoder”.

## **6.**

### **Miscellaneous features**

You may sometimes deal with categorical columns that indicate the range of values in observation points; for instance, an ‘age’ column can contain categories such as ‘0-20’, ‘20-40’, ’40-60’ etc. While there may be many methods to handle such attributes, the most popular ones are:

#### **A.**

Dividing the categorical value ranges into two distinct columns, by first creating a dummy data frame with just one feature as “age” and then splitting the column on the delimiter “(-)” into two columns “start” and “end” using “*split()*” and “*lambda()*” functions .

#### **B.**

Replacing the categorical value ranges with select measure like the mean value of the range, by utilizing the “*split\_mean()*” function.

## **V – Splitting the data set into Training and Testing data subsets**

The ML algorithms are required to learn from sample data set to be able to generate predictions from the input data set. In general, we divide the data set into a proportion of 65:35 or 80:20, which means that 65% of the

data is utilized as the training subset, and 35% of the data is utilized as the testing subset. However, this split ratio is adjusted according to the form and size of the data set .

It is almost impossible and futile to manually split the data set while making sure the data set is divided randomly. The Scikit-Learn library offers us a tool called the “Model Selection library” to assist with this task. There is a class in the Scikit-Learn library called “*train\_test\_split*”. With the use of this class, we can readily divide the data set into the “training” and “testing” datasets in desired ratios. Certain important parameters that should be considered with the use of this tool are:

- 

**Test\_size** - It helps in determining the size of the data to be divided as the testing data set, as a fraction of the total data set. For example, entering 0.3 as the “test\_size” value, the data set will be divided at 30 percent as the test data set. If you specify this parameter, the next parameter may be ignored.

- 

**Train\_size** – This parameter is only specified if the “test\_size” has not been specified already. The process works similar to the “test\_size” function, except that the percentage of the data set specified is for the “training set”.

- 

**Random\_state** - An integer is entered here for the Scikit-Learn class, on the basis of which the “random number generator” will be activated during the data set split. Alternatively, an instance of “RandomState” class can be entered that will then generate random numbers. If you don't enter either of the functions, the default will be activated, which leverages the “RandomState” instance used by “np.random”.

For example, the data set in the picture below can be split into two subsets: ‘X’ subset for the “independent features” and ‘Y’ subset for the “dependent variables” and also happens to be the last column of the data set.

Country	Age	Salary	Purchased
France	44	72000	No
Spain	27	48000	Yes
Germany	30	54000	No
Spain	38	61000	No
Germany	40	nan	Yes
France	35	58000	Yes
Spain	nan	52000	No
France	48	79000	Yes
Germany	50	83000	No
France	37	67000	Yes

Now we can use the code below for splitting the “x” data set into two subsets: “xTrain” and “xTest” and likewise, divide the “y” data set into two subsets “yTrain” and “yTest”.

```
“from sklearn.model_selection import train_test_split xTrain, xTest,
yTrain, yTest = train_test_split(x, y, test_size = 0.2, random_state = 0)”
```

## Day 6: Machine Learning Libraries

Machine learning libraries are sensitive routines and functions that are written in any given language. Software developers require a robust set of libraries to perform complex tasks without needing to rewrite multiple lines of code. Machine learning is largely based on mathematical optimization, probability, and statistics.

In the field of machine learning, Python is highly preferred credited to its consistent development time and flexibility. It is well suited to develop sophisticated models and production engines that can be directly plugged into production systems. One of its greatest assets being an extensive set of libraries that can help researchers who are less equipped with developer knowledge to easily execute machine learning.

The Scikit-Learn library has evolved as the gold standard for the development of machine learning models with the use of Python, offering a wide variety of supervised and unsupervised ML algorithms. It is touted as one of the most user friendly and cleanest ML libraries to date. For example, decision trees, clustering, linear and logistics regressions and K-means. Scikit-learn uses a couple of basic Python libraries: NumPy and SciPy and adds a set of algorithms for data mining tasks including classification, regression and clustering. It is also capable of implementing tasks like feature selection, transforming data and ensemble methods in only a few lines.

In 2007, David Cournapeau developed the foundational code of Scikit-Learn during his participation in the “Summer of code” project for Google. Scikit-learn has become one of Python's most famous open source machine learning libraries since its launch in 2007. But it wasn't until 2010 that Scikit-Learn was released for public use. Scikit-Learn is an open sourced and BSD licensed, data mining and data analysis tool used to develop supervise and unsupervised machine learning algorithms build on Python. Scikit-learn offers various ML algorithms such as “classification”, “regression”, “dimensionality reduction”, and “clustering”. It also offers modules for feature extraction, data processing, and model evaluation.

Designed as an extension to the “SciPy” library, Scikit-Learn is based on “NumPy” and “Matplotlib”, the most popular Python libraries. NumPy expands Python to support efficient operations on big arrays and multidimensional matrices. Matplotlib offers visualization tools and science computing modules are provided by SciPy.

For scholarly studies, Scikit-Learn is popular because it has a well-documented, easy-to-use and flexible API. Developers are able to utilize Scikit-Learn for their experiments with various algorithms by only altering a few lines of the code. Scikit-Learn also provides a variety of training datasets, enabling developers to focus on algorithms instead of data collection and cleaning.

Many of the algorithms of Scikit-Learn are quick and scalable to all but huge datasets. Scikit-learn is known for its reliability and automated tests are available for much of the library. Scikit-learn is extremely popular with beginners in machine learning to start implementing simple algorithms .

## **Prerequisites for application of Scikit-Learn library**

The Scikit-Learn library is based on the SciPy (Scientific Python), which needs to be installed before you can use SciKit-Learn on your system. This stack involves the following:

- NumPy (Base n-dimensional array package)
- Matplotlib (Comprehensive 2D/3D plotting)
- SciPy (Fundamental library for scientific computing)
- IPython (Enhanced interactive console)
- SymPy (Symbolic mathematics)
- Pandas (Data structures and analysis)
-

## Seaborn (data visualization)

### Installing Scikit-Learn

The latest version of Scikit-Learn can be found on “Scikit-Learn.org” and requires “Python (version  $\geq$  3.5); NumPy (version  $\geq$  1.11.0); SciPy (version  $\geq$  0.17.0); joblib (version  $\geq$  0.11)”. The plotting capabilities or functions of Scikit-learn start with “plot\_” and require “Matplotlib (version  $\geq$  1.5.1)”. Certain Scikit-Learn examples may need additional applications: “Scikit-Image (version  $\geq$  0.12.3), Pandas (version  $\geq$  0.18.0)” .

With prior installation of “NumPy” and “SciPy”, the best method of installing Scikit-Learn is by utilizing “pip: pip install -U scikit-learn” or “conda: conda install scikit-learn”.

One must make sure that binary wheels are utilized with the use of the pip files and that “NumPy” and “SciPy” have not been recompiled from source, which may occur with the use of specific OS and hardware settings (for example, “Linux on a Raspberry Pi”). Developing “NumPy” and “SciPy” from source tends to be complicated (particularly on Windows). Therefore, they need to be setup carefully, making sure the optimized execution of linear algebra routines is achievable.

### Application of machine learning using Scikit-Learn library

To understand how Scikit-Learn library is used in the development of machine learning algorithm, let us use the “Sales\_Win\_Loss data set from IBM’s Watson repository” containing data obtained from sales campaign of a wholesale supplier of automotive parts. We will create a ML model to predict which sales campaign will be a winner and which will incur loss.

The data set can be imported with the Pandas library and explored using Pandas techniques such as “head (), tail (), and dtypes ()”. The plotting techniques from “Seaborn” will be used to visualize the data. To process the data Scikit-Learn’s “preprocessing.LabelEncoder ()” will be used and “train\_test\_split ()” to separate the data set into training subset and testing subset.

To generate predictions from our data set, three different algorithms will be used namely, “Linear Support Vector Classification and K-nearest neighbors classifier”. To compare the performances of these algorithms Scikit-Learn library technique, “accuracy\_score”, will be used. The performance score of the models can be visualized with the use of the Scikit-Learn and Yellowbrick visualization tool.

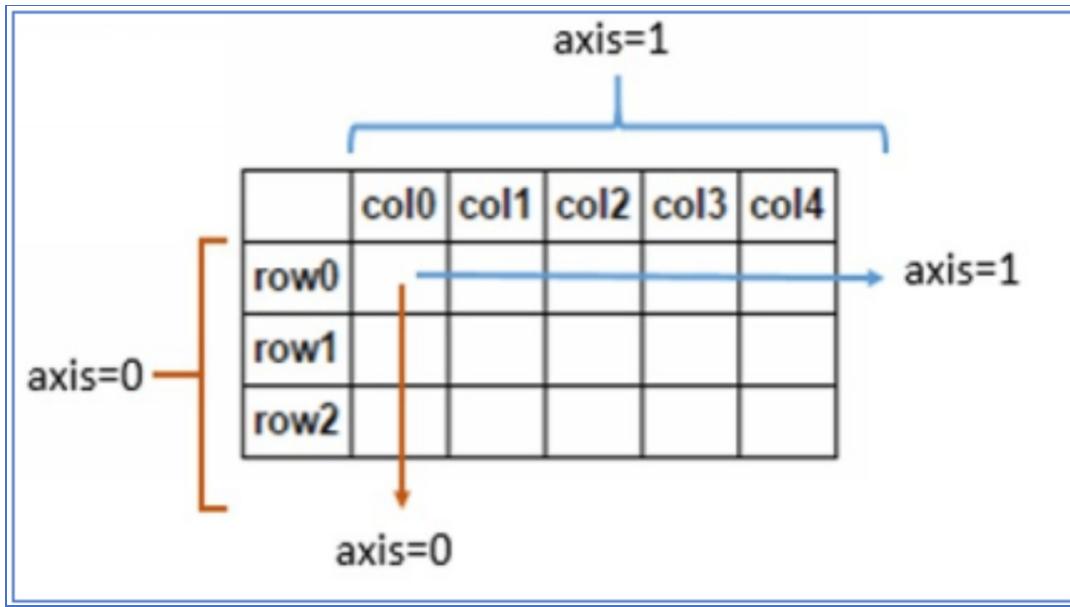
### **Importing the data set**

To import the “Sales\_Win\_Loss data set from IBM’s Watson repository”, first step is importing the “Pandas” module by executing the “*import pandas as pd*” command.

