Machine Learning

BY: DR. DARSHAN INGLE.

Objectives:

1. Make a learner ready for solving real life problems using ML

2. Understand components of a machine learning algorithm.

3. Apply machine learning tools to build and evaluate predictors.

4. How machine learning uses computer algorithms to search for patterns in data.

5. How to uncover hidden themes in large collections of documents using topic modelling.

6. How to use data patterns to make decisions and predictions with real-world examples

7. How to prepare data, deal with missing data and create custom data analysis solutions for different industries.

8. Understand the possibilities and limitations of ML, and know how to formulate your own ML problem.

9. Be ready to learn "NLP using ML".

Introduction

- Data is the new oil and Machine Learning is a powerful concept and framework for making the best out of it.

- The idea of making intelligent, sentient, and self-aware machines is not something that suddenly came into existence in the last few years.

- In fact a lot of lore from Greek mythology talks about intelligent machines and inventions having self-awareness and intelligence of their own.

- With faster computers, better processing, better computation power, and more storage, we have been living in what I like to call, the **“age of information”** or the **“age of data”.**

- Day in and day out, we deal with managing *Big Data* and building intelligent systems by using concepts and methodologies from *Data Science*, *Artificial Intelligence*, *Data Mining*, and *Machine Learning*.

Introduction

- Of course, most of you must have heard many of the terms I just mentioned and come across sayings like *“data is the new oil”*.
- The main challenge that businesses and organizations have embarked on in the last decade is to use approaches to try to make sense of all the data that they have and use valuable information and insights from it in order to make better decisions.
- Indeed with great advancements in technology, including availability of cheap and massive computing, hardware (including GPUs) and storage, we have seen a thriving ecosystem built around domains like Artificial Intelligence, Machine Learning, and most recently Deep Learning.

The Need for Machine Learning

- Human beings are perhaps the most advanced and intelligent lifeform on this planet at the moment.
- We can think, reason, build, evaluate, and solve complex problems.
- The human brain is still something we ourselves haven’t figured out completely and hence artificial intelligence is still something that’s not surpassed human intelligence in several aspects.
- Thus you might get a pressing question in mind as to why do we really need Machine Learning?
- What is the need to go out of our way to spend time and effort to make machines learn and be intelligent?
- The answer can be summed up in a simple sentence, **“To make data-driven decisions at scale”.**

Making Data-Driven Decisions

- Getting key information or insights from data is the key reason businesses and organizations invest heavily in a good workforce as well as newer paradigms and domains like Machine Learning and artificial intelligence.

- The idea of data-driven decisions is not new. Fields like operations research, statistics, and management information systems have existed for decades and attempt to bring efficiency to any business or organization by using data and analytics to make data-driven decisions.

- The art and science of leveraging your data to get actionable insights and make better decisions is known as making data-driven decisions.

Making Data-Driven Decisions

- Another important aspect of this problem is that often we use the power of reasoning or intuition to try to make decisions based on what we have learned over a period of time and on the job.
- Our brain is an extremely powerful device that helps us do so. *Consider problems like* understanding what your fellow colleagues or friends are speaking, recognizing people in images, deciding whether to approve or reject a business transaction, and so on. While we can solve these problems almost involuntary, can you explain someone the process of how you solved each of these problems? Maybe to some extent, but after a while, it would be like, “Hey! My brain did most of the thinking for me!”
- This is exactly why it is difficult to make machines learn to solve these problems like regular computational programs like computing loan interest or tax rebates.
- Solutions to problems that cannot be programmed inherently need a different approach where **we use the data itself to drive decisions instead of using programmable logic, rules, or code to make these decisions.**

Efficiency and Scale

- While getting insights and making decisions driven by data are of paramount importance, it also needs to be done with efficiency and at scale.
- The key idea of using techniques from Machine Learning or artificial intelligence is **to automate processes or tasks by learning specific patterns from the data.**
- We all want computers or machines to tell us when a stock might rise or fall, whether an image is of a computer or a television, whether our product placement and offers are the best, determine shopping price trends, detect failures or outages before they occur, and the list just goes on!
- While human intelligence and expertise is something that we definitely can’t do without, we need to solve real-world problems at huge scale with efficiency.

A REAL-WORLD PROBLEM AT SCALE

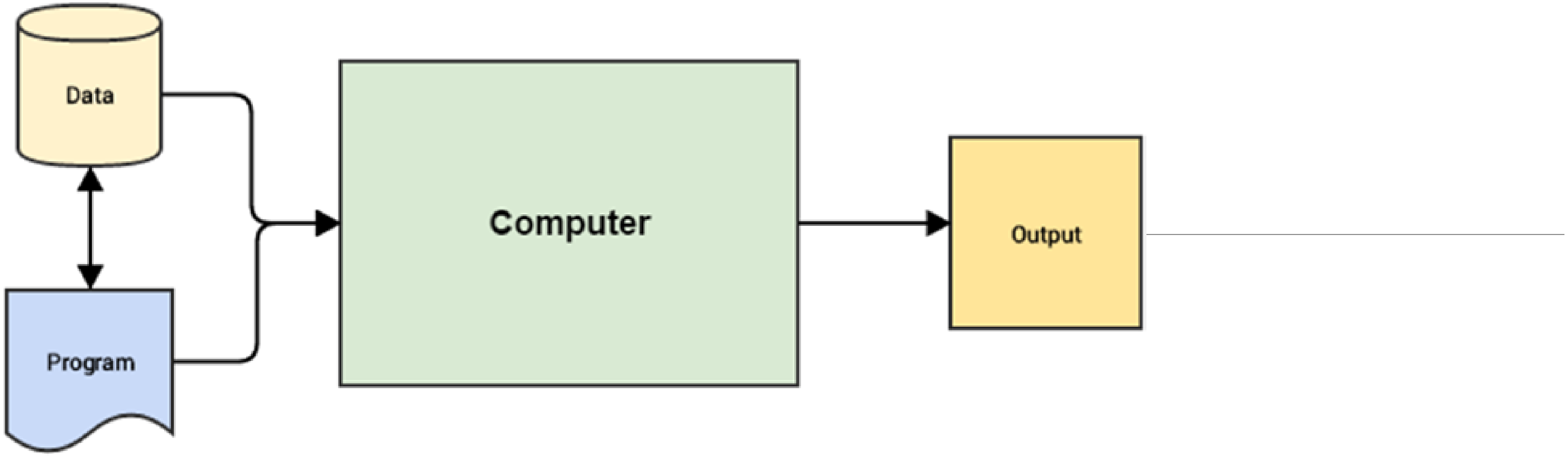
Consider the following real-world problem. You are the manager of a world-class infrastructure team for the DSS Company that provides Data Science services in the form of cloud based infrastructure and analytical platforms for other businesses and consumers. Being a provider of services and infrastructure, you want your infrastructure to be top-notch and robust to failures and outages. Considering you are starting out of St. Louis in a small office, you have a good grasp over monitoring all your network devices including routers, switches, firewalls, and load balancers regularly with your team of 10 experienced employees. Soon you make a breakthrough with providing cloud based Deep Learning services and GPUs for development and earn huge profits. However, now you keep getting more and more customers. The time has come for expanding your base to offices in San Francisco, New York, and Boston. You have a huge connected infrastructure now with hundreds of network devices in each building! How will you manage your infrastructure at scale now? Do you hire more manpower for each office or do you try to leverage Machine Learning to deal with tasks like outage prediction, auto-recovery, and device monitoring? Think about this for some time from both an engineer as well as a manager's point of view.

Traditional Programming Paradigm

- Computers, while being extremely sophisticated and complex devices, are just another version of our well

known idiot box, the television!

- “How can that be?” is a very valid question at this point.
- Let’s consider a television or even one of the so-called smart TVs, which are available these days. In theory as well as in practice, the TV will do whatever you program it to do. It will show you the channels you want to see, record the shows you want to view later on, and play the applications you want to play! The computer has been doing the exact same thing but in a different way.
- Traditional programming paradigms basically involve the user or programmer to write a set of instructions or operations using code that makes the computer perform specific computations on data to give the desired results.
- Figure on the next slide depicts a typical workflow for traditional programming paradigms.



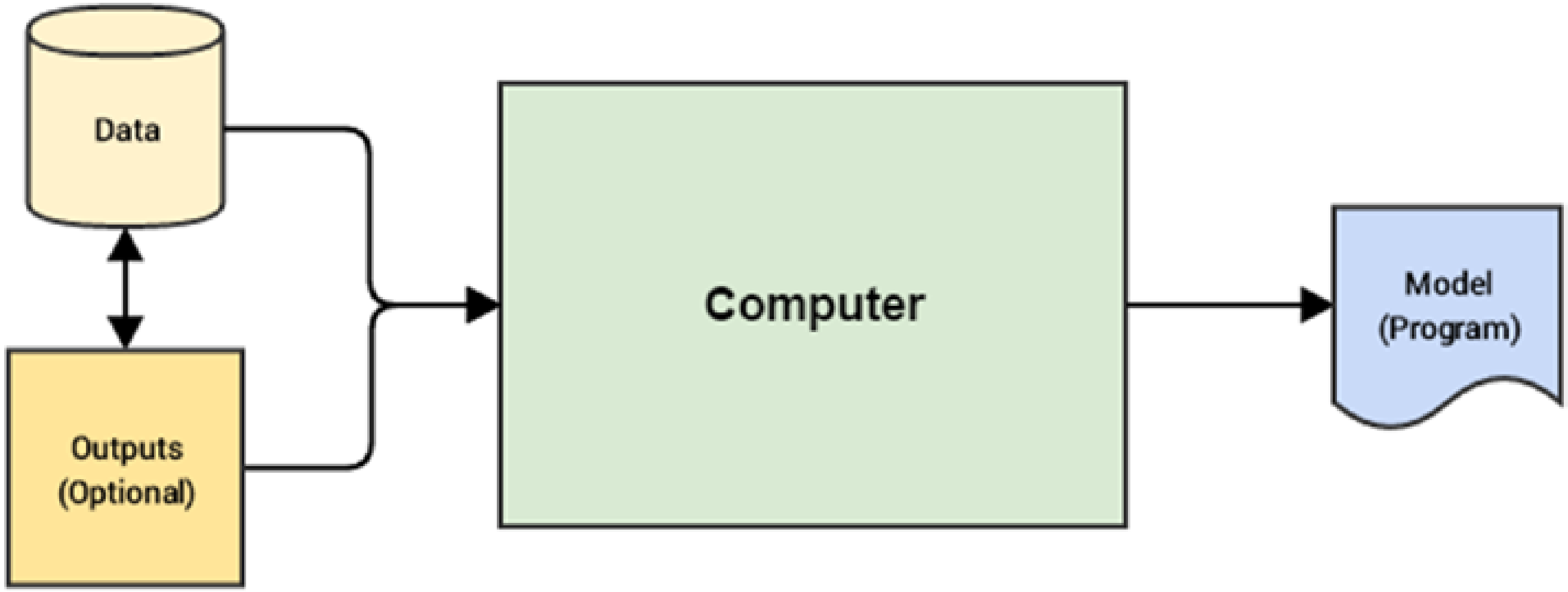
- You can get the idea that the core inputs that are given to the computer are data and one or more programs that are basically code written with the help of a programming language, such as high-level languages like Java, Python, or low-level like C.
- Programs enable computers to work on data, perform computations, and generate output. A task that can be performed really well with traditional programming paradigms is ***computing your annual tax***.
- Now, let’s think about the real-world infrastructure problem we discussed in the previous section for **DSS Company**.
- **Do you think a traditional programming approach might be able to solve this problem?**

Why Machine Learning?

- While the traditional programming paradigm is quite good and human intelligence and domain expertise is definitely an important factor in making data-driven decisions, we need *Machine Learning to make faster and better decisions*.

- The Machine Learning paradigm tries to *take into account data and expected outputs or results if any and uses the computer to build the program, which is also known as a* ***model***.

- This program or model can then be used in the future to make necessary decisions and give expected outputs from new inputs.



In the Machine Learning paradigm, the machine, in this context the computer, tries to use input data and expected outputs to try to learn inherent patterns in the data that would ultimately help in building a model analogous to a computer program, which would help in making data-driven decisions in the future (predict or tell us the output) for new input data points by using the learned knowledge from previous data points (its knowledge or experience).

We would not need hand-coded rules, complex flowcharts, case and if-then conditions, and other criteria that are typically used to build any decision making system or a decision support system.

The basic idea is to use Machine Learning to make insightful decisions.