Then we leverage a variable url as “[https://community.watsonanalytics.com/wp-content/uploads/2015/04/WA\\_Fn-UseC\\_-Sales-Win-Loss.csv](https://community.watsonanalytics.com/wp-content/uploads/2015/04/WA_Fn-UseC_-Sales-Win-Loss.csv)” for storage of the URL from where you will be downloading the dataset.

Now, “*read\_csv() as sales\_data = pd.read\_csv(url)*” technique will be used to read the above “csv or comma separated values” file, which is supplied by the Pandas module. The csv file will then be converted into a Pandas data framework, with the return variable as “*sales\_data*”, where the framework will be stored.

For new ‘Pandas’ users, the “*pd.read\_csv()*” technique in the code mentioned above will generate a tabular data structure called “data framework”, where an index for each row is contained in the first column, and the label / name for each column in the first row are the initial column names acquired from the data set. In the above code snippet, the “*sales data*” variable results in a table depicted in the picture below.



In the diagram above, the “row0, row1, row2” represent the individual record index and the “col0, col1, col2” represent the names for individual columns or features of the data set.

With this step, you have successfully stored a copy of the data set and transformed it into a “Pandas” framework!

Now, using the “*head()* as *Sales\_data.head()*” technique, the records from the data framework can be displayed as shown below to get a “feel” of the information contained in the data set.

	opportunity number	supplies subgroup	supplies group	region	route to market	elapsed days in sales stage	opportunity result
0	1641984	Exterior Accessories	Car Accessories	Northwest	Fields Sales	76	Won
1	1658010	Exterior Accessories	Car Accessories	Pacific	Reseller	63	Loss
2	1674737	Motorcycle Parts	Performance & Non-auto	Pacific	Reseller	24	Won
3	1675224	Shelters & RV	Performance & Non-auto	Midwest	Reseller	16	Loss

## Data Exploration

Now that we have our own copy of the data set, which has been transformed into a “Pandas” data frame, we can quickly explore the data to understand what information can tell can be gathered from it and accordingly to plan a course of action.

In any ML project, data exploration tends to be a very critical phase. Even a fast data set exploration can offer us significant information that could be easily missed otherwise, and this information can propose significant questions that we can then attempt to answer on the basis of our project .

Some third-party Python libraries will be used here to assist us with the processing of the data so that we can efficiently use this data with the powerful algorithms of Scikit-Learn. The same “*head()*” technique that we used to see some initial records of the imported data set in the earlier section can be used here. As a matter of fact, “*(head)* ” is effectively capable of doing much more than displaying data records and customize the “*head()*” technique to display only selected records with commands like “*sales\_data.head(n=2)*”. This command will selectively display the first 2 records of the data set. At a quick glance, it’s obvious that columns such as “Supplies Group” and “Region” contain string data, while columns such as

“Opportunity Result”, “Opportunity Number”, etc. are comprised of integer values. It can also be seen that there are unique identifiers for each record in the ‘Opportunity Number’ column. Similarly, to display select records from the bottom of the table, the “`tail() as sales_data.tail()`” can be used.

To view the different data types available in the data set, the Pandas technique “`dtypes() as sales_data.dtypes`” can be used. With this information, the data columns available in the data framework can be listed with their respective data types. We can figure out, for example, that the column “Supplies Subgroup” is an “object” data type and that the column “Client Size By Revenue” is an “integer data type”. So, we have an understanding of columns that either contains integer values or string data.

## Data Visualization

At this point, we are through with basic data exploration steps, so we will not attempt to build some appealing plots to portray the information visually and discover other concealed narratives from our data set. Of all the available python libraries providing data visualization features, “Seaborn” is one of the best available options so we will be using the same. Make sure that python plots module provided by “Seaborn” has been installed on your system and ready to be used. Now follow the steps below, generate the desired plot for the data set:

**Step 1** - Import the “Seaborn” module with the command “`import seaborn as sns`”.

**Step 2** - Import the “Matplotlib” module with command “`import matplotlib.pyplot as plt`” .

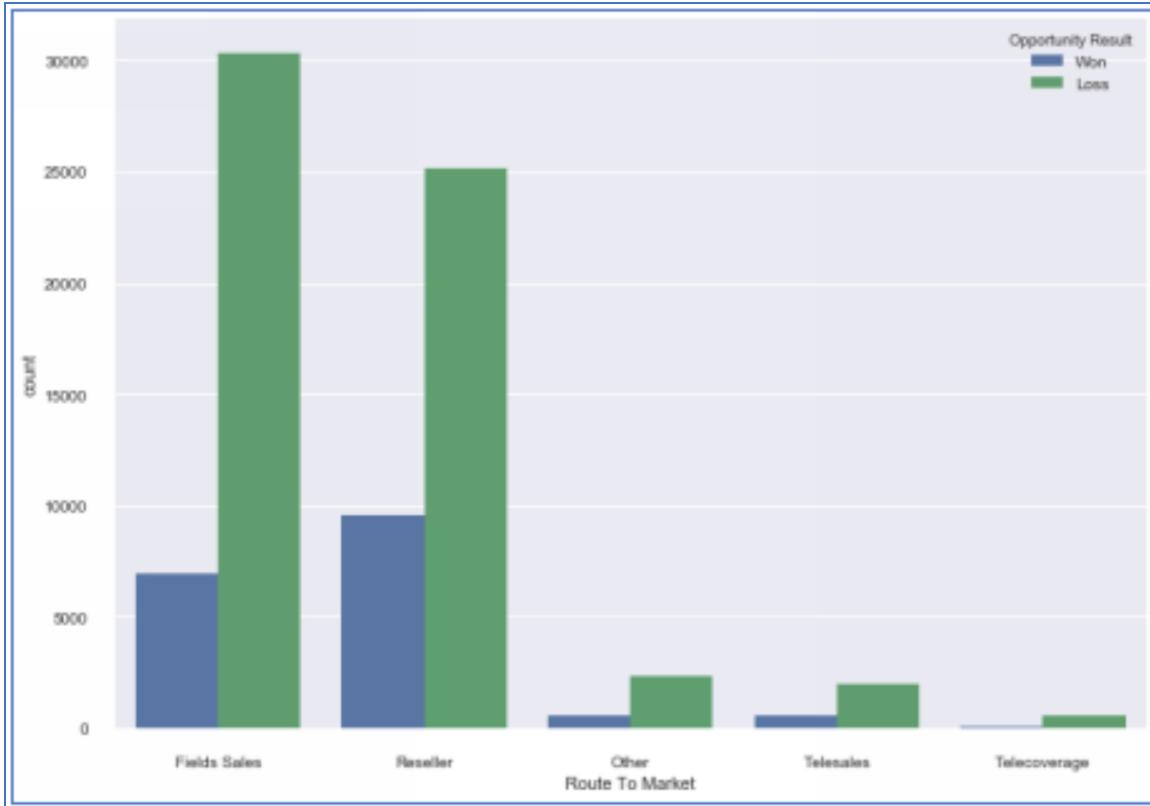
**Step 3** - To set the “background color” of the plot as white, use command “`sns.set(style='whitegrid', color_codes=True)`”.

**Step 4** - To set the “plot size” for all plots, use command “`sns.set(rc={'figure.figsize':(11.7,8.27)})`”.

**Step 5** – To generate a “countplot”, use command “`sns.countplot('Route To Market', data=sales_data, hue = 'Opportunity Result')`”.

**Step 6** – To remove the top and bottom margins, use command “`sns.despine(offset=10, trim=True)`”.

**Step 7** – To display the plot, use command “`plotplt.show()`”.



Quick recap - The “Seaborn” and “Matplotlib” modules were imported first. Then the “`set()`” technique was used to define the distinct characteristics for our plot, such as plot style and color. The background of the plot was defined to be white using the code snippet “`sns.set(style= "whitegrid", color_codes= True)`”. Then the plot size was defined using command “`sns.set(rc= {'figure.figsize': (11.7,8.27)})`” that define the size of the plot as “11.7px and 8.27px”.

Next the command “`sns.countplot('Route To Market',data= sales_data, hue='Opportunity Result')`” was used to generate the plot. The “`countplot()`” technique enables creation of a count plot, which can expose multiple arguments to customize the count plot according to our requirements. As part of the first “`countplot()`” argument, the X-axis was defined as the column “Route To Market” from the data set. The next argument concerns the source of the data set, which would be “sales\_data” data framework we imported earlier. The third argument is the color of the bar graphs that were defined as “blue” for the column labeled “won” and “green” for the column labeled “loss” .

## Data Pre-processing

By now, you should have a clear understanding of what information is available in the data set. From the data exploration step, we established that majority of the columns in our data set are “string data”, but “Scikit-Learn” can only process numerical data. Fortunately, the Scikit-Learn library offers us many ways to convert string data into numerical data, for example, “*LabelEncoder()*” technique. To transform categorical labels from the data set such as “won” and “loss” into numerical values, we will use the “*LabelEncoder()*” technique.

Let's look at the pictures below to see what we are attempting to accomplish with the “*LabelEncoder()*” technique. The first image contains one column labeled “color” with three records, namely, “Red”, “Green” and “Blue”. By utilizing the “*LabelEncoder()*” technique, the record in the same “color” column can be converted to numerical values, as shown in the second image.