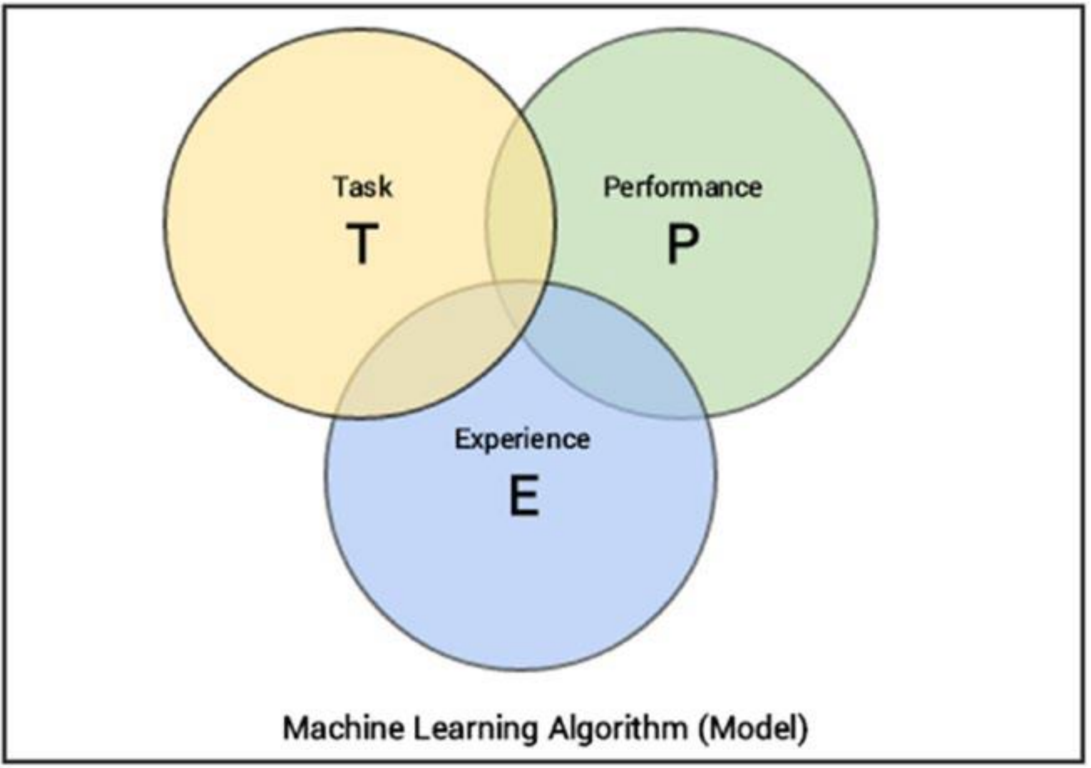
DSS Company.

- In the traditional programming approach, we talked about hiring new staff, setting up rule-based monitoring systems, and so on. If we were to use a Machine Learning paradigm shift here, we could go about solving the problem using the following steps.
- Leverage device data and logs and make sure we have enough historical data in some data store (database, logs, or flat files)
- Decide key data attributes that could be useful for building a model. This could be device usage, logs, memory, processor, connections, line strength, links, and so on.
- Observe and capture device attributes and their behavior over various time periods that would include normal device behavior and anomalous device behavior or outages. These outcomes would be your outputs and device data would be your inputs.
- Feed these input and output pairs to any specific Machine Learning algorithm in your computer and build a model that learns inherent device patterns and observes the corresponding output or outcome
- Deploy this model such that for newer values of device attributes it can predict if a specific device is behaving normally or it might cause a potential outage

- Thus once you are able to build a Machine Learning model, you can easily deploy it and build an intelligent system around it such that you can not only monitor devices reactively but you would be able to proactively identify potential problems and even fix them before any issues crop up.
- Imagine building self-heal or auto-heal systems coupled with round the clock device monitoring.
- The possibilities are indeed endless and you will not have to keep on hiring new staff every time you expand your office or buy new infrastructure.

Formal Definition of machine learning

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."



We can simplify the definition as follows. Machine Learning is a field that consists of learning algorithms that:

1. Improve their performance *P*
2. At executing some task *T*
3. Over time with experience *E*

Defining the Task, T

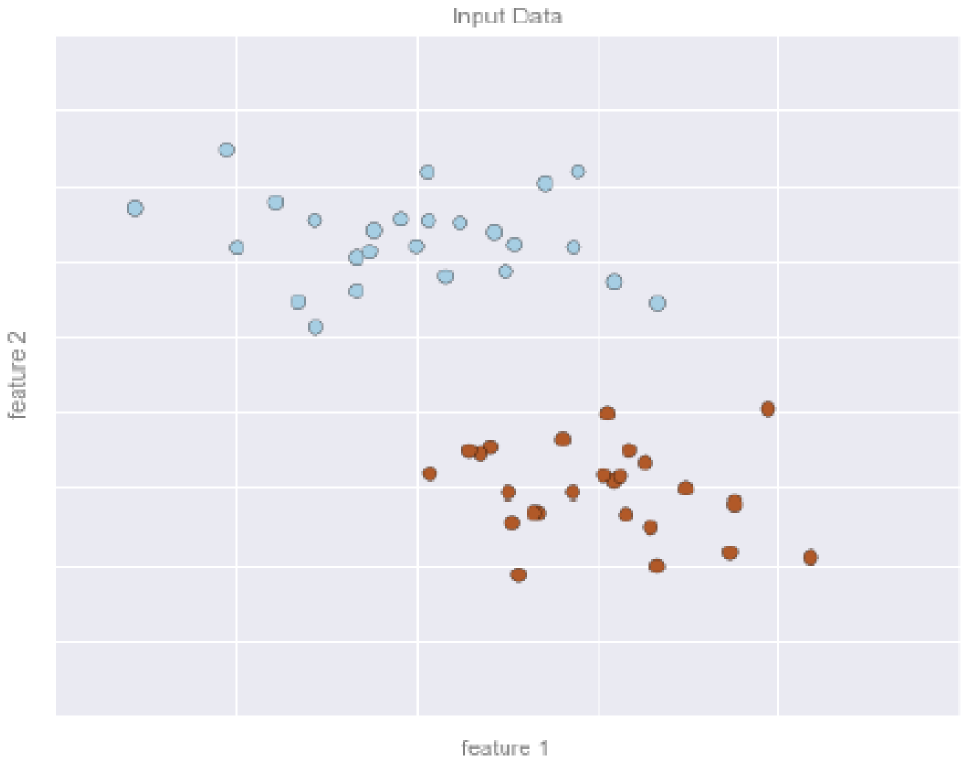
- Task, T, which can be defined in a two-fold approach. From a problem standpoint, the task, T, is basically the real-world problem to be solved at hand, which could be anything from ~~finding the best marketing or product mix to predicting infrastructure failures.~~
- In the Machine Learning world, it is best if you can define the task as concretely as possible such that you talk about what the exact problem is which you are planning to solve and how you could define or formulate the problem into a specific Machine Learning task.
- Machine Learning based tasks are difficult to solve by conventional and traditional programming approaches.
- A task, *T,* can usually be defined as a Machine Learning task based on the process or workflow that the system should follow to operate on data points or samples.
- Typically a data sample or point will consist of multiple data attributes (also called ***features*** in Machine Learning). A typical data point can be denoted by a ***vector (Python list)*** such that each element in the vector is for a specific data feature or attribute.

THE TYPICAL TASKS THAT COULD BE CLASSIFIED AS MACHINE LEARNING TASKS

1. **Classification or categorization**: This typically encompasses the list of problems or tasks where the machine has to take in data points or samples and assign a specific class or category to each sample. A simple example would be classifying animal images into dogs, cats, and zebras.

A simple data set for classification ➔

There are a number of possible models for such a classification task, the simplest is **drawing a straight line through the plane between them.**

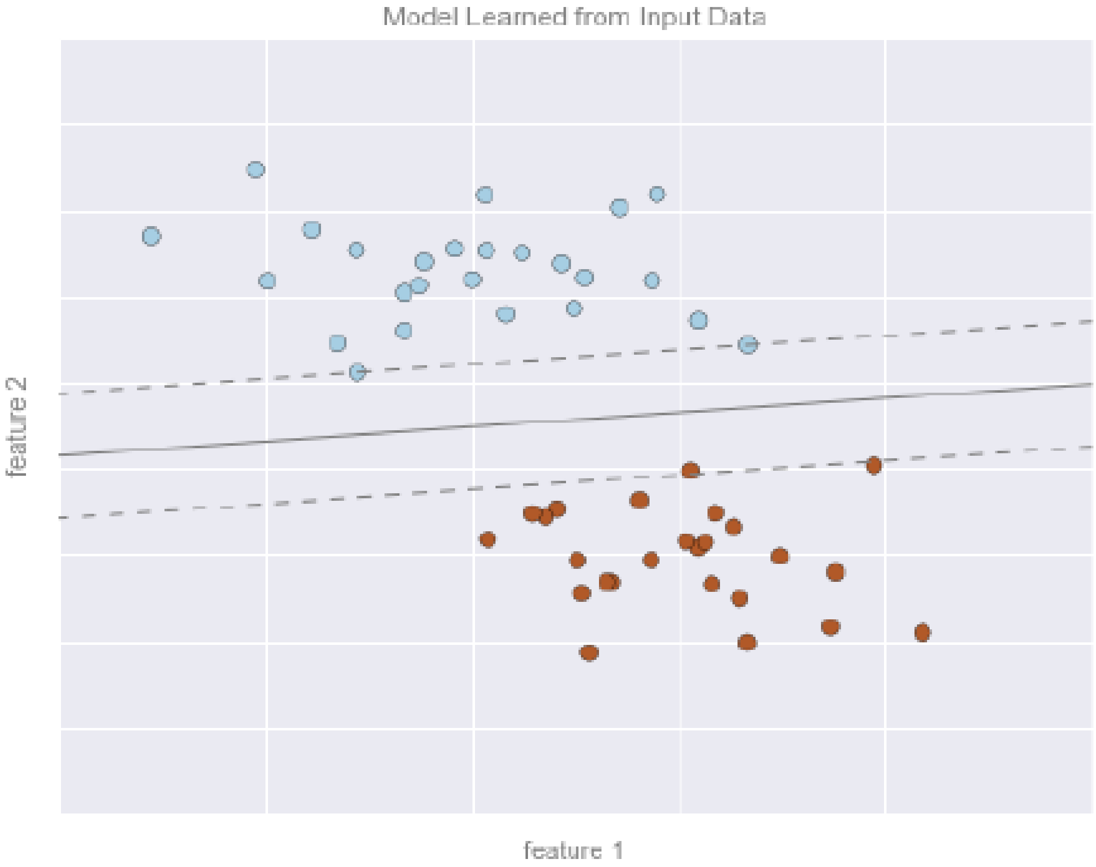


A simple classification model →

Now that this model has been trained, it **can be generalized to new, unlabeled data**.

In other words, we can take a new set of data, draw this model line through it, and

assign labels to the new points based on this model. **This stage is usually called *prediction*.**



- **For example**, this is similar to the task of **automated spam detection for email**; in this case, we might use the following features and labels: ***feature 1, feature 2,* etc**. normalized counts of important words or phrases (“Won a Lottery,” “Nigerian prince,” “HDFC bank statement” etc.) • ***label*** “spam” or “not spam”

- For the training set, these labels might be determined by individual inspection of a small representative sample of emails; for the remaining emails, the label would be determined using the model.

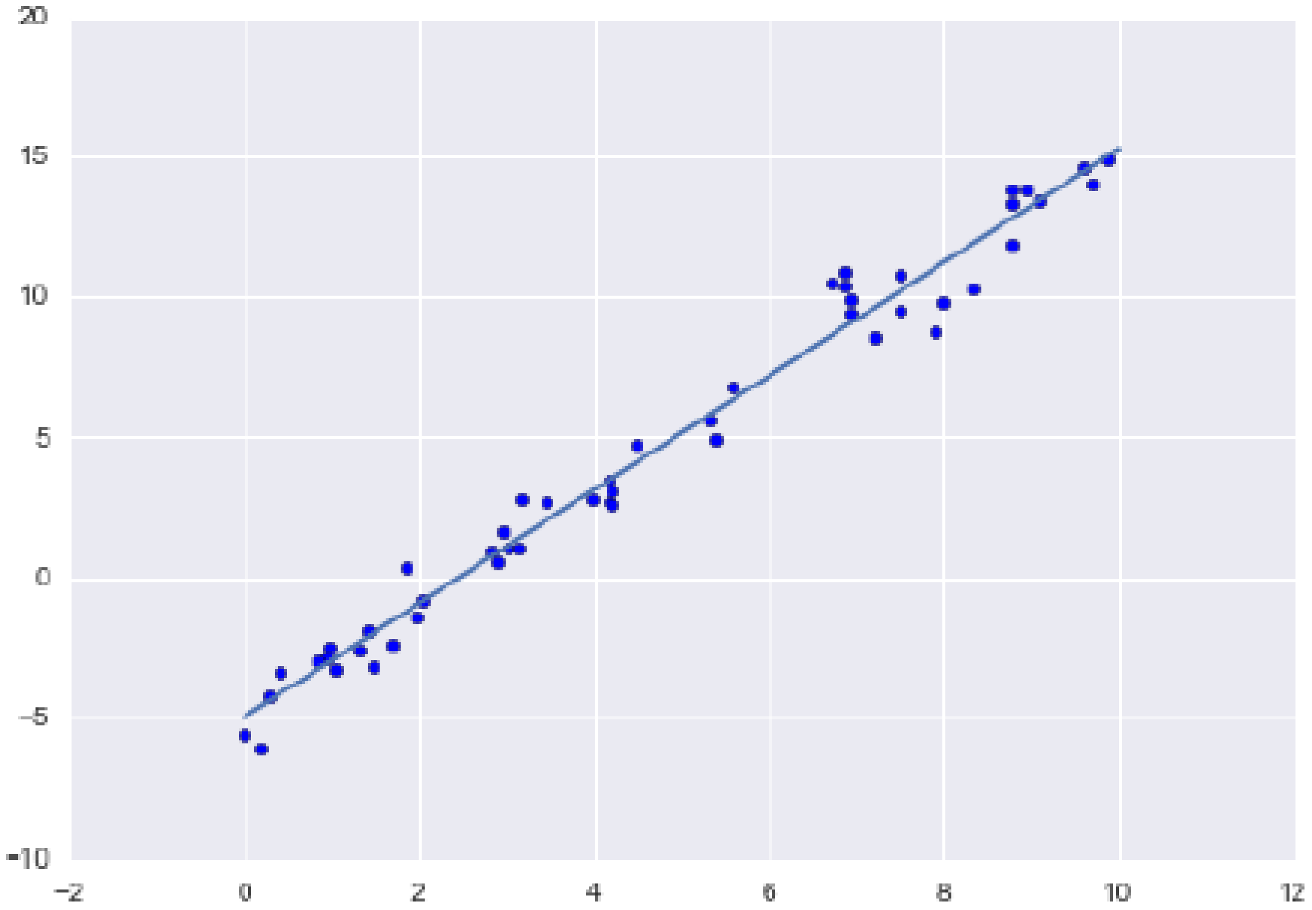
- For a suitably trained classification algorithm with enough well-constructed features (typically thousands or millions of words or phrases), this type of approach can be very effective.

- We will see an example of such text-based classification in “**Naive Bayes Classification**”

B. **Regression**:

These types of tasks usually involve performing a prediction such that a real numerical value is the output instead of a class or category for an input data point.

The best way to understand a regression task would be to take the case of a real-world problem of predicting housing prices considering the plot area, number of floors, bathrooms, bedrooms, and kitchen as input attributes for each data point.



Note:

The **classification** and **regression** examples, we just looked at are examples of **supervised learning** algorithms, in which we are trying to build a model that will predict labels for new data.

Unsupervised learning involves models that describe data without reference to any known labels. One common case of unsupervised learning is **clustering**, in which data is automatically assigned to some number of discrete groups.

Anomaly detection:

- These tasks involve the machine going over event logs, transaction logs, and other data points ~~such that it can find anomalous or unusual patterns or events that are different~~ from the normal behavior.

- Examples for this include trying to find denial of service attacks from logs, indications of fraud, and so on.

- This task **can be solved by supervised learning** like classification technique, but as the nature of attacks, hacks, viruses change very frequently, hence the trained model may not give accurate results.

- Better solution to this is **clustering.**

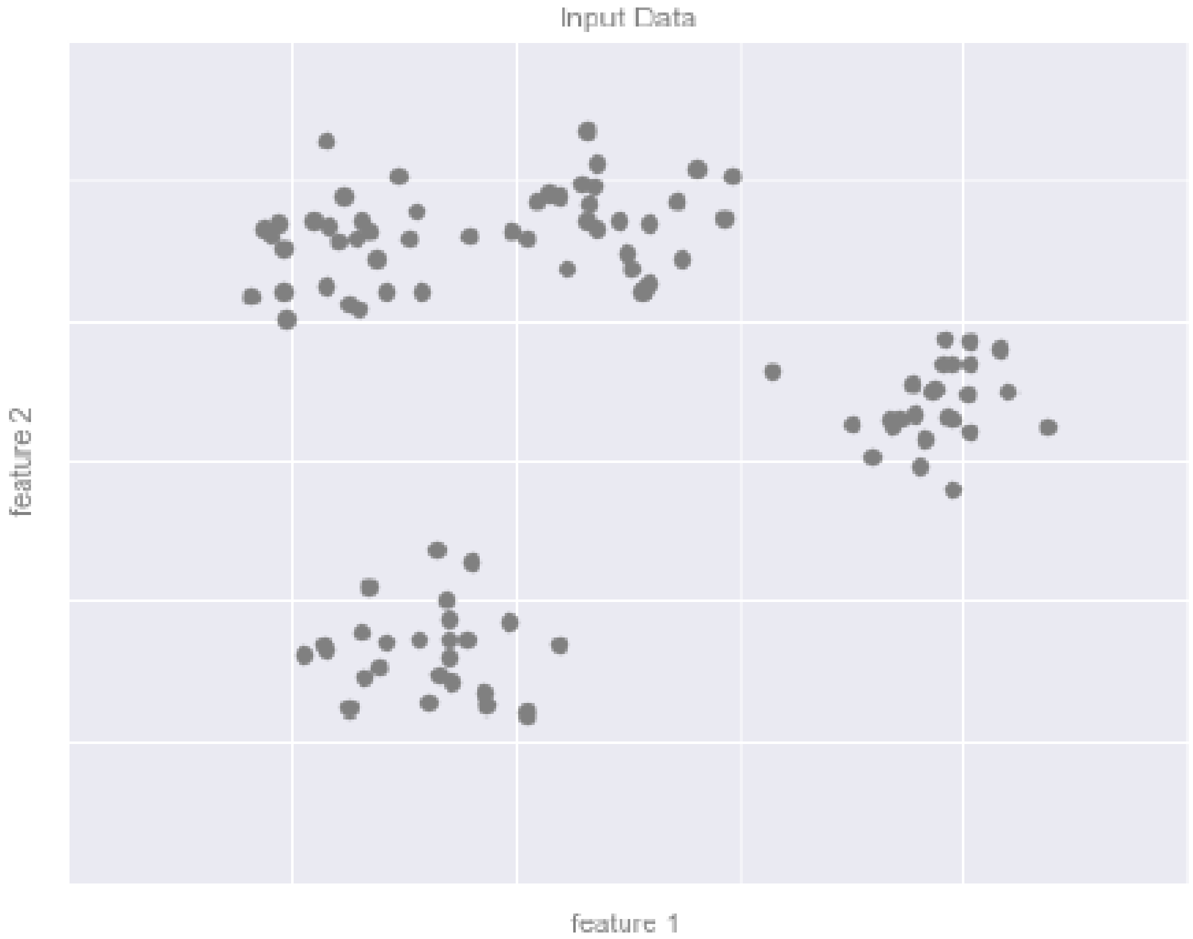
Clustering:

Inferring labels on unlabeled data By eye, it is clear that each of these points is part of a distinct group.

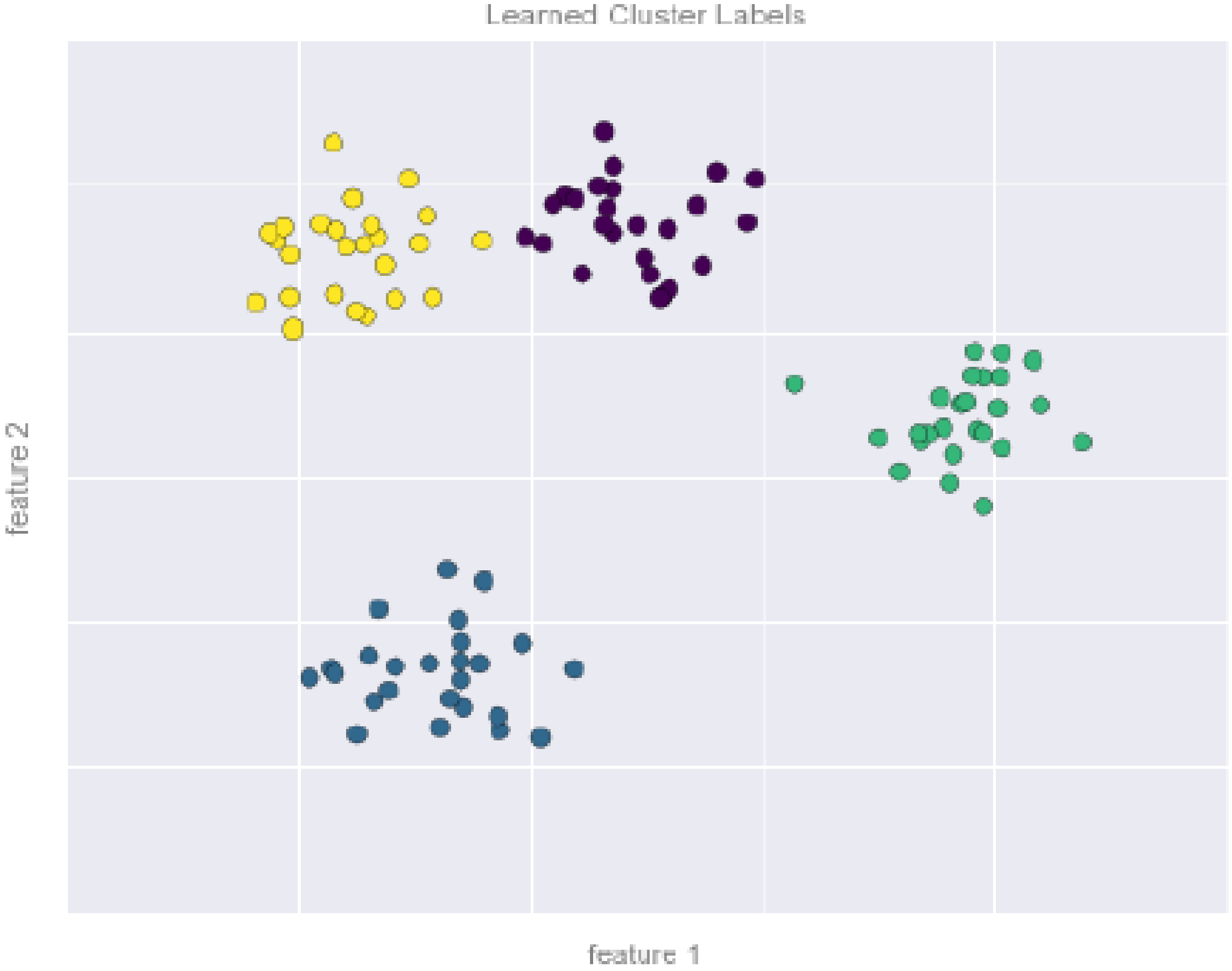
Given this input, **a clustering model** will use the intrinsic structure of the data to determine which points are related ; using the very fast and simple ***k*-means algorithm.**

***k*-means fits a model** consisting of ***k* cluster centers**; the optimal centers are assumed to be those that **minimize the distance** of each point from its **assigned center**.

In the figure **, event logs** are classified according to their **features**, forming clusters.



After clustering, we get,



D. **Structured annotation**:

This usually involves performing some analysis on input data points and adding structured metadata as annotations to the original data that depict extra information and relationships among the data elements.

Simple examples would be annotating text with their parts of speech, named entities, grammar, and sentiment.

Annotations can also be done for images like assigning specific categories to image pixels, annotate specific areas of images based on their type, location, and so on.

E. **Translation**:

Automated machine translation tasks are typically of the nature such that if you have input data samples belonging to a specific language, you translate it into output having another desired language.

Natural language based translation is definitely a huge area dealing with a lot of text data.

Clustering or grouping:

- Clusters or groups are usually formed from input data samples by making the machine learn or observe inherent latent patterns, relationships and similarities among the input data points themselves.

- Usually there is a lack of pre-labeled or pre-annotated data for these tasks hence they form a part of unsupervised Machine Learning (which we will discuss later on).

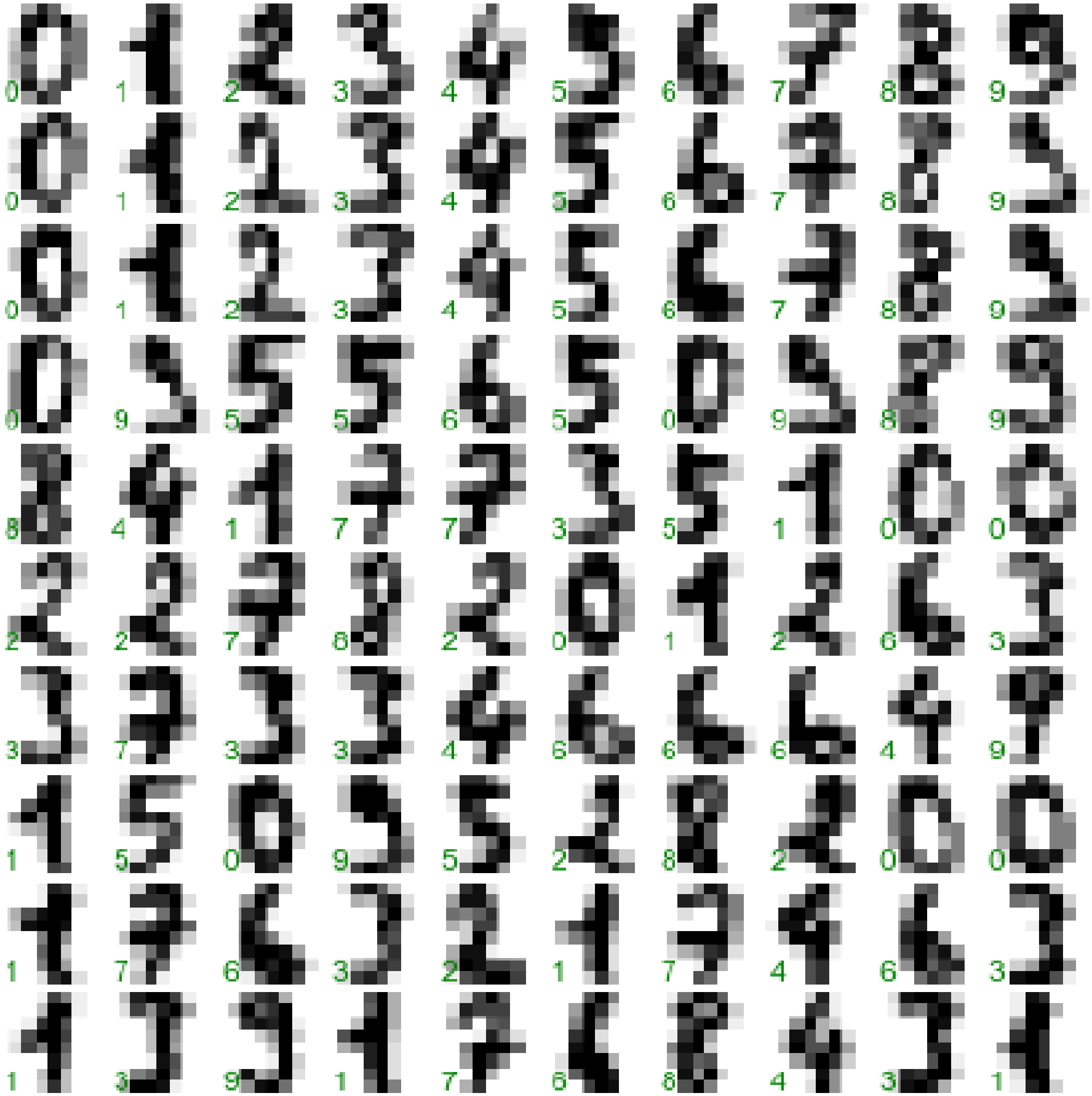
- Examples would be grouping similar products, events and entities.