	Color		Color
0	1	0	Red
1	2	1	Green
2	3	2	Blue

Let's begin the real process of conversion now. Using the “*fit transform()*” technique given by “*LabelEncoder()*”, the labels in the categorical column like “Route To Market” can be encoded and converted to numerical labels comparable to those shown in the diagrams above. The function “*fit transform()*” requires input labels identified by the user and consequently returns encoded labels.

To know how the encoding is accomplished, let's go through an example quickly. The code instance below constitutes string data in form of a list of cities such as ["Paris", "Paris", "Tokyo", "Amsterdam"] that will be encoded into something comparable to “[ 2, 2, 1,3]”.

**Step 1** - To import the required module, use command “*from sklearn import preprocessing*”.

**Step 2** – To create the Label encoder object, use command “*le = preprocessing.LabelEncoder()*”.

**Step 3** – To convert the categorical columns into numerical values, use command:

```
“encoded_value = le.fit_transform(["Paris", "Paris", "Tokyo",  
"Amsterdam"])”  
“print(encoded_value) [1 1 2 0] ”
```

And there you have it! We just converted our string data labels into numerical values. The first step was importing the preprocessing module that offers the “*LabelEncoder()*” technique. Followed by the development of an object representing the “*LabelEncoder()*” type. Then the “*fit\_transform()*” function of the object was used to distinguish between distinct classes of the list [ “Paris”, “Paris”, “Tokyo”, “Amsterdam”] and output the encoded values of “[ 1 1 20]”.

Did you observe that the “*LabelEncoder()*” technique assigned the numerical values to the classes in alphabetical order according to the initial letter of the classes, for example “(A)msterdam” was assigned code “0”, “(P)aris” was assigned code “1” and “(T)okyo” was assigned code “2”.

## Creating Training and Test subsets

To know the interactions between distinct characteristics and how these characteristics influence the target variable, a ML algorithm must be trained on a collection of information. We need to split the complete data set into two subsets to accomplish this. One subset will serve as the training data set, which will be used to train our algorithm to construct machine learning models. The other subset will serve as the testing data set, which will be used to test the accuracy of the predictions generate by the ML model.

The 1<sup>st</sup> phase in this stage is separation of feature and target variables by following the steps below:

**Step 1** – To select data excluding select columns, use command “*select columns other than 'Opportunity Number', 'Opportunity Result'cols = [col for col in sales\_data.columns if col not in ['Opportunity Number','Opportunity Result']]*”.

**Step 2** – To drop these select columns, use command “*dropping the 'Opportunity Number' and 'Opportunity Result' columns  
data = sales\_data[cols]*”.

**Step 3** – To assign the Opportunity Result column as “target”, use command “`target = sales_data['Opportunity Result']  
data.head(n=2)`”.

The “Opportunity Number” column was removed since it just acts as a unique identifier for each record. The “Opportunity Result” contains the predictions we want to generate, so it becomes our “target” variable and can be removed from the data set for this phase. The first line of the above code will select all the columns except “Opportunity Number” and “Opportunity Result” in and assign these columns to a variable “cols”. Then using the columns in the “cols” variable, a new data framework was developed. This is going to be the “feature set”. Next, the column “Opportunity Result” from the “`sales_data`” data frame was used to develop a new data framework called “target”.

The second phase in this stage is to separate the date frameworks into trainings and testing subsets by following the steps below. Depending on the data set and desired predictions, it needs to be split into training and testing subset accordingly. For this exercise, we will use 75% of the data as training subset, and rest 25% will be used for the testing subset. We will leverage the “`train_test_split()`” technique in “Scikit-Learn” to separate the data using steps and code as below:

**Step 1** – To import required module, use command “`from sklearn.model_selection import train_test_split`”.

**Step 2** – To separate the data set, use command “`split data set into train and test sets` `data_train, data_test, target_train, target_test = train_test_split(data, target, test_size = 0.30, random_state = 10)`”.

With the code above, the “`train_test_split`” module was first imported, followed by the use of “`train_test_split()`” technique to generate “training subset (`data_train, target_train`)” and “testing subset (`data_test, data_train`)”. The “`train_test_split()`” technique’s first argument pertains to the features that were divided in the preceding stage; the next argument relates to the target (“Opportunity Result”). The third “test size” argument is the proportion of the data we wish to divide and use as a testing subset. For this example, we will busing 30% of the data for the testing subset,

although this percentage share can vary as needed. The fourth 'random state' argument is used to make sure that the results can be reproduced every time.

## Building the Machine Learning Model

The "machine\_learning\_map" provided by Scikit-Learn is widely used to choose the most appropriate ML algorithm for the data set. For this exercise, we will be using "Linear Support Vector Classification" and "K-nearest neighbors classifier" algorithms .

### Linear Support Vector Classification

"Linear Support Vector Classification" or "Linear SVC" is a sub-classification of "Support Vector Machine (SVM)" algorithm, which we have reviewed in chapter 2 of this book titled "Machine Learning Algorithms". Using Linear SVC, the data can be divided into different planes so the algorithm can identify the optimal group structure for all the data classes.

Here are the steps and code for this algorithm to build our first ML model:

**Step 1** – To import the required modules, use commands "*from sklearn.svm import LinearSVC*" and "*from sklearn.metrics import accuracy\_score*".

**Step 2** – To develop an LinearSVC object type, use command "*svc\_model = LinearSVC(random\_state=0)*" .

**Step 3** – For training the algorithm and generate predictions from the testing data, use command "*pred = svc\_model.fit(data\_train, target\_train).predict(data\_test)*" .

**Step 4** – To display the model accuracy score, use command "*print ('LinearSVC accuracy:', accuracy\_score(target\_test, pred, normalize = True))*" .

With the code above, the required modules were imported in the first step. We then developed a type of Linear SVC with the use of the "svc\_model" object with "random\_state" as '0'. The "random\_state" command instructs the built-in random number generator for shuffling the data in a particular manner. In step 3, the "Linear SVC" algorithm is trained on the training data set and subsequently used to generate predictions for

the target from the testing data. The “*accuracy\_score()*” technique was used in the end to verify the “accuracy score” of the model, which could be displayed as “LinearSVC accuracy : 0.777811004785”, for instance.

### K-nearest Neighbors Classifier

The “k-nearest neighbors(k-NN)” algorithm is referred to as “a non-parametric method used for classification and regression in pattern recognition”. In cases of classification and regression, “the input consists of the nearest k closest training examples in the feature space”. K-NN is a form of “instance-based learning” or “lazy learning”, in which the function is only locally estimated and all calculations are delayed until classification. The output is driven by the fact, whether the classification or regression method is used for k-NN:

- 

“k-nearest neighbors classification” - The “output” is a member of the class. An “object” is classified by its neighbors’ plurality vote, assigning the object to the most prevalent class among its nearest “k-neighbors”, where “k” denotes a small positive integer. If  $k=1$ , the “object” is simply allocated to the closest neighbor’s class.

- 

“k-nearest neighbors regression” - The output is the object’s property value, which is computed as an average of the k-nearest neighbors values.

A helpful method for both regression as well as classification can be assigning weights to the neighbors’ contributions, to allow closer neighbors to make more contributions in the average, compared to the neighbors located far apart. For instance, a known “weighting scheme” is to assign each neighbor a weight of “ $1/d$ ”, where “d” denotes the distance from the neighbor. The neighbors are selected from a set of objects for which the “class” (for “k-NN classification”) or the feature value of the “object” (for “k-NN regression”) is known.

Here are the steps and code for this algorithm to build our next ML model :

**Step 1** – To import required modules, use command “`from sklearn.neighbors import KNeighborsClassifier`” and “`from sklearn.metrics import accuracy_score`”.

**Step 2** – For creation of object of the classifier, use command “`neigh = KNeighborsClassifier(n_neighbors=3)`”.

**Step 3** – For training of the algorithm, use command “`neigh.fit(data_train, target_train)`”.

**Step 4** – To generate predictions, use command “`pred = neigh.predict(data_test)`”.

**Step 5** – To evaluate accuracy, use command “`print ('KNeighbors accuracy score:', accuracy_score(target_test, pred))`”.

With the code above, the required modules were imported in the first step. We then developed the object “`neigh`” of type “`KNeighborsClassifier`” with the volume of neighbors as “`n_neighbors=3`”. In the next step, the “`fit()`” technique was used for training the algorithm on the dataset. Next, the model was tested on the testing data set using “`predict()`” technique. Finally, the accuracy score was obtained, which could be “`KNeighbors accuracy score : 0.814550580998`”, for instance .

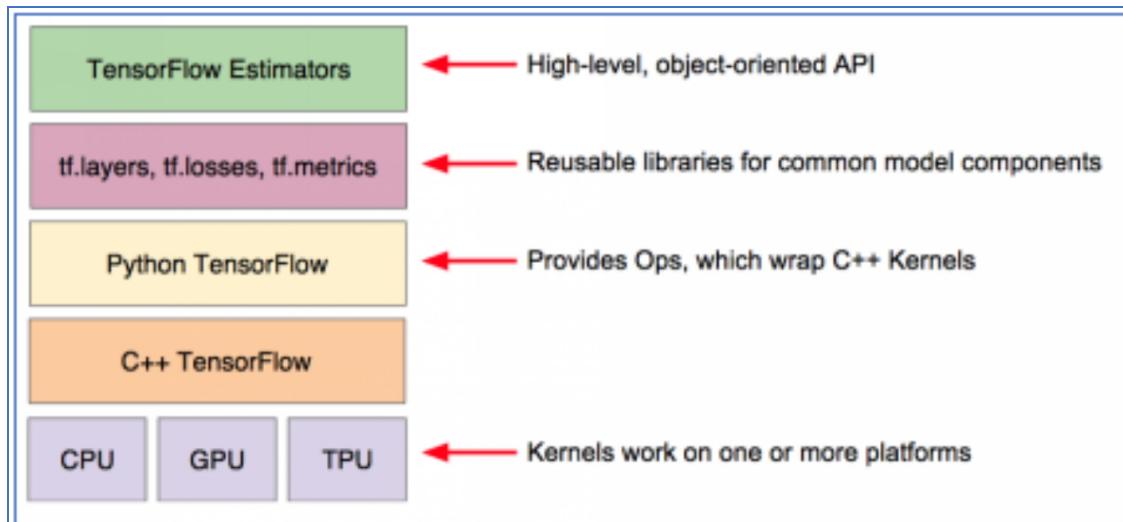
Now that our preferred algorithms have been introduced, the model with the highest accuracy score can be easily selected. In Scikit-Learn, we can utilize the “Yellowbrick library” for drawing comparison of the distinct models' efficiency visually, which offers techniques for depicting various scoring techniques visually.