G. Transcriptions:

These tasks usually entail various representations of data that are usually continuous and unstructured and converting them into more structured and discrete data elements.

Examples include speech to text, optical character recognition, images to text, and so on.

Example of **Transcriptions: images to text** The handwritten digits data; each sample is represented by one **8×8 grid of pixels**



Defining the Experience, E

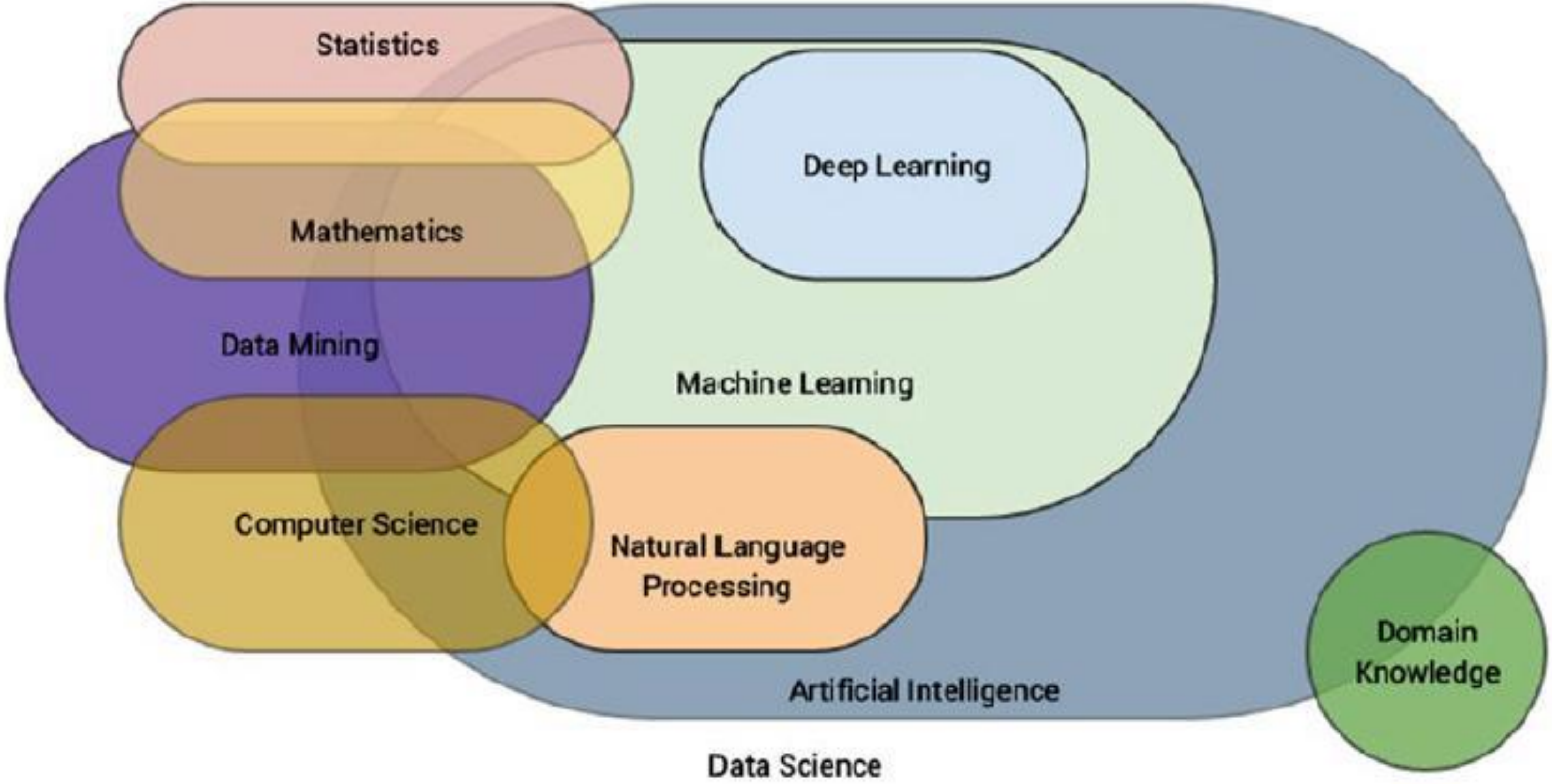
- At this point, you know that any learning algorithm typically needs data to learn over time and perform a specific task, which we named as T. The process of consuming a dataset that ~~consists of data samples or data points such that a learning algorithm or model learns~~ inherent patterns is defined as the experience, *E* which is gained by the learning algorithm.
- Any experience that the algorithm gains is from data samples or data points and this can be at any point of time. You can feed it data samples in one go using historical data or even supply fresh data samples whenever they are acquired.
- Thus, the idea of a model or algorithm gaining experience usually occurs as an iterative process, also known as training the model. You could think of the model to be an entity just like a human being which gains knowledge or experience through data points by observing and learning more and more about various attributes, relationships and patterns present in the data. Of course, there are various forms and ways of learning and gaining experience including supervised, unsupervised, and reinforcement learning

Defining the Performance, P

- Let’s say we have a Machine Learning algorithm that is supposed to perform a task, T, and is gaining experience, E, with data points over a period of time.
- **But how do we know if it’s performing well or behaving the way it is supposed to behave?**
- This is where the performance, *P,* of the model comes into the picture. The performance, P, is usually a quantitative measure or metric that’s used to see how well the algorithm or model is performing the task, T, with experience, E.
- Typical performance measures include **accuracy, precision, recall, F1 score, sensitivity, specificity, error rate, misclassification rate**, and many more. Performance measures are usually evaluated on training data samples (used by the algorithm to gain experience, E) as well as data samples which it has not seen or learned from before, which are usually known as validation and test data samples.
- The idea behind this is to **generalize the algorithm** so that it **doesn’t become too biased only on the training data points** and **performs well in the future on newer data points**.

- A simple example would be that sometimes we would want to penalize misclassification or false positives more than correct hits or predictions. In such a scenario, we might need to use a modified cost function or priors such that we give a scope to sacrifice hit rate or overall accuracy for more accurate predictions with lesser false positives.
- A real-world example would be an intelligent system that predicts if we should give a loan to a customer. It’s better to build the system in such a way that it is more cautious against giving a loan than denying one.
- The simple reason is because one big mistake of giving a loan to a potential defaulter can lead to huge losses as compared to denying several smaller loans to potential customers.
- To conclude, you need to take into account all parameters and attributes involved in task, T, such that you can decide on the right performance measures, P, for your system.

Machine learning is a Multi-Disciplinary Field



Important Concepts

In this section, we discuss some key terms and concepts from applied mathematics, namely linear algebra and probability theory.

These concepts are widely used across Machine Learning and form some of the foundational structures and principles across Machine Learning algorithms, models, and processes.

- **Scalar**

A *scalar* usually denotes a single number as opposed to a collection of numbers.

A simple example might be

$x = 5$ or $x \in R$, where x is the scalar element pointing to a single number or a real-valued single number.

- **Vector**

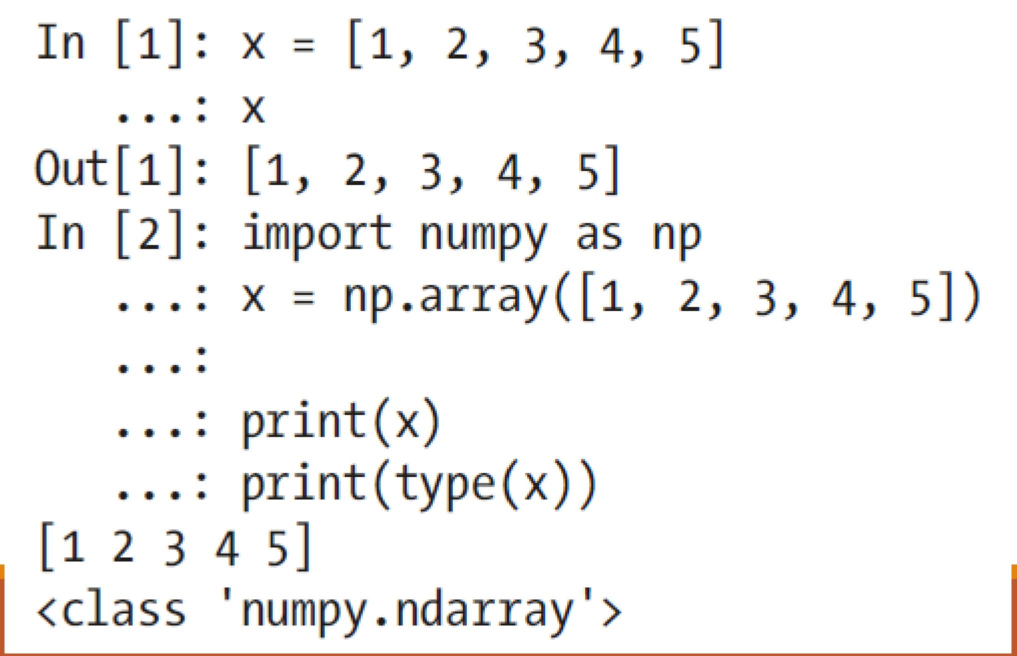
A *vector* is defined as a structure that holds an array of numbers which are arranged in order.

This basically means the order or sequence of numbers in the collection is important.

Vectors can be mathematically denoted as $x = [x1, x2, \ldots, xn]$, which basically tells us that x is a one-dimensional vector having n elements in the array.

Each element can be referred to using an array index determining its position in the vector.

The following snippet shows us how we can represent simple vectors in Python.



```
In [1]: x = [1, 2, 3, 4, 5]
   ...: x
Out[1]: [1, 2, 3, 4, 5]
In [2]: import numpy as np
   ...: x = np.array([1, 2, 3, 4, 5])
   ...:
   ...: print(x)
   ...: print(type(x))
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

Thus you can see that Python lists as well as numpy based arrays can be used to represent vectors.

Each row in a dataset can act as a one-dimensional vector of n attributes, which can serve as inputs to learning algorithms.

- **Matrix**

A *matrix* is a two-dimensional structure that basically holds numbers.

It’s also often referred to as a 2D array.

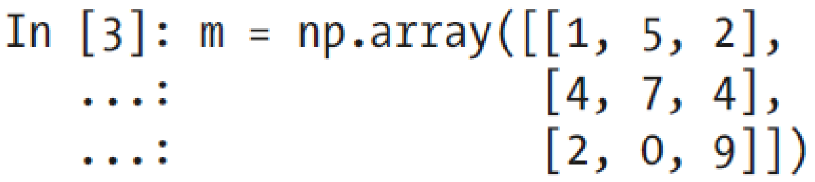
Each element can be referred to using a row and column index as compared to a single vector index in case of vectors.

Mathematically, you can depict a matrix as

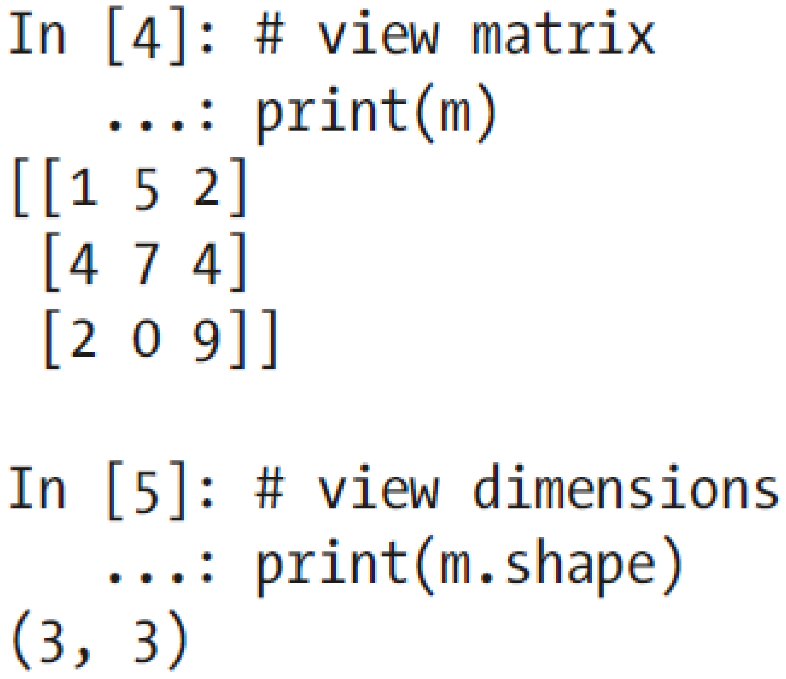
$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

such that *M* is a 3 x 3 matrix having three rows and three columns and each element is denoted by *mrc* such that *r* denotes the row index and *c* denotes the column index.

Matrices can be easily represented as list of lists in Python and we can leverage the numpy array structure as depicted in the following snippet.



```
In [3]: m = np.array([[1, 5, 2],
   ...:               [4, 7, 4],
   ...:               [2, 0, 9]])
```



```
In [4]: # view matrix
   ...: print(m)
[[1 5 2]
 [4 7 4]
 [2 0 9]]

In [5]: # view dimensions
   ...: print(m.shape)
(3, 3)
```

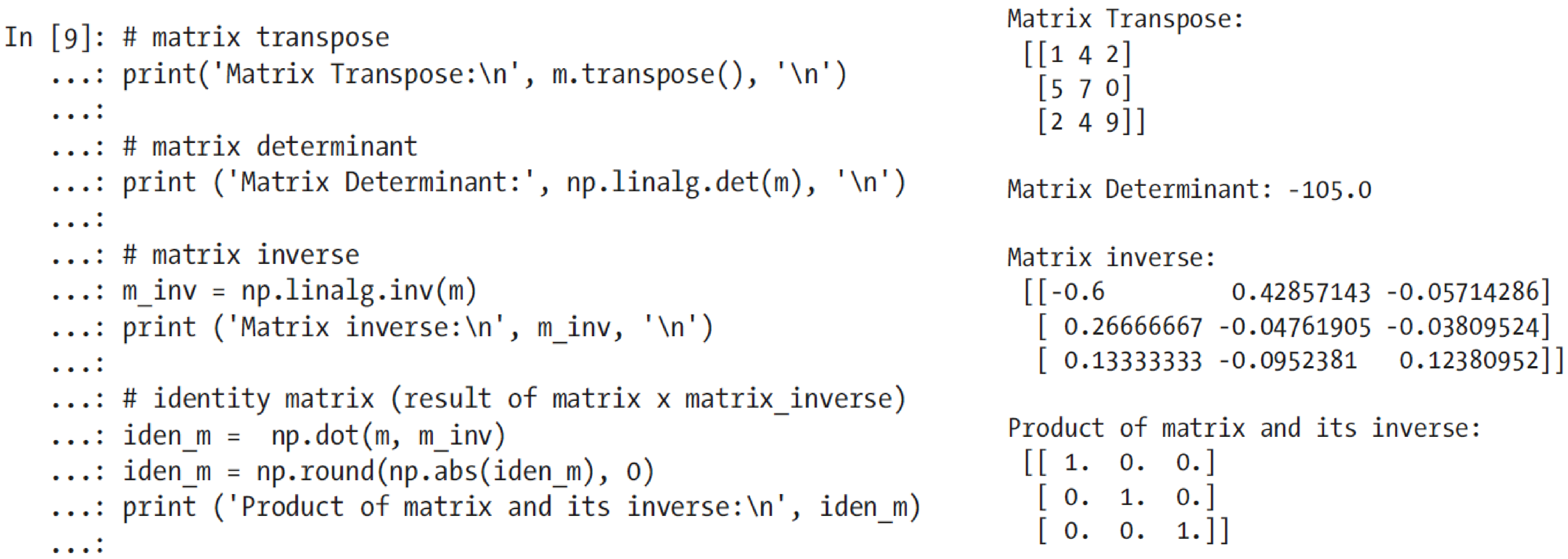
Thus you can see how we can easily leverage numpy arrays to represent matrices.

You can think of a dataset with rows and columns as a matrix such that the data features or attributes are represented by columns and each row denotes a data sample.

We will be using the same analogy later on in our analyses.

Of course, you can perform matrix operations like add, subtract, products, inverse, transpose, determinants, and many more.

The following snippet shows some popular matrix operations.



```
In [9]: # matrix transpose
   ...: print('Matrix Transpose:\n', m.transpose(), '\n')
   ...:
   ...: # matrix determinant
   ...: print ('Matrix Determinant:', np.linalg.det(m), '\n')
   ...:
   ...: # matrix inverse
   ...: m_inv = np.linalg.inv(m)
   ...: print ('Matrix inverse:\n', m_inv, '\n')
   ...:
   ...: # identity matrix (result of matrix x matrix_inverse)
   ...: iden_m =  np.dot(m, m_inv)
   ...: iden_m = np.round(np.abs(iden_m), 0)
   ...: print ('Product of matrix and its inverse:\n', iden_m)
   ...:
```

```
Matrix Transpose:
 [[1 4 2]
  [5 7 0]
  [2 4 9]]

Matrix Determinant: -105.0

Matrix inverse:
 [[-0.6         0.42857143 -0.05714286]
  [ 0.26666667 -0.04761905 -0.03809524]
  [ 0.13333333 -0.0952381   0.12380952]]

Product of matrix and its inverse:
 [[ 1.  0.  0.]
  [ 0.  1.  0.]
  [ 0.  0.  1.]]
```

Tensor

You can think of a *tensor* as a generic array.

Tensors are basically arrays with a variable number of axes.

~~An element in a three-dimensional tensor *T* can be denoted by *Tx,y,z* where *x, y, z* denote the three axes for~~ specifying element T.

Norm

The *norm* is a measure that is used to compute the size of a vector often also defined as the measure of distance from the origin to the point denoted by the vector.

Mathematically, the *p*th norm of a vector is denoted as follows.

$$L^p = \left\|x_p\right\| = \left(\sum_i \left|x_i\right|^p\right)^{\frac{1}{p}}$$

such that *p* ≥ 1 *and p* ∈ *R*. Popular norms in Machine Learning include the *L1* norm used extensively in Lasso regression models and the *L2* norm, also known as the Euclidean norm, used in ridge regression models.

- **Eigen Decomposition**

This is basically a matrix decomposition process such that we decompose or break down a matrix into a set of eigen vectors and eigen values.

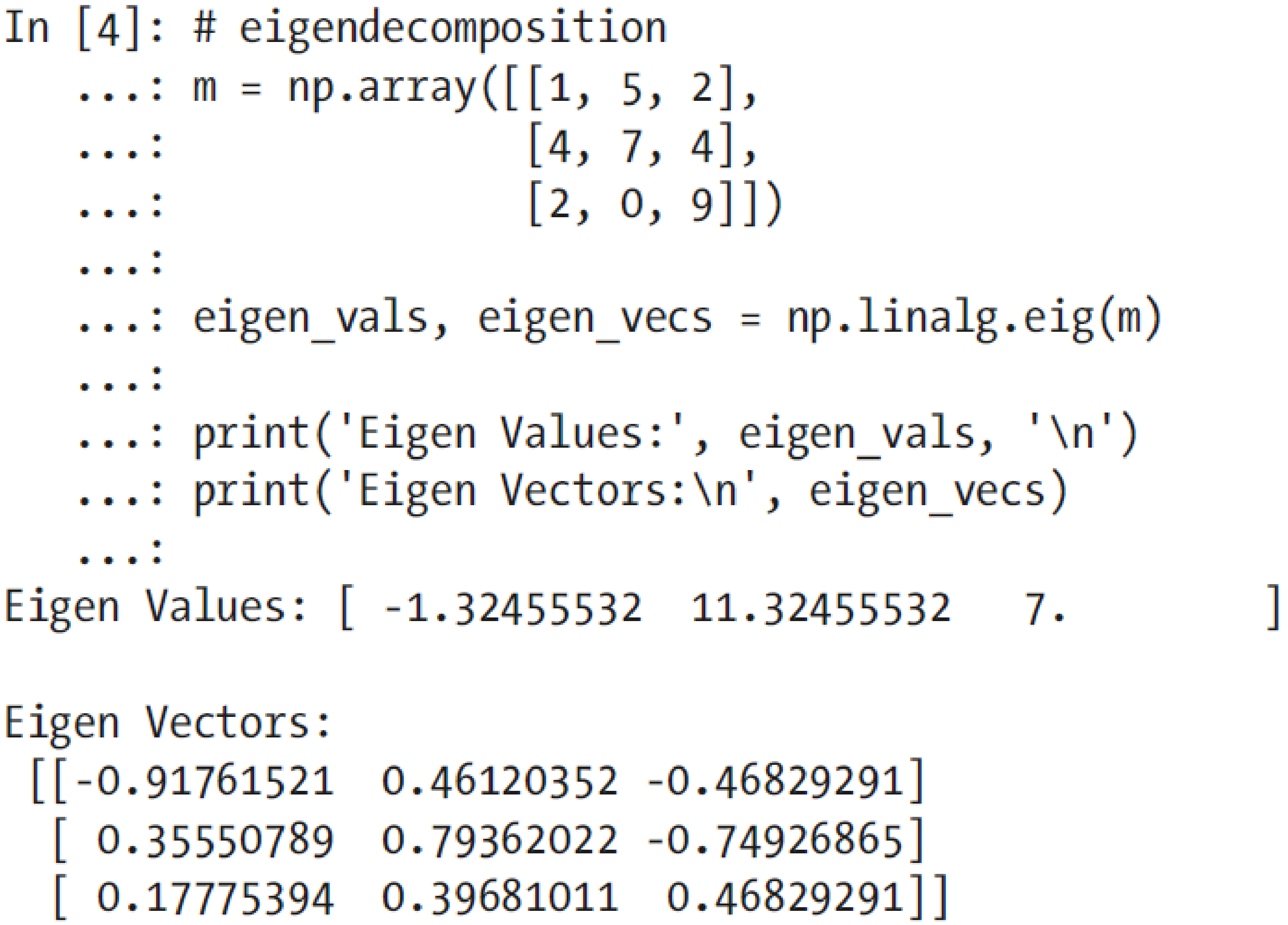
The most used type of matrix decomposition is the **eigendecomposition** that decomposes a matrix into ***eigenvectors*** and ***eigenvalues***.

This decomposition also plays a role in methods used in machine learning, such as in the **Principal Component Analysis** method or PCA.

To know about Eigenvalues and Eigenvectors:

https://machinelearningmastery.com/introduction-to-eigendecomposition-eigenvalues-and-eigenvectors/

The following Python snippet depicts how to extract eigen values and eigen vectors from a matrix.



```
In [4]: # eigendecomposition
   ...: m = np.array([[1, 5, 2],
   ...:               [4, 7, 4],
   ...:               [2, 0, 9]])
   ...:
   ...: eigen_vals, eigen_vecs = np.linalg.eig(m)
   ...:
   ...: print('Eigen Values:', eigen_vals, '\n')
   ...: print('Eigen Vectors:\n', eigen_vecs)
   ...:
Eigen Values: [ -1.32455532  11.32455532   7.        ]