# Day 7: Neural Network Training With Tensorflow

TensorFlow can be defined as a Machine Learning platform providing end-to-end service with a variety of free and open sources. It has a system of multilayered nodes that allow for quick building, training, and deployment of artificial neural networks with large data sets.

It is touted as a “simple and flexible architecture to take new ideas from concept to code to state-of-the-art models and to publication at a rapid pace”. For example, Google uses TensorFlow libraries in its image recognition and speech recognition tools and technologies.

Higher-level APIs such as “tf.estimator” can be used for specifying predefined architectures, such as “linear regressors” or “neural networks”. The picture below shows the existing hierarchy of the TensorFlow tool kit:



The picture shown below provides the purposes of the different layers:

Toolkit(s)	Description
Estimator (tf.estimator)	High-level, OOP API.
tf.layers/tf.losses/tf.metrics	Libraries for common model components.
TensorFlow	Lower-level APIs

The two fundamental components of TensorFlow are:

1. A “graph protocol buffer”
2. A “runtime” that can execute the graph

The two component mentioned above are similar to “Python” code and the “Python interpreter”. Just as “ Python interpreter” can run Python code on several hardware systems, TensorFlow can be operated on various hardware systems, like CPU, GPU, and TPU.

To make a decision regarding which API(s) should be used, you must consider the API offering the highest abstraction level to solve the target problem. Easier to use, but (by design) less flexible, are the greater abstract levels. It is recommended to first begin with the highest-level API and make everything work. If for certain unique modelling issues, you need extra flexibility, move one level down. Notice that each level is constructed on the lower level APIs. It should thus be quite simple to decrease the hierarchy.

For the development of majority of Machine Learning models, we will use “tf.estimator” API, which significantly lowers the number of code lines needed for development. Also, “tf.estimator” is compatible with Scikit-Learn API.

The programming of computers needs a human programmer. Many lines of code are used by humans to instruct a computer to provide solutions to our problems. However, the computer can attempt to fix the issue itself

through machine learning and neural networks. A neural network is “a function that learns the expected output for a given input from training datasets”. For instance, you can train the neural network with many samples bear pictures to construct a neural network that recognizes pictures of a bear. The resulting network operates as functionality to generate the “bear” label as output for the bear picture input. Another more convenient example would be: training the neural network using multiple user activity logs from gaming servers and generate an output stating which users are very likely to convert to paying customer.

Unlike the “Artificial Neural Network” (explained in detail in Chapter 2 of this book), the “Neural Network” features only a single neuron, also called as “perception”. It is a straightforward and fundamental mechanism that can be implemented with basic math. The primary distinction between traditional programming and a neural network is that computers running on neural network learn from the provided training data set to determine the parameters (weights and prejudice) on their own without needing any human assistance. Algorithms like “back propagation” and “gradient descent” may be used to train the parameters. It can be stated that the computer tries to increase or decrease every parameter a bit, in the hope that the optimal combination of Parameters can be found to minimize the error compared with training data set.

## Fundamentals of Neural Network

•

Neural networks need clear and informative big data to be trained. You can think of Neural networks as a toddler. They start by observing how their parents are walking. Then they attempt to walk on their own, and the kid learns how to accomplish future tasks with every step. Similarly, the Neural network may fail a few times, but it learns how to generate desired predictions after a few failing attempts.

•

For complicated issues such as image processing, it is advisable to use Neural Networks. Neural networks belong to a group of algorithms called “representation learning

algorithms". These algorithms are capable of simplifying complicated issues by generating simple (or "representative") form, which tends to be more difficult for conventional (non-representation) algorithms.

•

To determine what type of neural network model is suitable for solving the issue at hand, let the data dictate how you fix the issue. For instance, "recurring neural networks" are more appropriate if the issue pertains to sequence generation. While it might be better for you to use "convolutional neural networks" to solve an image-related issue.

•

In order to run a deep neural network model, hardware specifications are vital. Neural networks have been around for a long time now, but they are recently experiencing an upsurge primarily credited to the fact that computer resources today are better and more effective. If you want to address a real-life problem using a neural network, it is wise to purchase high-end hardware.

## **Training a Neural Network using TensorFlow**

In this exercise, we will develop a model of neural networks for classifying clothing images such as sneakers and shirts using TensorFlow library .

### **I – Import the dataset**

For this example, we will be using "Fashion MNIST" data set with 60,000 pictures representing 10 different categories. The low resolution pictures (28 to 28 pixels) indicate individual clothing items. For the classic MNIST dataset, "Fashion MNIST" is utilized as a "drop-in" replacement. The "MNIST" data set includes pictures of handwritten numbers (0, 1, and so on.) in the same format as the clothing items used in this example. To train the network, we will use 60,000 pictures, and 10,000 pictures will be

used to assess the accuracy with which the network has learned how to classify pictures.

The “Fashion MNIST” data set is accessible directly from TensorFlow, using the import command as below:

```
“fashion_mnist = keras.datasets.fashion_mnist (train_images,  
train_labels), (test_images, test_labels) = fashion_mnist.load_data()”
```

After the dataset has been loaded system will return 4 different “NumPy arrays” including:

- - The “*train\_images*” and “*train\_labels*” arrays, which serve as the “training dataset” for the model.
  - The “*test\_images*” and “*test\_labels*” arrays, which serve as the “testing dataset” that the model can be tested against.’

Now we need to create labels for an array of integers (0 to 9), corresponding to each category/class of the clothing picture in the data set, using command below, which will look like the table represented in the picture below. This will be useful in generating predictions using our model.

```
“class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',  
'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']”
```

Label	Class
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

## II – Data Exploration

By exploring the data set using the commands listed below, you will be able to get some preliminary sense of the data set:

To view the total no. of images in the “training data set” and the size of each image – “`train_images.shape`”, which will produce the output displayed as “(60000, 28, 28)”, stating we have 60,000 pictures of 28 to 28 pixel size.

To view the total no. of labels in the “training dataset” – “`len(train_labels)`”, which will produce the output displayed as “60000” stating we have 60,000 labels in the training data set.

To view the data type of each label used in the “training dataset”– “`train_labels`”, which will produce the output displayed as “`array([9, 0, 0, ..., 3, 0, 5], dtype=uint8 )`” stating each label is an integer between 0 and 9.

To view the total no. of images in the “testing dataset” and the size of each image – “`test_images.shape`”, which will produce the output displayed as “(10000, 28, 28)”, stating we have 10,000 pictures of 28 to 28 pixel size in the testing data set .

To view the total number of labels in the “testing dataset” – “`len(test_labels)`”, which will produce the output displayed as “10000”,

stating we have 10,000 labels in the testing data set.

### **III – Data Pre-processing**

To make the data suitable for training the model, it needs to be pre-processed. It is essential to pre-process the data sets to be used for training and testing in the same manner.

For instance, you notice the first picture in the training data set has the pixel values between 0 and 255, by executing the commands below:

```
“plt.figure() ” “plt.imshow(train_images[0]) ” “plt.colorbar() ”  
“plt.grid(False) ” “plt.show()”
```

These pixel values need to be scaled to fall between 0 to 1, prior to being utilized as input for the Neural Network model. Therefore, the values need to be divided by 255, for both the data subsets, using commands below:

```
“train_images = train_images / 255.0 ” “test_images = test_images /  
255.0”
```

The final pre-processing step here would be to make sure that the data is in desired format prior to building the Neural Network by viewing the first 20 pictures from the training dataset and displaying the “class name” under each picture, using commands below:

```
“plt.figure(figsize=(10,10)) ” “for i in range(20) :  
    plt.subplot(5,5,i+1 )    plt.xticks([])    plt.yticks([])    plt.grid(False)  
    plt.imshow(train_images[i], cmap=plt.cm.binary )  
    plt.xlabel(class_names[train_labels[i]]) ” “plt.show()”
```

### **IV – Building the Neural Network Model**

To build up the “Neural Network”, the constituting layers of the model first need to be configured and only then the model can be compiled .

#### **Configuring the layers**

The “layers” are the fundamental construction block of a neural network. These “layers” take out information from the data entered, generating representations that tend to be extremely valuable addressing the problem.

Majority of “deep learning” involves stacking and linking fundamental layers together. The parameters that are learned during practice are

available in most of the layers, like “tf.keras.layers.Dense”. To configure the required layers, use command below:

```
“model = keras.Sequential( [    keras.layers.Flatten(input_shape=(28, 28)),    keras.layers.Dense(128, activation=tf.nn.relu),    keras.layers.Dense(10, activation=tf.nn.softmax ) ] )”
```

The “*tf.keras.layers.Flatten*” is the first layer in this network, which turns the picture format from a 2-dimesnional array of 28x28 size to a 1-dimension array with “28x28 = 784” pixels. Consider this layer as unchained rows of pixels in the picture that arranged these pictures but without any learning parameters and capable of only altering the data .

The network comprises of couple of “*tf.keras.layers.Dense* ” layers after pixels are flattened. These are neural layers that are fully or densely connected. There are 128 nodes or neurons in the first Dense layer. The succeeding and final layer is a ten node layer of “*Softmax* ”, which generated an array of ten different probability scores amounting to “1”. Every single node includes a probability score indicating that one of the ten classes is likely to contain the existing picture.

### **Compiling the model**

Before being able to train the model, some final tweaks are needed to be made in the model compilation step, such as:

- Loss function— This provides a measure of the model's accuracy during training. This feature should be minimized so that the model is "directed" in the correct direction.
- Optimizer —These are the updates made to the model on the basis of the data it can view as well as its “loss function”.
- Metrics — Used for monitoring the training and testing procedures. For example, the code below utilizes accuracy, measured by computing the fraction of the pictures that were classified accurately.

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

## V – Training the Model

The steps listed below are utilized for training the “Neural Network Model”:

- Feed the training data to the model, using “*train\_images*” and “*train\_labels*” arrays.
- Allow the network to learn the association of pictures and corresponding labels.
- Generate predictions using the model for a predefined test date set, for example, the “*test\_images*” array. Then the predictions must be verified by matching the labels from the “*test\_labels*” array.

You can begin to train the network by utilizing the “*model.fit*” method. To verify the system is a "fit" for the training data, use command “*model.fit*(*train\_images*, *train\_labels*, *epochs*=5)”.

The epochs are displayed as below, suggesting that the model has reached accuracy of around 0.89 or 89% of the training data:

*“Epoch 1/5  
60000/60000 [=====] - 4s 75us/sample - loss:  
0.5018 - acc: 0.8241*

*Epoch 2/5  
60000/60000 [=====] - 4s 71us/sample - loss:  
0.3763 - acc: 0.8643*

*Epoch 3/5  
60000/60000 [=====] - 4s 71us/sample - loss:  
0.3382 - acc: 0.8777*

*Epoch 4/5  
60000/60000 [=====] - 4s 72us/sample - loss:  
0.3138 - acc: 0.8846*

*Epoch 5/ 5  
60000/60000 [=====] - 4s 72us/sample - loss:  
0.2967 - acc: 0.8897*

*<tensorflow.python.keras.callbacks.History at 0x7f65fb64b5c0>*

## **VI – Measuring the accuracy of the Neural Network Model**

To test the accuracy of the network, it must be verified against the testing data set by executing the commands below:

```
“test_loss, test_acc = model.evaluate(test_images, test_labels) ”  
“print('Test accuracy:', test_acc)”
```

The output can be obtained as shown below, which suggests that the accuracy of the test result is around 0.86 or 86%, which is slightly less than the accuracy of the training data set. This is a classic example of “overfitting” when the performance or accuracy of the model is lower on new input or testing data than the training data.

*“10000/10000 [=====] - 1s  
51us/sample - loss: 0.3653 - acc: 0.8671  
Test accuracy: 0.8671”*

## **VII – Generate predictions using the Neural Network Model**

Now that our model has been trained sufficiently, we are ready to generate predictions from the model, by utilizing command “*predictions = model.predict (test\_images)*”.

In the code below, the network has generated a prediction for labels of each picture in the testing data set. The prediction is generated as an array of ten integers with the “confidence” index for each of the ten categories (refer the import data stage) corresponding to the test picture.

*“predictions[0]”*

```
“array([6.58371528e-06, 1.36480646e-10, 4.17183337e-08,
1.15178166e-10,
8.30939484e-07, 1.49914682e-01, 3.11488043e-06, 4.63472381e-
02,
6.10820061e-05, 8.03666413e-01], dtype=float32) ”
```

To view the label with the highest “confidence” index, by utilizing command “*np.argmax (predictions[0])*”.

A result “9”, will suggest that the model has maximum confidence on the test image belonging to “*class\_names[9]*” or according to our labels table, ankle boot.

To verify this prediction, use command “*test\_labels[0]*”, which should generate output as “9”.

To view the whole set of predictions for the ten classes, use command below:

```
“def plot_image(i, predictions_array, true_label, img) :
predictions_array, true_label, img = predictions_array[i], true_label[i],
img[i] plt.grid(False )
plt.xticks([] )
plt.yticks([] )

plt.imshow(img, cmap=plt.cm.binary )

predicted_label = np.argmax(predictions_array )
if predicted_label == true_label :
color = 'blue '
else :
color = 'red '

plt.xlabel('{} {:2.0f}% ({})'.format(class_names[predicted_label] ,
100*np.max(predictions_array),
class_names[true_label]) ,
color=color )
```

```

def plot_value_array(i, predictions_array, true_label) :
    predictions_array, true_label = predictions_array[i], true_label[i]
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])
    thisplot = plt.bar(range(10), predictions_array, color="#777777")
    plt.ylim([0, 1])
    predicted_label = np.argmax(predictions_array)

    thisplot[predicted_label].set_color('red')
    thisplot[true_label].set_color('blue')

```

Now, for example, you may want to generate a prediction for a specific picture in the testing data set. You can do this by utilizing the command below:

“Grab an image from the test dataset  
*img = test\_images[0]*  
*print(img.shape)*”  
“(28, 28)”

To use the “tf.keras” models to generate this prediction, the picture must be added to a list, since these models have been optimized to generate predictions on a “collection of dataset” at a time. Use command below to accomplish this:

“# Add the image to a batch where it's the only member . *img = (np.expand\_dims(img,0))*  
*print(img.shape)*”  
“(1, 28, 28)”

Now, to generate the prediction for the picture by utilizing “tf.keras” use the command below:

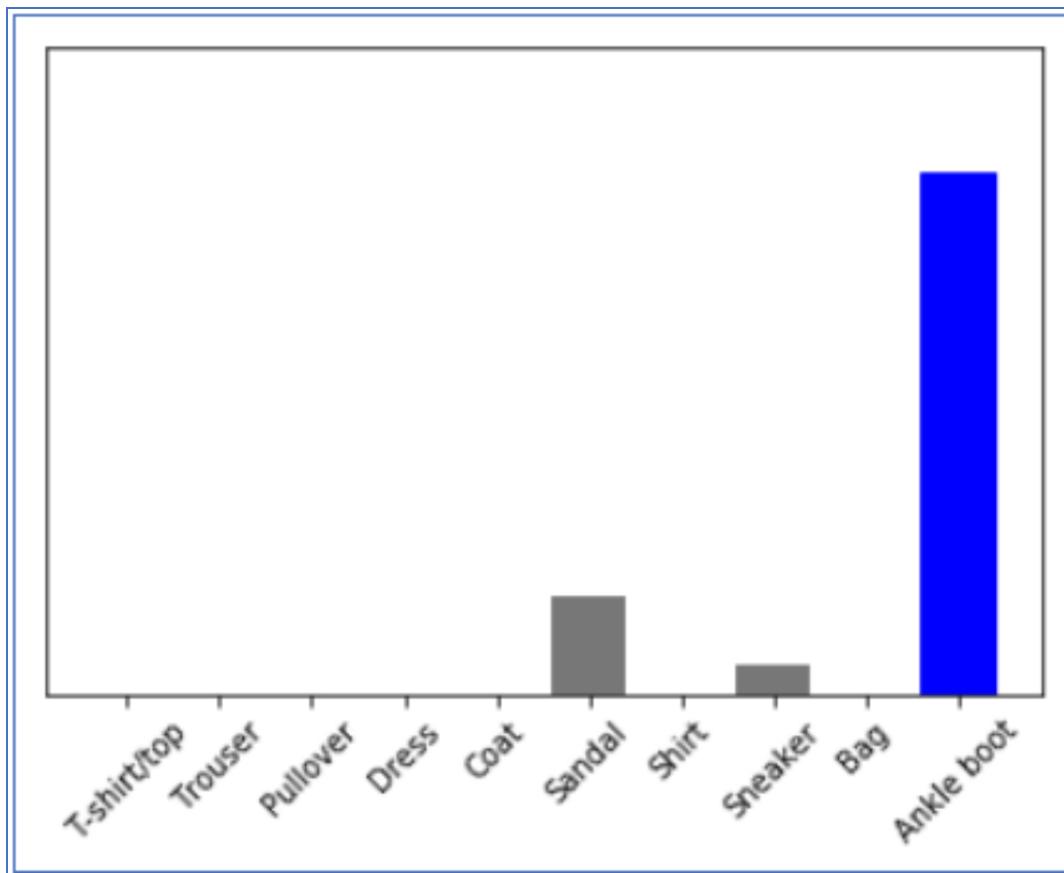
“*predictions\_single = model.predict(img)*  
*print(predictions\_single)*”

The predictions generated will resemble the code below:

```
“[[6.5837266e-06    1.3648087e-10    4.1718483e-08    1.1517859e-10  
8.3093937e-07  
1.4991476e-01    3.1148918e-06    4.6347316e-02    6.1082108e-05  
8.0366623e-01]]”
```

To generate a graph or plot for the prediction (as shown in the picture below), use command below:

```
“plot_value_array(0, predictions_single, test_labels )  
plt.xticks(range(10), class_names, rotation=45 )  
plt.show()”
```



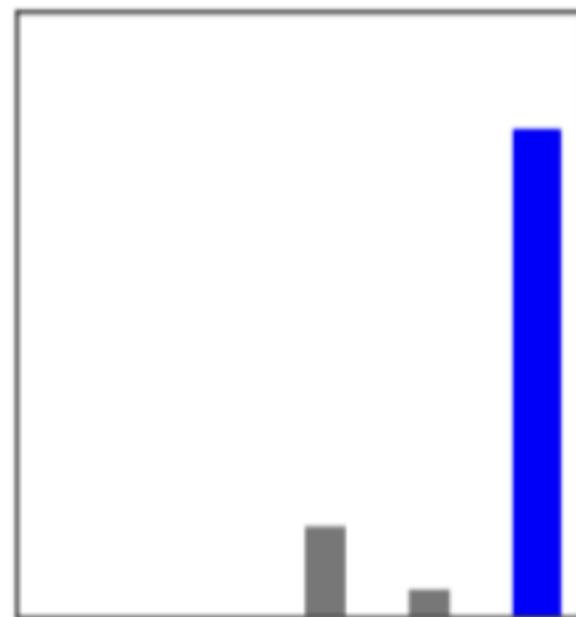
The “*model.predict*” generated the output as a “list of lists” for every single picture in the testing data set. To generate predictions specifically for the specific image, we used earlier, use command below:

```
“prediction_result = np.argmax(predictions_single[0] )  
print(prediction_result)”
```

The output or prediction generated should be “9”, as we obtained earlier.