Eigen Vectors:
 [[-0.91761521  0.46120352 -0.46829291]
  [ 0.35550789  0.79362022 -0.74926865]
  [ 0.17775394  0.39681011  0.46829291]]
```

Singular Value Decomposition

The process of *singular value decomposition*, also known as **SVD**, is another matrix decomposition or factorization process such that we are able to break down a matrix to obtain singular vectors and singular values.

Any real matrix will always be decomposed by SVD even if eigen decomposition may not be applicable in some cases.

Mathematically, SVD can be defined as follows. Considering a matrix *M* having dimensions *m x n* such that *m* denotes total rows and *n* denotes total columns, the SVD of the matrix can be represented with the following equation

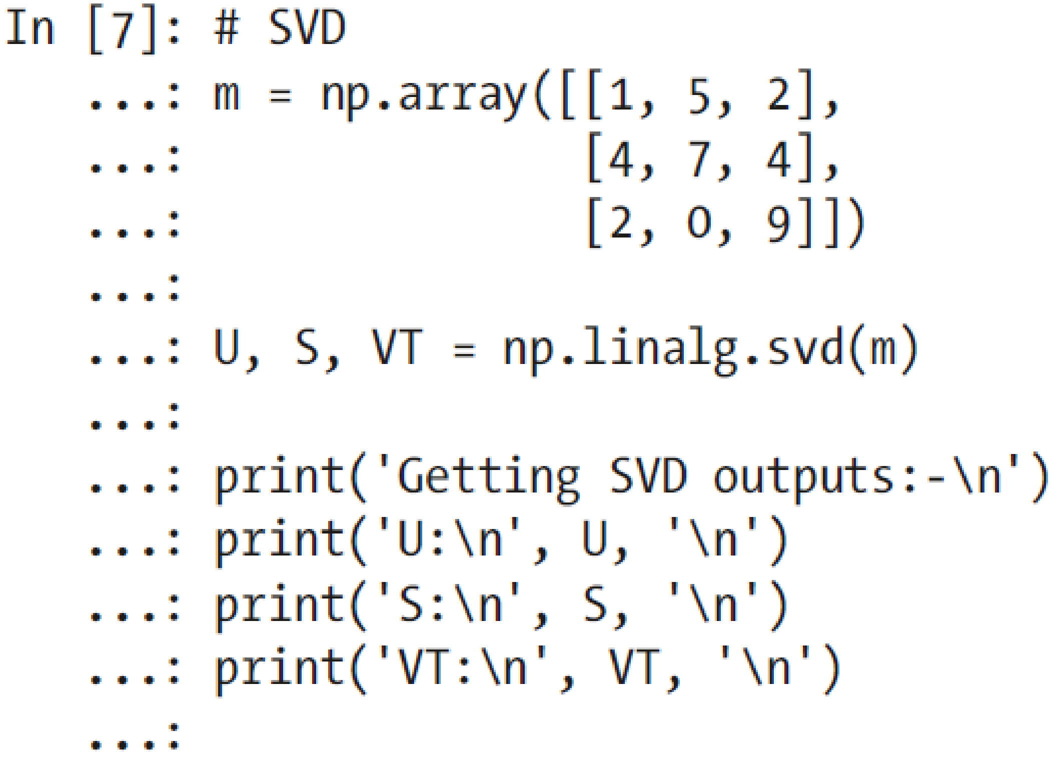
$$M_{m \times n} = U_{m \times m} S_{m \times n} V^{T}_{n \times n}$$

- This gives us the following main components of the decomposition equation.
  - *U$_{m \times m}$* is an *m* x *m* unitary matrix where each column represents a left singular vector
  - *S$_{m \times n}$* is an *m* x *n* matrix with positive numbers on the diagonal, which can also be represented as a vector of the singular values
  - *VT$_{n \times n}$* is an *n* x *n* unitary matrix where each row represents a right singular vector

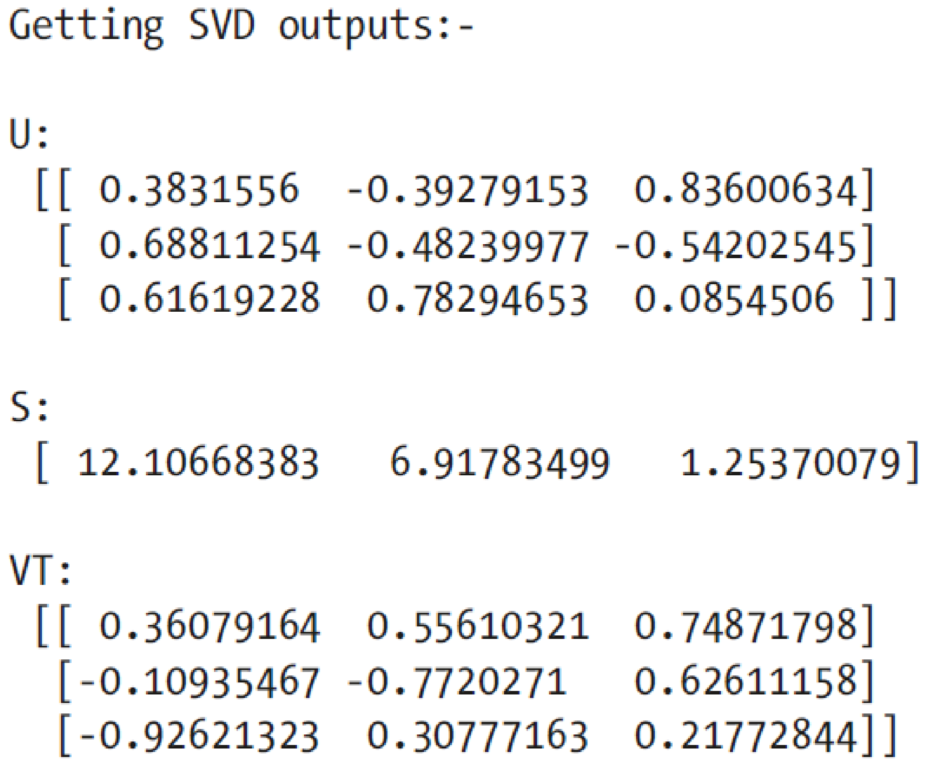
- In some representations, the rows and columns might be interchanged but the end result should be

the same, i.e., *U* and *V* are always orthogonal.

The following snippet shows a simple SVD decomposition in Python.



```
In [7]: # SVD
   ...: m = np.array([[1, 5, 2],
   ...:               [4, 7, 4],
   ...:               [2, 0, 9]])
   ...:
   ...: U, S, VT = np.linalg.svd(m)
   ...:
   ...: print('Getting SVD outputs:-\n')
   ...: print('U:\n', U, '\n')
   ...: print('S:\n', S, '\n')
   ...: print('VT:\n', VT, '\n')
   ...:
```



```
Getting SVD outputs:-

U:
 [[ 0.3831556  -0.39279153  0.83600634]
  [ 0.68811254 -0.48239977 -0.54202545]
  [ 0.61619228  0.78294653  0.0854506 ]]

S:
 [ 12.10668383   6.91783499   1.25370079]