Ankle boot 80% (Ankle boot)



## Extra content

*Python Machine Learning: An hands-on introduction to artificial intelligence coding, a project-based guide with practical exercises (Book 2)* has been structured as a 7-day-course with seven chapter (one per day), to guide the reader in a journey into the huge worl of Python.

The journey is thought and structured by Computer Programming Academy as a month long course. So it is just began!

This book is part of a series with other two:

- *Python Programming: An hands-on introduction to computer programming and algorithms, a project-based guide with practical exercises (Book 1)*
- *Python Data Science : An hands-on introduction to big data analysis and data mining, a project-based guide with practical exercises (Book 3)*

Here below a free sneak peak of Book 1 *Python Programming* and Book 3 *Python Data Science*, enjoy!

# **Python Programming: An hands-on introduction to computer programming and algorithms, a project-based guide with practical exercises (Book 1)**

This book will discuss various fundamental concepts of the Python programming language. There are 7 chapters in this book, crafted specifically to help you master basic and advanced python programming concepts required to develop web based programs and applications in just a week.

The first chapter of this book starts with an introduction to computer programming and some of the most widely used programming languages. You will also learn the fundamental elements of computer programming language such as basic operators, functions, decision making, among others. The importance of mathematical concepts such as algebra and statistics in computer programming has also been explained. Chapter 2 will provide a detailed overview of Python and its historical development. Step by step instructions to install Python on your operating systems have also been included. The concept of Python comments, variables and data types that serve as a prerequisite to the learning of Python programming have been explained in detail.

Chapter 3 is a detailed overview of the basic concepts of Python programming focusing on various programming elements such as Booleans, Tuples, Sets, Dictionaries and much more. The nuances of how to write efficient and effective Python codes have been explained in detail along with plenty of examples and sample exercises to help you solidify your understanding of these concepts. Chapter 4 pertains to the advance Python programming concepts that are relatively more complicated and require a solid understanding of the basic concepts. You will learn how to use OOPS concepts, different loops and conditional statements to generate sophisticated commands. This chapter also includes plenty of examples and sample exercises so you can verify what you have learned.

Like most programming languages, Python boasts a number of built-in functions to make your life easier while coding a software program. Chapter 5 contains a list of all such built-in functions, methods and

keywords that can be used to easily develop and run advance codes. Chapter 6 will provide a detailed overview of Django which is web framework that is popularly used in the development of web based programs and applications. You will learn how to install Django on your computer and follow the step by step coding instructions to develop your own web based program and notes taking application.

The final chapter, “Python Applications”, will provide details on how Python programming is being used in the development and testing of software programs, machine learning algorithms and Artificial Intelligence technologies to solve real world problems. These cutting edge technologies have resulted in tools and programs that are being utilized across the industrial spectrum to solve real world problems and become more futuristic. This chapter also includes various Python programming tips and tricks that will help you take your coding skills to the next level.

## **Day 1: Computer Programming 101**

Humans have evolved their medium of communication over centuries, resulting in a wide variety of languages spoken across the world. However, all manmade languages have a shared set of features that are remarkably standard across the board. Every language has a script containing different parts of a structured sentence such as nouns, verbs, adjectives, and other elements.

This is where we can draw a bridge to the computer programming languages, which are also composed of a variety of fundamental elements. We will look at each of these elements in detail later in this chapter. However, computer languages allow humans to interact with and guide the computing machines to perform desired operations. It allows the development and implementation of advance computing technologies. These programming languages also allow computers to interact with one another. Here is a quick overview of some of the most popular computer programming languages.

### **C**

C Language can be defined as a structure-oriented (functions are stored as a self-contained unit), medium level programming language widely

utilized in the development of “low-level” applications (pertaining to hardware components of the computer). In 1972, Bell Laboratories developed the C language for implementation in the UNIX system. A number of sophisticated and advanced programming languages such as Java, JavaScript, C++, C# and Perl are derived from the “grandmother” C language. Until the introduction of Java, the C language was the most dominant high-level language of the industry. Some of the only operating systems like IBM System/370 were also developed using the C language. The C language is rated low on the scale of learning difficulty primarily owing to the limited number of keywords,<sup>32</sup> that need to be trained on, and thus, it often serves as a foundational language for coding beginners. It is often used for the development of software applications that require integration to an operating system like UNIX, Linux and Windows. Some of the most popular C language based applications are: Word processors, OS development, database systems, network drivers and interpreters, compilers and assemblers, spreadsheets and graphics packages. Facebook’s TAO systems are developed using the C language.

## C++

Developed in 1983 as an extension of the C language, C++ can be defined as an object oriented (grouping of function and the associated dataset into an object), “medium level” (interacting with the programming layer of the computer) programming language that can be used for the development of general purpose software. It allows coding in similar syntax as the C language making C++ a perfect example of a “hybrid language”. The C++ language with a robust standard library and (STL) quick processing and compilation mechanism, is used to develop various application suites such as “Microsoft Office”, graphics editing tools, video editors, gaming packages and even entire Operating Systems. The “BlackBerry” operating system and the latest Microsoft Office suite are developed entirely on the C++ language. The C++ language is widely perceived as the enhanced version of the C language with object oriented up to approach that can be used to generate efficient and lean code. It also provides a high level of abstraction to allow improved management of large development projects. The C++ language tends to be the first programming language taught at college level. Some of the major companies and organizations using C++ are Amazon, Google, Adobe software, Mozilla,

Winamp, and Lockheed Martin. The C++ language is specifically used in the development of Embedded Firmware, Client Server Applications, Drivers, and system programs .

## C#

In 2000, Microsoft released C# (pronounced as C-sharp) as part of its .Net framework, which was developed using other languages like C, C++, and Java as a foundational basis. In 2003, C# became an ISO certified multi-paradigm programming language with powerful features including high functionality, object oriented, imperativeness, declarative attributes and component orientation. Developers use C# a lot to write codes for the XML web services applications as well as Applications connected with Microsoft .Net for the Windows Operating System. The C# language is the go to programming language for Microsoft applications and the language of choice for the Windows Presentation Foundation (WPF). With the introduction of .Net Standard and .Net Core, the .Net ecosystem evolved into cross-platform frameworks and standards, capable of running on Windows, Linux and Mac. The C# language is ideal for beginners and has similar capabilities as Java. It is a high level programming language with high similarity to the English language reading, making it easy to learn and use. It is still not as high level and easy to learn for beginners as Python. Game development is another popular application for the C# language, said to be the language of choice to develop and enhance games on the “Unity Game Engine”. Developers can write android and iOS applications in C# using Microsoft Xamarin framework.

## Java

Java, now owned by Oracle, was introduced in 1991 by Sun Microsystems as a high-level, memory managed language called “Oak” to add capabilities to the C++ language. It is the leading development language and framework with features like general-purpose, object-oriented making it ideal for web based application development. Java runs on the principle of WORA (Write Once Run Anywhere) and has cross-platform capability, making it ideal for developing mobile and gaming applications at the enterprise level. The Java Server Pages (JSP) is used to develop web based applications. Java allows applications to be accessed through a browser and easily downloadable. The Java byte code is compiled from the Java language and runs on the Java Virtual Machine (JVM).

These JVMs are available for a majority of operating systems like Windows and Mac. Some programs that are developed using Java are Eclipse, Lotus Notes, Minecraft, Adobe Creative Suite and open office. Google's Android operating system and app development are primarily driven by Java. It is a robust and interpreted language with high application portability, extensive network library and automatic memory management.

## **JavaScript**

Due to a similarity in the name, people often assume that there is an underlying connection to Java, but it's far from the truth. JavaScript was developed in 1995 by the company Netscape and called "LiveScript". JavaScript processes commands on the computer instead of a server and runs inside a client browser. It is primarily used in web development to make webpages more dynamic and manipulate various elements such as: creating a calendar functionality, printing time and date, adding webpage scrolling abilities and other features that cannot be developed using plain HTML.

Web server called NodeJS runs entirely on JavaScript on the server-side. JavaScript is frequently used by front-end web developers and game developers in a variety of domains such as marketing, information technology, engineering, healthcare and finance. A British agency called Cyber-Duck was developed with JavaScript and uses public APIs to access data concerning crime and enable authorities to review and safeguard local areas. Pete Smart and Robert Hawkes created "Tweetmap", that serves as a depiction of the world map in proportion to the number of "tweets" generated by each country. The fundamental features of JavaScript are considered relatively easy to understand and master. A comprehensive JavaScript library called "JQuery" containing multiple frameworks is widely used by the developers as reference.

## **Python**

Python first introduced in 1989 and is touted as extremely user-friendly and easy to learn programming language for amateurs and entry level coders. It is considered perfect for people who have newly taken up interest in programming or coding and need to understand the fundamentals of programming. This emanates from the fact that Python reads almost like English language. Therefore, it requires less time to understand how the

language works and focus can be directed in learning the basics of programming. Here are some of the fundamental elements of computer programming language:

- **Data Type** – This concept is applicable to every programming language ever designed. The data type is simply a representation of the type of data that needs to be processed by the computer. Some of the most common data types are string, numeric, alphanumeric, decimals, among others. Each programming language has its own definition of the data types and keywords used to write the code. For example, the keyword “char” is used to define string data type in C and Java.
- **Variable** – Data values can be stored on a computer by specifying desired label or name to select computer memory locations. These labels are referred to as variables. For instance, you can store values like “Christmas is on” and “12/25” using variables like “A” and “B” and then subsequently execute scientific program to retrieve desired output. Every programming language will have unique keywords and syntax to create and use required variables.
- **Keywords** – Each programming language has a basic syntax with certain words reserved to indicate specific meaning and cannot be used to create variable names. For example, C programming language used words like “int” and “float” to indicate data types; therefore, you will not be able to create variables named “int” or “float”.
- **Basic Operators** – Programming language operators refer to symbols that inform the program compiler to perform the indicated mathematical, logical or relational operation and produce desired output. For example, the arithmetic operator “+” in C programming language will execute the addition

command on indicated values. Similarly, relational operator “>” will allow you to compare data values and generate true or false result.

•

**Decision Making** – This element pertains to selection of one of the provided options on the basis of the provided conditions. For instance, if a remark needs to be printed, the programming code needs to include one or more required conditional statements that will be processed through the workflow of the program. “If” and “If else” conditional statements are some of the decision making statements used in C and Python.

•

**Functions** – A set of reusable and organized code that can be utilized to execute a related action is called as function. They offer enhanced modularity for the app and more reusability of the code. For instance, built in functions like “main ()” or “printf ()” can be written and used in C programming language. Different languages refer to functions using different terminologies like subroutine, method, or procedure.

•

**File I/O** – Data values can be stored in various formats such as images, plain texts, rich media, and more using computer files. You can organize these files into distinct directories. In short, files hold data and directories hold files. For instance , the extension “.c” will be added to the end of C programming files and extension “.java” to all Java files. The input files can be created in text editing tools like MS Word or Notepad and output files allows reading of the data from the file. The output files are used to show the results on the screen by executing the input to the program from the command prompt.

## **Importance of Mathematics in Computer Programming**

The discipline of mathematics is extremely important to learn and understand the fundamental concepts of computer programming. Various concepts of “Discrete Mathematics”, such as probability, algebra, set theory, logic notation, among others, are intricate parts of computer programming. Algebra is frequently used in programming languages. For example, “Boolean Algebra” can be utilized in logical operations and “Relational Algebra” can be utilized in databases. Another example is used of “Number Theory” in the development of cryptocurrency.

Computer science algorithms, including machine learning algorithms, consist of a set of instructions required in the implementation of an application or program. A basic algorithm is as simple as a mathematics statement written using logical operator “+” ( $5+7 = 12$ ) to code for the addition of data values. The whole concept of data analysis and problem solving is dependent on the mathematical equations that are analyzed to understand the crux of an error. By addressing those issues directly using the mathematics of the program, hard fixes can be easily made to the application.

Statistics is also widely used in data mining and compression, as well as speech recognition and image analysis software. The field of Artificial Intelligence and Machine Learning share a lot of core concepts from the field of statistics. “Statistical learning” is a descriptive statistics-based learning framework that can be categorized as supervised or unsupervised. “Supervised statistical learning” includes constructing a statistical model to predict or estimate output based on single or multiple inputs, on the other hand, “unsupervised statistical learning” involves inputs but no supervisory output, but helps in learning data relationships and structure. One way of understanding statistical learning is to identify the connection between the “predictor” (autonomous variables, attributes) and the “response” (autonomous variable), in order to produce a specific model which is capable of predicting the “response variable (Y)” on the basis of “predictor factors (X)” .

“ $X = f(X) + \varepsilon$  where  $X = (X_1, X_2, \dots, X_p)$ ” , where “f” is an “*unknown function*” & “ $\varepsilon$  ” is “*random error (reducible & irreducible)*” .

If there are a number of inputs "X" easily accessible, but the output "B" production is unknown, "f" is often treated as a black box, provided

that it generates accurate predictions for "Y". This is called "prediction". There are circumstances in which we need to understand how "Y" is influenced as "X" changes. We want to estimate "f" in this scenario, but our objective is not simply to generate predictions for "Y". In this situation, we want to establish and better understand the connection between "Y" and "X". Now "f" is not regarded as a black box since we have to understand the underlying process of the system. This is called "inference". In everyday life, various issues can be categorized into the setting of "predictions", the setting of "inferences", or a "hybrid" of the two.

The "parametric technique" can be defined as an evaluation of "f" by calculating the set parameters (finite summary of the data) while establishing an assumption about the functional form of "f". The mathematical equation of this technique is " $f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p$ ". The "parametric models" tend to have a finite number of parameters which is independent of the size of the data set. This is also known as "model-based learning". For example, "k-Gaussian models" are driven by parametric techniques.

On the other hand, the "non-parametric technique" generates an estimation of "f" on the basis of its closeness to the data points, without making any assumptions on the functional form of "f". The "non-parametric models" tend to have a varying number of parameters, which grows proportionally with the size of the data set. This is also known as "memory-based learning". For example, "kernel density models" are driven by a non-parametric technique.

## **Python Data Science: An hands-on introduction to big data analysis and data mining, a project-based guide with practical exercises (Book 3)**

This book will discuss the fundamental concepts of data science technologies that can be used to analyze raw data and generate predictions and resolve business problems.

The first chapter of the book will help you understand the importance of data science technologies in our everyday lives ranging from weather forecasting to cyber attacks. You will also learn different types of data and various data science implementation strategies. A detailed overview of the “Team Data Science Process”, which is a data science lifecycle widely used for projects that require the deployment of applications based on artificial intelligence and/or machine learning algorithms, has been provided in the second chapter. You will learn the objectives defined at each of the 5 stages of this lifecycle along with the deliverables that are created at the end of each stage.

The third chapter is all about big data and big data analytics. You will learn the 5 Vs of big data and the 3 important actions required to gain insights from big data. You will also learn the different steps involved in big data analysis and some of its applications in healthcare, finance, and other industrial sectors. The chapter entitled “Basics of Data Mining” will provide an explicit overview of the data mining process and its applications. You will also learn the advantages and challenge of the data mining process in resolving real world data problems. Some of the most widely used data mining tools that are being used by data analysts are also explained.

The fifth chapter deals exclusively with some of the key data analysis frameworks, including ensemble learning, decision trees and random forests. These are the most popular machine learning algorithms that are capable of processing a large volume of unstructured and unorganized data to generate useful insights and predictions. You will learn the advantages and disadvantages of these frameworks as well as the steps required to implement random forest regression on a real life dataset. Chapter six, entitled “Data Analysis Libraries”, is a deep dive into the functioning of

different Python based data analysis libraries including IPython, Jupyter Notebook, Pandas, Matplotlib among others. You will learn how these powerful libraries can be used to analyze real life data set with select open source sample dataset that you can download and gain hands-on experience with.

The final chapter of this book will explain to you how the data analysis helps resolve business issues using customer and/or predictive analytics. Customer analytics is at the heart of all marketing activities and is an umbrella term used for techniques such as “predictive modeling”, “data visualization”, “information management”, and “segmentation”. You will learn the important concept of marketing and sales funnel analytics as well as the three main types of predictive models to analyze customer behavior. The concepts of exploratory analysis of customer data and personalized marketing have been explained in detail, along with some of their industrial applications. To make the best use of this book, I recommend that you download the free resources provided in this book and perform hands-on exercises to solidify your understanding of the concepts explained. The skillset of data analysis is always in demand, with a lot of high pay job opportunities. Here’s hoping this book has taken you a step closer to your dream job !

We have also provided review exercises to help you test your understanding through this process. Every chapter of this book has real life examples and applications included to solidify your understanding of each concept.

## **Day 1: Introduction to Data Science**

In the world of technology, Data is defined as “information that is processed and stored by a computer”. Our digital world has flooded our realities with data. From a click on a website to our smart phones tracking and recording our location every second of the day, our world is drowning in the data. From the depth of this humongous data, solutions to our problems that we have not even encountered yet could be extracted. This very process of gathering insights from a measurable set of data using mathematical equations and statistics can be defined as “data science”. The role of data scientists tends to be very versatile and is often confused with a computer scientist and a statistician. Essentially anyone, be it a person or a

company that is willing to dig deep to large volumes of data to gather information, can be referred to us data science practitioners. For example, companies like Walmart keeps track of and record of in-store and online purchases made by the customers, to provide personalized recommendations on products and services. The social media platforms like Facebook that allow users to list their current location is capable of identifying global migration patterns by analyzing the wealth of data that is handed to them by the users themselves.

The earliest recorded use of the term data science goes back to 1960 and credited to Pete Naur, who reportedly used the term data science as a substitute for computer science and eventually introduced the term “datalogy”. In 1974, Naur published a book titled “Concise Survey of Computer Methods”, with liberal use of the term data science throughout the book. In 1992, the contemporary definition of data science was proposed at “The Second Japanese-French Statistics Symposium”, with the acknowledgment of emergence of a new discipline focused primarily on types, dimensions and structures of data.

*“Data science continues to evolve as one of the most promising and in-demand career paths for skilled professionals. Today, successful data professionals understand that they must advance past the traditional skills of analyzing large amounts of data, data mining, and programming skills. In order to uncover useful intelligence for their organizations, data scientists must master the full spectrum of the data science life cycle and possess a level of flexibility and understanding to maximize returns at each phase of the process”.*

– University of California, Berkley

An increasing interest by business executives has significantly contributed to the recent rise in popularity of the term data science. However, a large number of journalists and academic experts, do not acknowledge data science as a separate area of study from the field of statistics. A group within the same community considers data science is the popular term for “data mining” and “big data”. The very definition of data science is up for debate within the tech community. The field of study that requires a combination of skill set including computer programming skills, domain expertise and proficiency in statistics and mathematical algorithms

to be able to extract valuable insight from large volumes of raw data is referred to as data science .