VT:
 [[ 0.36079164  0.55610321  0.74871798]
  [-0.10935467 -0.7720271   0.62611158]
  [-0.92621323  0.30777163  0.21772844]]
```

SVD as a technique and the singular values in particular are very useful in summarization based algorithms and various other methods like dimensionality reduction.

More info: https://machinelearningmastery.com/singular-value-decomposition-for-machine-learning/

- **Random Variable**

Used frequently in probability and uncertainty measurement, a *random variable* is basically a variable that can take on various values at random.

These variables can be of discrete or continuous type in general.

- **Probability Distribution**

A *probability distribution* is a distribution or arrangement that depicts the likelihood of a random variable or variables to take on each of its probable states.

There are usually two main types of distributions based on the variable being discrete or continuous.

- **Conditional Probability**

The *conditional probability* rule is used when we want to determine the probability that an event is going to take place, such that another event has already taken place.

This is mathematically represented as follows.

$$P(x|\,y)=\frac{P(x,y)}{P(y)}$$

This tells us the conditional probability of *x,* given that *y* has already taken place.

Bayes Theorem

This is another rule or theorem which is useful when we know the probability of an event of interest ***P(A)***, the conditional probability for another event based on our ***event of interest P(B | A)*** and we want to determine the conditional probability of our event of interest given the other event has taken place ***P(A | B)***.

This can be defined mathematically using the following expression.

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$

Statistics

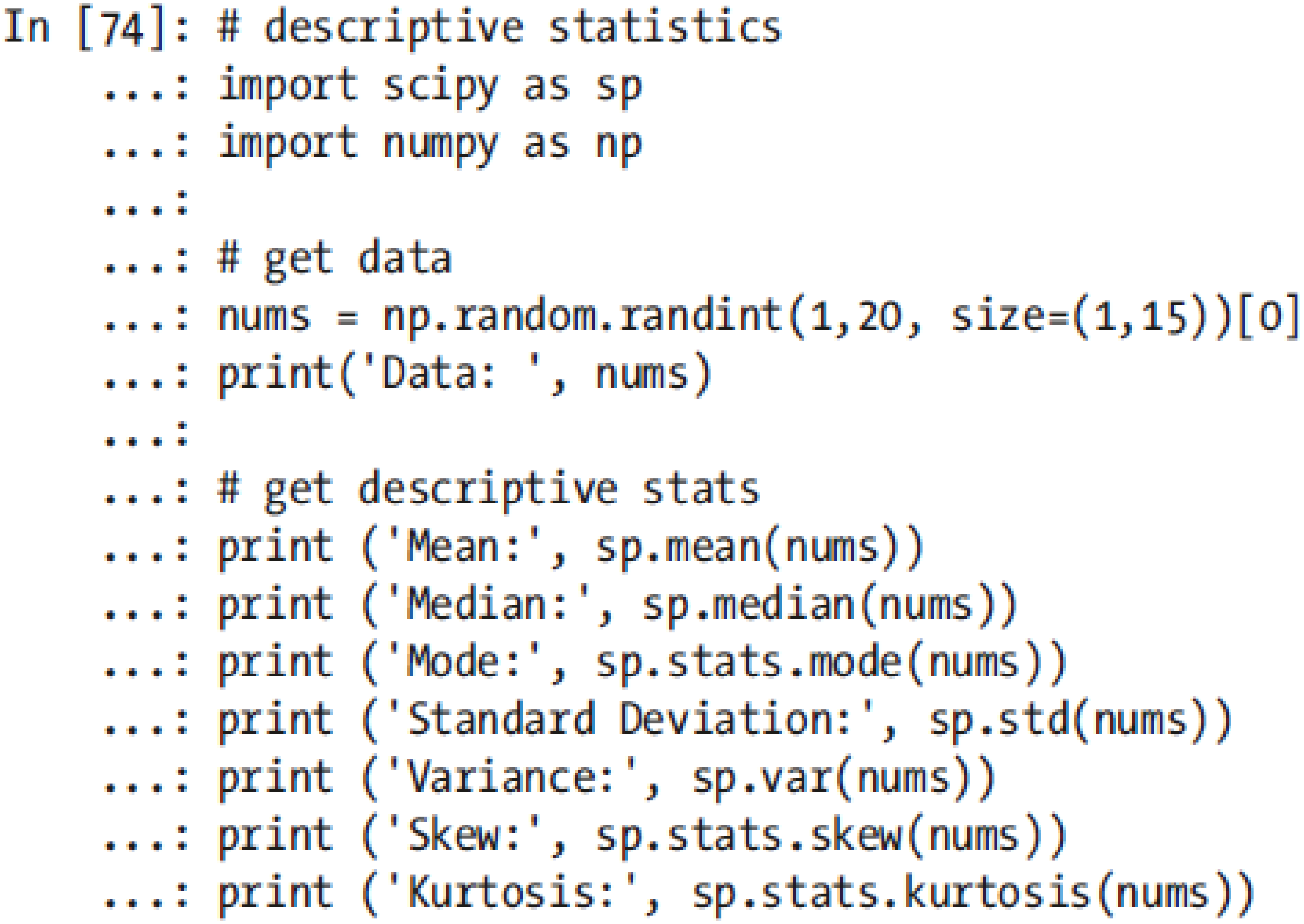
- The field of statistics can be defined as a specialized branch of mathematics that consists of frameworks and methodologies to collect, organize, analyze, interpret, and present data.

- Generally this falls more under applied mathematics and borrows concepts from linear algebra, distributions, probability theory, and inferential methodologies.

- There are two major areas under statistics that are mentioned as follows.
  a) **Descriptive statistics**
  b) **Inferential statistics**

Descriptive statistics is used to understand basic characteristics of the data using various aggregation and summarization measures to describe and understand the data better.

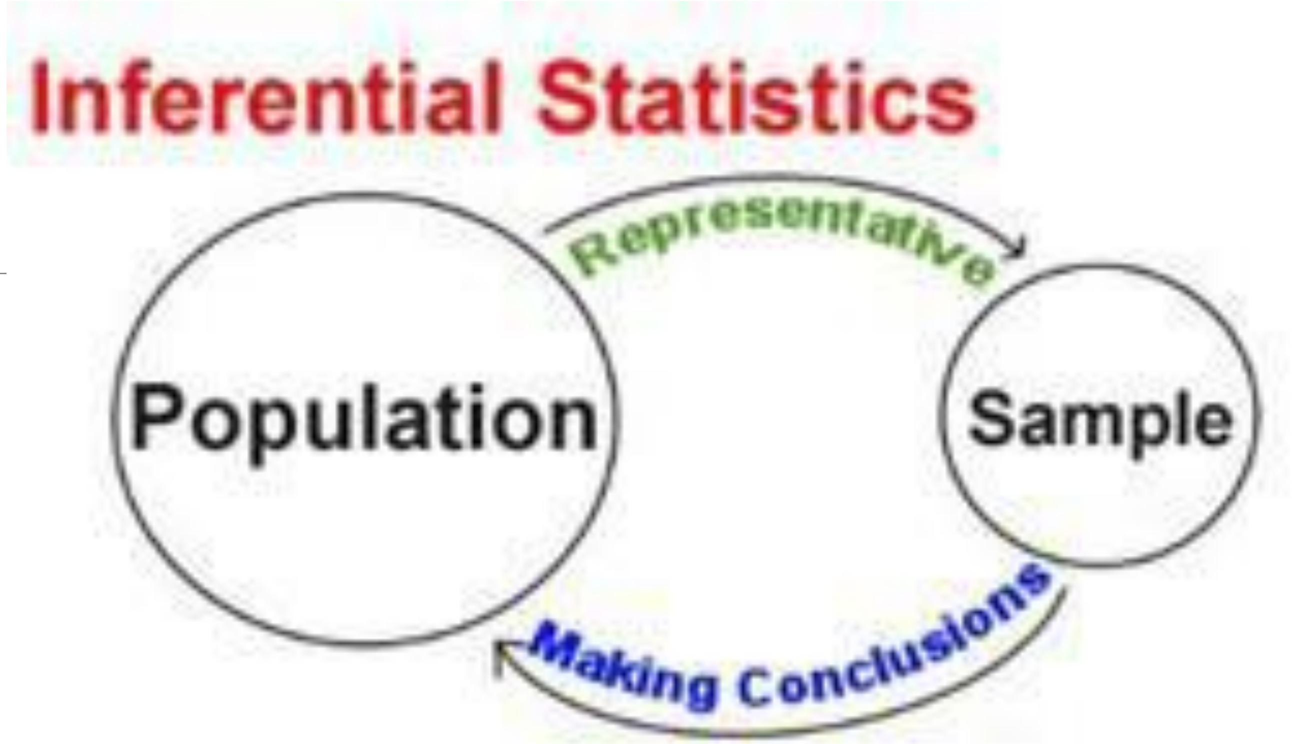
These could be standard measures like **mean, median, mode, skewness, kurtosis, standard deviation, variance**, and so on.



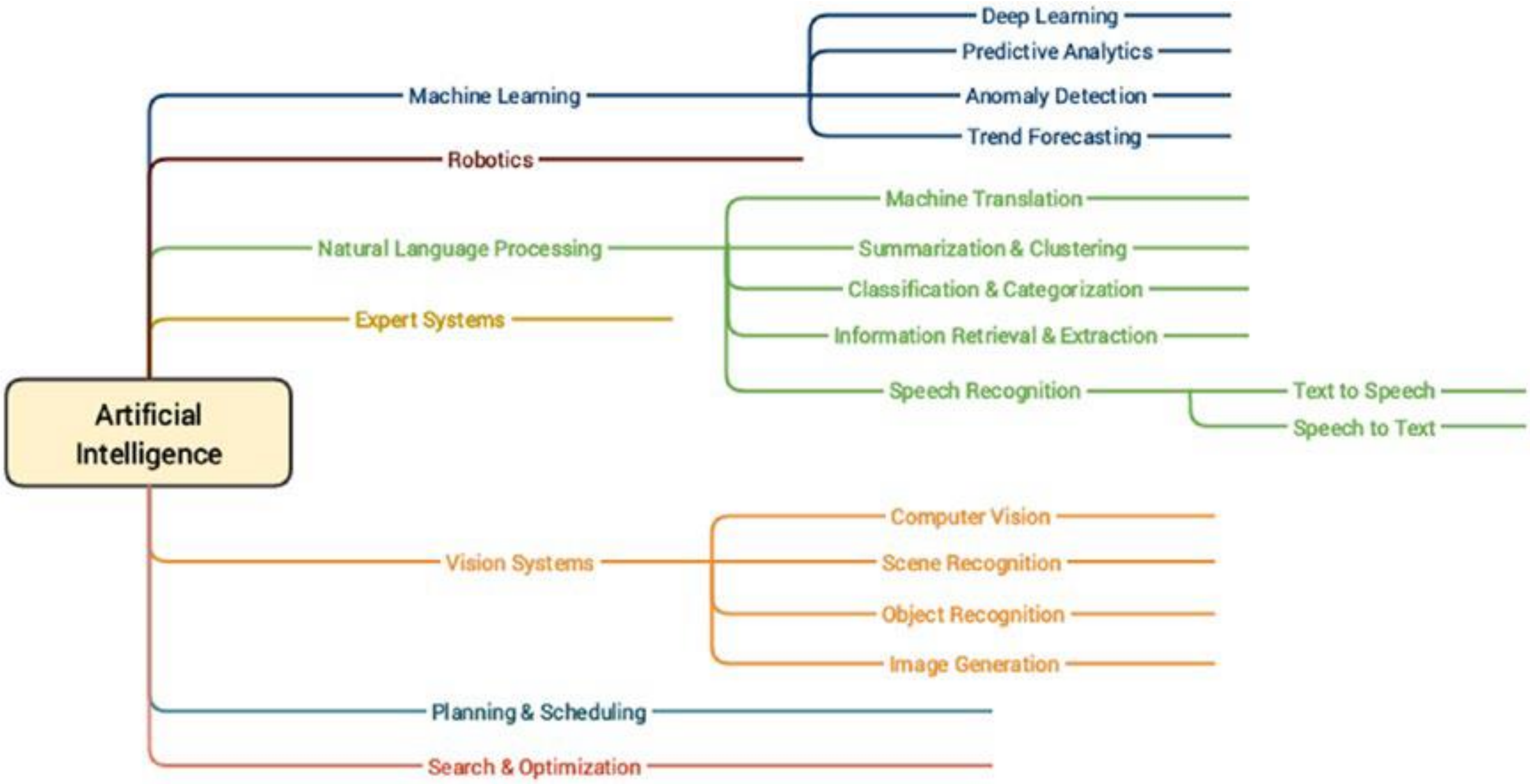
```
In [74]: # descriptive statistics
    ...: import scipy as sp
    ...: import numpy as np
    ...:
    ...: # get data
    ...: nums = np.random.randint(1,20, size=(1,15))[0]
    ...: print('Data: ', nums)
    ...:
    ...: # get descriptive stats
    ...: print ('Mean:', sp.mean(nums))
    ...: print ('Median:', sp.median(nums))
    ...: print ('Mode:', sp.stats.mode(nums))
    ...: print ('Standard Deviation:', sp.std(nums))
    ...: print ('Variance:', sp.var(nums))
    ...: print ('Skew:', sp.stats.skew(nums))
    ...: print ('Kurtosis:', sp.stats.kurtosis(nums))
```

Inferential statistics are used when we want to test hypothesis, draw inferences, and conclusions about various characteristics of our data sample or population.

Frameworks and techniques like hypothesis testing, correlation, and regression analysis, forecasting, and predictions are typically used for any form of inferential statistics.



Diverse major facets under the AI umbrella



Machine Learning Methods

- Machine Learning has multiple algorithms, techniques, and methodologies that can be used to build models to solve real-world problems using data.

- This section tries to classify these Machine Learning methods under some broad categories to give some sense to the overall landscape of Machine Learning methods that are ultimately used to perform specific **Machine Learning tasks** *we discussed previously*.

- Typically the same Machine Learning methods can be classified in multiple ways under multiple umbrellas.

Following are some of the major broad areas of Machine Learning methods.

1. Methods based on the amount of **human supervision** in the learning process
   - a. Supervised learning
   - b. Unsupervised learning
   - c. Semi-supervised learning
   - d. Reinforcement learning

2. Methods based on the ability to **learn from incremental data samples**
   - a. Batch learning
   - b. Online learning

3. Methods based on their approach to **generalization from data samples**
   - a. Instance based learning
   - b. Model based learning

Supervised Learning

- Supervised learning methods or algorithms include learning algorithms that take in data samples (known as training data) and associated outputs (known as labels or responses) with each data sample during the model training process. The main objective is to learn a mapping or association between input data samples x and their corresponding outputs y based on multiple training data instances.

- This learned knowledge can then be used in the future to predict an output y′ for any new input data sample x′ which was previously unknown or unseen during the model training process.

- These methods are termed as supervised because the model learns on data samples where the desired output responses/labels are already known beforehand in the training phase.

- Supervised learning basically tries to model the relationship between the inputs and their corresponding outputs from the training data so that we would be able to predict output responses for new data inputs based on the knowledge it gained.

- Supervised learning methods are extensively used in predictive analytics where the main objective is to predict some response for some input data that’s typically fed into a trained supervised ML model.

- Supervised learning methods are of two major classes based on the type of ML tasks they aim to solve.
  1. Classification
  2. Regression

---

1. Classification

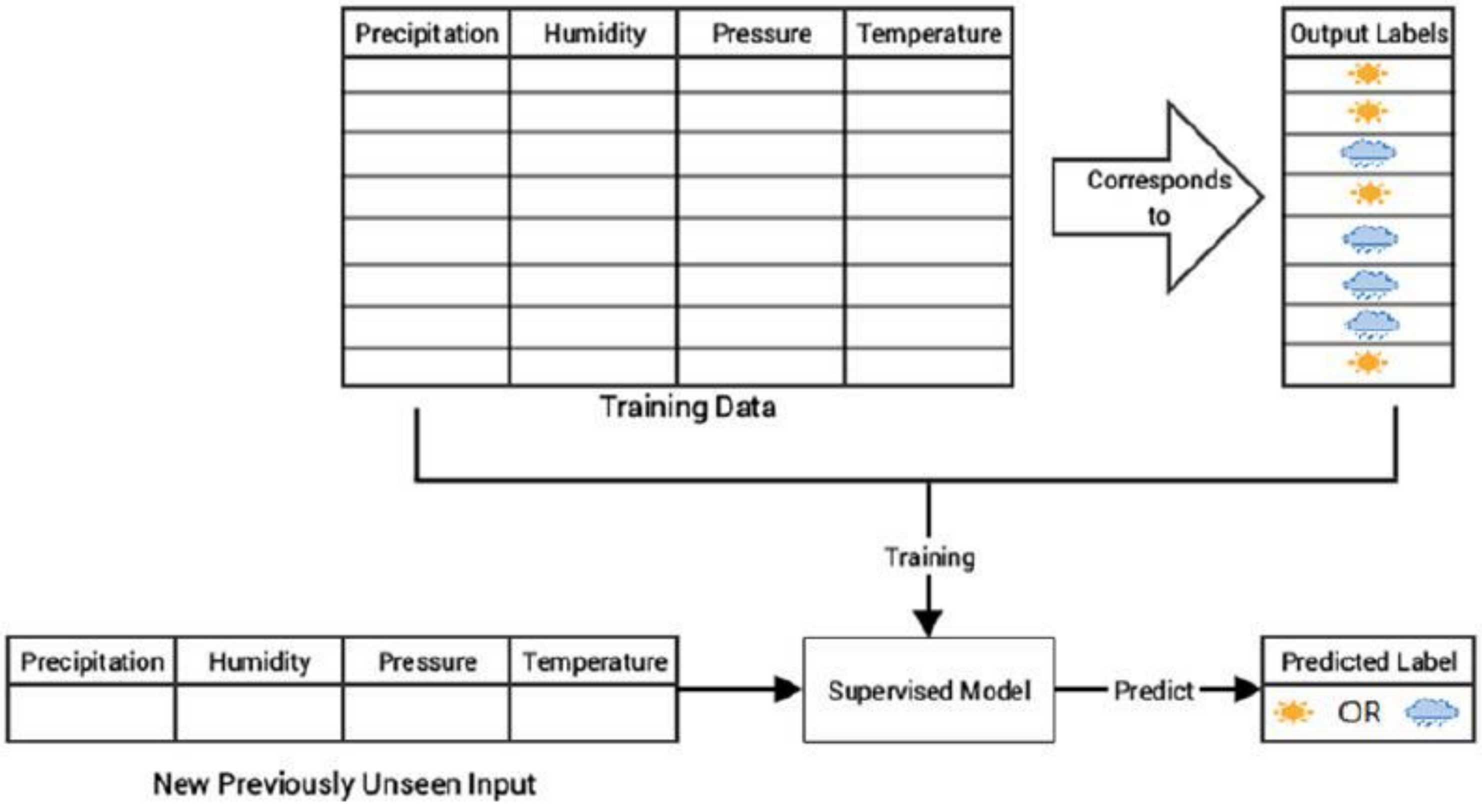
Here, the key objective is to predict output labels or responses that are categorical in nature for input data based on what the model has learned in the training phase.

Output labels here are also known as classes or class labels are these are categorical in nature meaning they are unordered and discrete values.

Thus, each output response belongs to a specific discrete class or category.

Suppose we take a real-world example of predicting the weather. Let’s keep it simple and say we are trying to predict if the weather is sunny or rainy based on multiple input data samples consisting of attributes or features like humidity, temperature, pressure, and precipitation. Since the prediction can be either sunny or rainy, there are a total of two distinct classes ; hence this problem can also be termed as a ***binary classification problem***.

Here the **supervised model** has, **feature vectors**, (precipitation, humidity, pressure, and temperature) for **each data sample/observation** and their corresponding **class labels** as either **sunny** or **rainy**.



Binary classification problem

- A task where the total number of distinct classes is more than two becomes a multi-class classification problem where each prediction response can be any one of the probable classes from this set.