## **Importance of Data Science**

Data Science is heavily used in Predictive analysis. For example, weather forecast requires collection and analysis of data from a variety of sources, including satellites, radars, and aircraft, to build data models that are even capable of predicting the occurrence of catastrophes of the nature such as hurricanes, tornadoes and flash floods. Another branch of data science is “big data and big data analytics”, which are used by organizations to address complex technical problems as well as for resource management. You will learn more about big data later in this book. The ability of data science to analyze the challenges facing any and all industrial sectors like healthcare, travel, finance, retail, and e-commerce has contributed significantly to its increasing popularity among business executives.

Data science has made the use of advanced machine learning algorithms possible, which has a wide variety of applicability across multiple industrial domains. For example, the development of self-driving cars that are capable of collecting real-time data using their advanced cameras and sensors to create a map of their surroundings and make decisions pertaining to the speed of the vehicle and other driving maneuvers. Companies are always on the prowl to better understand the need of their customers. This is now achievable by gathering the data from existing sources like customer’s order history, recently viewed items, gender, age and demographics and applying advanced analytical tools and algorithms over this data to gain valuable insights. With the use of machine learning algorithms, the system can generate product recommendations for individual customers with higher accuracy. The smart consumer is always looking for the most engaging and enhanced user experience, so the companies can use these analytical tools and algorithms to gain a competitive edge and grow their business.

The ability to analyze and closely examine Data trends and patterns using Machine learning algorithms has resulted in the significant application of data science in the cybersecurity space. With the use of data science, companies are not only able to identify the specific network terminal(s) that initiated the cyber attack but are also in a position to predict potential future attacks on their systems and take required measures to

prevent the attacks from happening in the first place. The use of “active intrusion detection systems” that are capable of monitoring users and devices on any network of choice and flag any unusual activity serves as a powerful weapon against hackers and cyber attackers. While the “predictive intrusion detection systems” that are capable of using machine learning algorithms on historical data to detect potential security threats serves as a powerful shield against the cyber predators.

Cyber attacks can result in a loss of priceless data and information resulting in extreme damage to the organization. To secure and protect the data set, sophisticated encryption and complex signatures can be used to prevent unauthorized access. Data science can help with the development of such impenetrable protocols and algorithms. By analyzing the trends and patterns of previous cyber attacks on companies across different industrial sectors, Data science can help detect the most frequently targeted data set and even predict potential future cyber attacks. Companies rely heavily on the data generated and authorized by their customers but in the light of increasing cyber attacks, customers are extremely wary of their personal information being compromised and are looking to take their businesses to the companies that are able to assure them of their data security and privacy by implementing advanced data security tools and technologies. This is where data science is becoming the saving grace of the companies by helping them enhance their cybersecurity measures. Over the course of the past 20 years, Data trends have drastically changed, signaling an ongoing increase in unstructured data. It is estimated that by the year 2020, “more than 80% of the data that we gather will be unstructured”. Conventionally, the data that we procured was primarily structured and could be easily analyzed using simple business intelligence tools but as reflected in the picture below unstructured and semi-structured data is on the rise. This, in turn, has warranted the development and use of more powerful and advanced analytical tools than the existing business intelligence tools that are incapable of processing such large volume and variety of data.

## **Types of Data**

Let us look at different types of data so you can choose the most appropriate analytical tools and algorithms based on the type of data that

needs to be processed. Data types can be divided into two at a very high level: qualitative and quantitative .

**Qualitative data** – Any data that cannot be measured and only observed subjectively by adding a qualitative feature to the object it's called as "qualitative data". Classification of an object using unmeasurable features results in the creation of qualitative data. For example, attributes like color, smell, texture, and taste. There are three types of qualitative data:

- 

**"Binary or binomial data"** – Data values that signal mutually exclusive events where only one of the two categories or options is correct and applicable. For example, true or false, yes or no, positive or negative. Consider a box of assorted tea bags. You try all the different flavors and group the ones that you like as "good" and the ones you don't as "bad". In this case, "good or bad" would be categorized as the binomial data type. This type of data is widely used in the development of statistical models for predictive analysis.

- 

**"Nominal or unordered data"** – Data characteristics that lack an "implicit or natural value" can be referred to as nominal data. Consider a box of M&Ms, you can record the color of each M&M in the box in a worksheet, and that would serve as nominal data. This kind of data is widely used to assess statistical differences in the data set, using techniques like "Chi-Square analysis", which could tell you "statistically significant differences" in the amount of each color of M&M in a box.

- 

**"Ordered or ordinal data"** – The characteristics of this Data type do have certain "implicit or natural of value" such as small, medium, or large. For example, online reviews on sites like "Yelp", "Amazon", and "Trip Advisor" have a rating scale from 1 to 5, implying a 5-star rating is better than 4 which is better than 3 and so on.

**Quantitative data** – Any characteristics of the data that can be measured objectively are called as "quantitative data". Classification of an object in using measurable features and giving it a numerical value results and creation of quantitative data. For example, product prices, temperature, dimensions like length, etc. There are two types of quantitative data:

- 

**"Continuous Data"** – Data values that can be defined to a further lower level, such as units of measurement like kilometers, meters, centimeters, and on and on, are called the continuous data type. For example, you can purchase a bag of almonds by weight like 500 g or 8 ounces. This accounts for the continuous data type, which is primarily used to test and verify different kinds of hypotheses such as assessing the accuracy of the weight printed on the bag of almonds.

- 

**"Discrete Data"** – numerical data value that cannot be divided and reduced to a higher level of precision, such as the number of cars owned by a person which can only be accounted for as indivisible numbers (you cannot have 1.5 or 2.3 cars), is called as discrete data types. For example, you can purchase another bag of ice cream bars by the number of ice cream bars inside the package, like four or six. This accounts for the discrete data type, which can be used in combination with a continuous data type to perform a regression analysis to verify if the total weight of the ice cream box (continuous data) is correlated with the number of ice cream bars (discrete data) inside.

## **Data science strategies**

Data science is mainly used in decision-making by making precise predictions with the use of “predictive causal analytics”, “prescriptive analytics”, and machine learning.

**Predictive causal analytics** –“predictive causal analytics” can be applied to develop a model that can accurately predict and forecast the

likelihood of a particular event occurring in the future. For example, financial institutions use predictive causal analytics based tools to assess the likelihood of a customer defaulting on their credit card payments, by generating a model that can analyze the payment history of the customer with all of their borrowing institutions.

**Prescriptive analytics** - The “prescriptive analytics” are widely used in the development of “intelligent tools and applications” that are capable of modifying and learning with dynamic parameters and make their own “decisions”. The tool not only predicts the occurrence of a future event but is also capable of providing recommendations on a variety of actions and its resulting outcomes. For example, the self driving cars gather driving related data with every driving experience and use it to train themselves to make better driving and maneuvering decisions.

**Machine learning to make predictions** – To develop models that can determine future trends based on the transactional data acquired by the company, machine learning algorithms are a necessity. This is considered as “supervised machine learning”, which we will elaborate on later in this book. For example, fraud detection systems use machine learning algorithms on the historical data pertaining to fraudulent purchases to detect if a transaction is fraudulent.

**Machine learning for pattern discovery** – To be able to develop models that are capable of identifying hidden data patterns but lack required parameters to make future predictions, the “unsupervised machine learning algorithms”, such as “Clustering”, need to be employed. For example, telecom companies often use the “clustering” technology to expand their network by identifying network tower locations with optimal signal strength in the targeted region.

## **Machine Learning Vs Data Science**

Data science is an umbrella term that encompasses machine learning algorithms. Here are some basic distinctions between the two terms.

### **Data components**

Data science pertains to the complete lifecycle of data and involves a variety of components including “ETL” (Extract, Transform, Load) pipeline

to collect and classify data, Data visualization, distributed computing, machine learning, artificial intelligence, Data engineering, dashboards, and System deployment introduction and environment among other components.

Machine learning models are provided input data and contain various components including: data separation, data exploration, problem-solving and appropriate model selection among other features.

### **Performance measures**

Data science has no standard for performance measurement and is determined on a case-by-case basis. Typically, performance measures are an indication of Data quality, Data timeliness, Data accessibility, Data visualization capability and data query capability.

Machine learning models have standard performance measures, with each algorithm having a measure to indicate the success of the model and describing the given training data set. For example, in “linear regression analysis”, the “Root Mean Square Error (RME) serves as an indication error(s) in the model.

### **Development method**

Data science project implementations are carried out in defined stages with project milestones that must be reached to fulfill set goals and targets within the constraints of time and resources. Machine learning projects are research-based and start with a hypothesis that is expected to be verified within the constraints of available data.

If you enjoyed this preview be sure to check out the full books on Amazon.com. Complete the journey and become a Python master!

# **Conclusion**

Thank you for making it through to the end of *Python Machine Learning: An hands-on introduction to artificial intelligence coding, a project-based guide with practical exercises (Book 2)* , let's hope it was informative and able to provide you with all of the tools you need to achieve your goals whatever they may be.

The next step is to make the best use of your machine learning knowledge that has significantly contributed to the creation of the “Silicon Valley” powerhouse. The use of machine learning technology has given rise to sophisticated machines, which are capable of studying human behaviors and activities in order to recognize any consistent patterns and generate predictions of products and services their consumers may be interested in. Under the shadow of their business model most futuristic businesses are gradually become technology firms with enterprise wide systems built upon machine learning algorithms. Consider some of the most innovative tech gadgets of the modern world that have been created in the past decade, such as “Amazon Alexa”, “Apple’s Siri”, and “Google Home”. The one aspect they all have in common is their underlying machine learning capabilities. Now that you have finished reading this book and mastered the use of Scikit-Learn and TensorFlow libraries, you are all set to start developing your own machine learning model by utilizing all the open sources readily available and explicitly mentioned in this book for that purpose .