- A simple example would be trying to predict numeric digits from scanned handwritten images. In this case it becomes a 10-class classification problem because the output class label for any image can be any digit from 0 - 9.

- Multi-label classification tasks are such that based on any input data sample, the output response is usually a vector having one or more than one output class label.

- A simple real-world problem would be trying to predict the category of a news article that could have multiple output classes like news, finance, politics, and so on.

II. Regression

- Machine Learning tasks where the main objective is value estimation can be termed as regression tasks.

- Regression based methods are trained on input data samples having output responses that are continuous numeric values unlike classification, where we have discrete categories or classes. Regression models make use of input data attributes or features (also called explanatory or independent variables) and their corresponding continuous numeric output values (also called as response, dependent, or outcome variable) to learn specific relationships and associations between the inputs and their corresponding outputs.

- With this knowledge, it can predict output responses for new, unseen data instances similar to classification but with continuous numeric outputs.

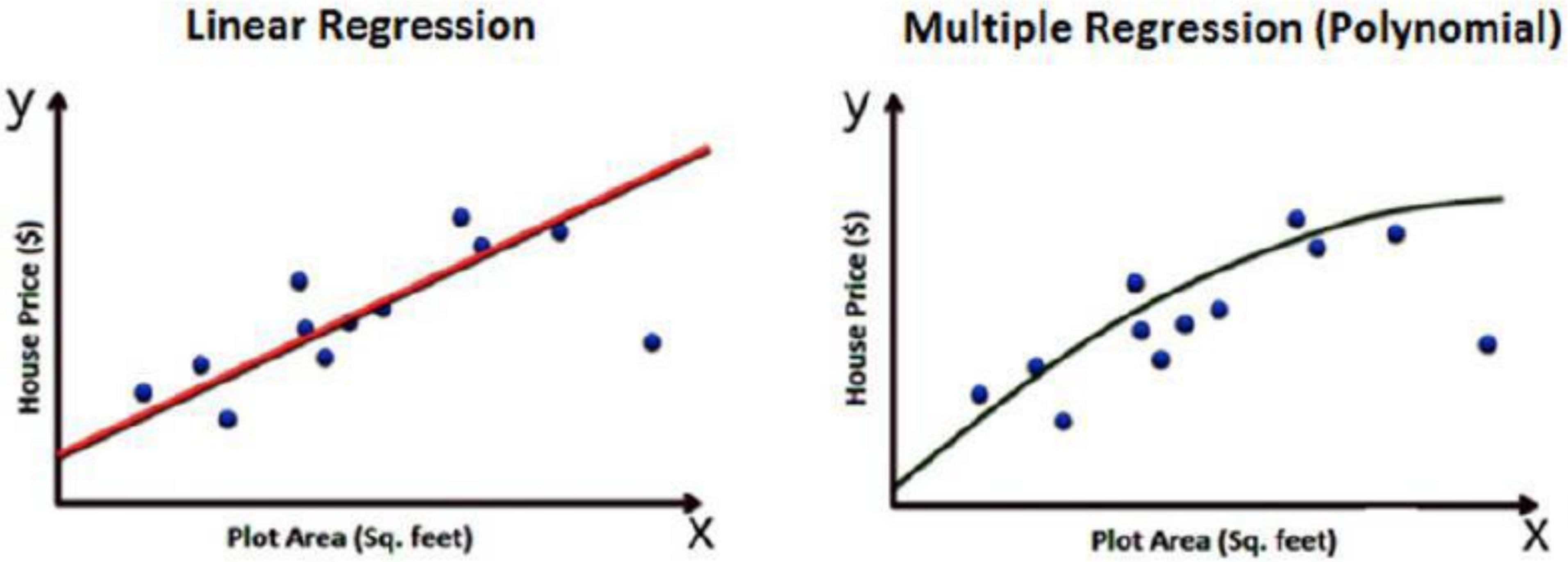
- **One of the most common real-world examples of regression is prediction of house prices. You can build a simple regression model to predict house prices based on data pertaining to land plot areas in square feet.**

- Figure below shows two possible regression models based on different methods to predict house prices based on plot area.

- **Linear Regression** models relationships on data with one feature variable ***x*** and a single response variable ***y.***

Multiple regression is also known as multivariable regression. These methods try to model data where we have one response output variable y in each observation but multiple explanatory variables in the form of a vector X instead of a single explanatory variable.

The idea is to predict *y* based on the different features present in *X*. **A real-world example would be extending our house prediction model** to build a more sophisticated model where we predict the house price based on multiple features instead of just plot area in each data sample.



Supervised learning: regression models for house price prediction

Unsupervised Learning

- Supervised learning methods usually require some training data where the outcomes which we are trying to predict are already available in the form of discrete labels or continuous values. However, often we do not have the liberty or advantage of having pre-labeled training data and we still want to extract useful insights or patterns from our data. In this scenario, unsupervised learning methods are extremely powerful. These methods are called unsupervised because the model or algorithm tries to learn inherent latent structures, patterns and relationships from given data without any help or supervision like providing annotations in the form of labeled outputs or outcomes.

- Unsupervised learning is more concerned with trying to extract meaningful insights or information from data rather than trying to predict some outcome based on previously available supervised training data. There is more uncertainty in the results of unsupervised learning but you can also gain a lot of information from these models that was previously unavailable to view just by looking at the raw data.

- Often unsupervised learning could be one of the tasks involved in building a huge intelligence system. For example, we could use unsupervised learning to get possible outcome labels for tweet sentiments by using the knowledge of the English vocabulary and then train a supervised model on similar data points and their outcomes which we obtained previously through unsupervised learning.

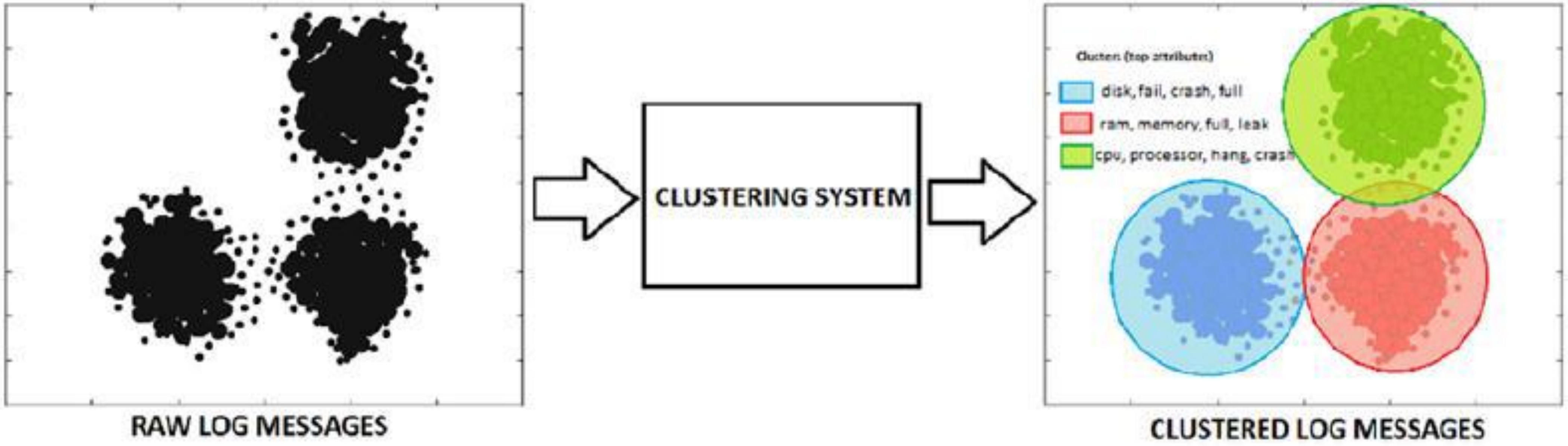
Unsupervised learning methods can be categorized under the following broad areas.

- Clustering
- Dimensionality reduction
- Anomaly detection
- Association rule-mining

Clustering

- Clustering methods are Machine Learning methods that try to find patterns of similarity and relationships among data samples in our dataset and then cluster these samples into various groups, such that each group or cluster of data samples has some similarity, based on the inherent attributes or features. These methods are completely unsupervised because they try to cluster data by looking at the data features without any prior training, supervision, or knowledge about data attributes, associations, and relationships.

- *Consider a real-world problem of running multiple servers in a data center and trying to analyze logs for typical issues or errors. Our main task is to determine the various kinds of log messages that usually occur frequently each week. In simple words, we want to group log messages into various clusters based on some inherent characteristics. A simple approach would be to extract features from the log messages, which would be in textual format and apply clustering on the same.*



Unsupervised learning: clustering log messages

It is quite clear from Figure that our systems have three distinct clusters of log messages where the first cluster depicts disk issues, the second cluster is about memory issues, and the third cluster is about processor issues. Top feature words that helped in distinguishing the clusters and grouping similar data samples (logs) together are also depicted in the figure. Of course, sometimes some features might be present across multiple data samples hence there can be slight overlap of clusters too since this is unsupervised learning. However, the main objective is always to create clusters such that elements of each cluster are near each other and far apart from elements of other clusters.

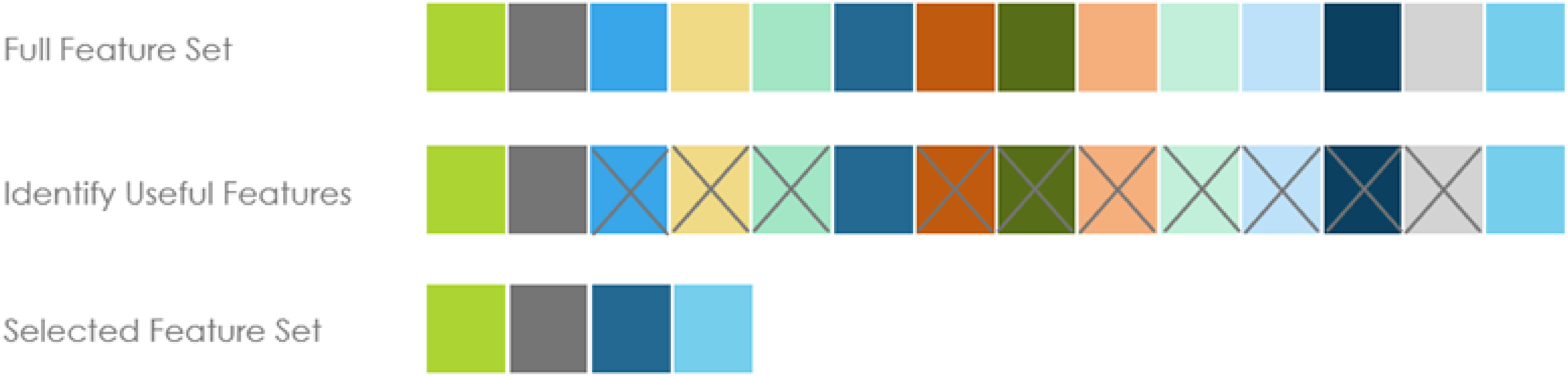
Dimensionality Reduction

- Once we start extracting attributes or features from raw data samples, sometimes our feature space gets bloated up with a humongous number of features. This poses multiple challenges including analyzing and visualizing data with thousands or millions of features, which makes the feature space extremely complex posing problems with regard to training models, memory, and space constraints. In fact this is referred to as the “curse of dimensionality”.

- Unsupervised methods can also be used in these scenarios, where we reduce the number of features or attributes for each data sample. These methods reduce the number of feature variables by extracting or selecting a set of principal or representative features.

- There are multiple popular algorithms available for dimensionality reduction like

  1. Principal Component Analysis (PCA),
  2. Nearest neighbors, and
  3. Discriminant analysis.

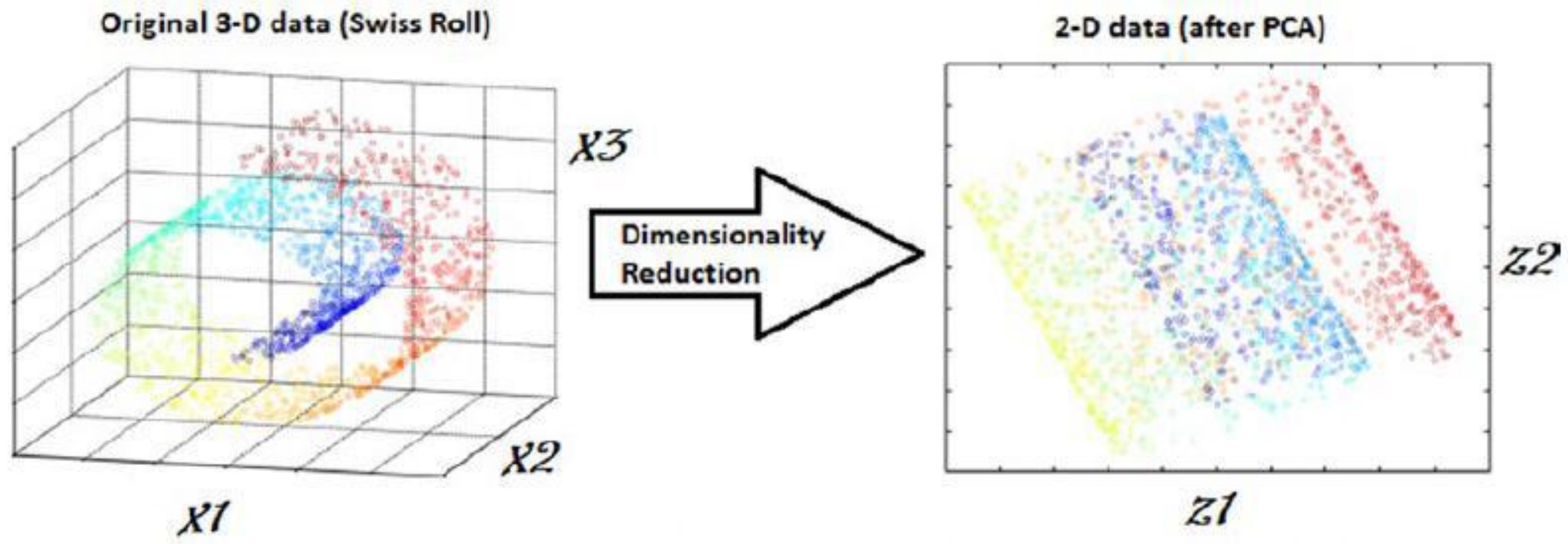


Full Feature Set

Identify Useful Features

Selected Feature Set

Figure below shows the output of a typical feature reduction process applied to a Swiss Roll 3D structure having three dimensions to obtain a two-dimensional feature space for each data sample using PCA.



It is clear that each data sample originally had three features or dimensions, namely D(x1, x2, x3) and after applying PCA, we reduce each data sample from our dataset into two dimensions, namely D’(z1, z2). Dimensionality reduction techniques can be classified in two major approaches as follows.

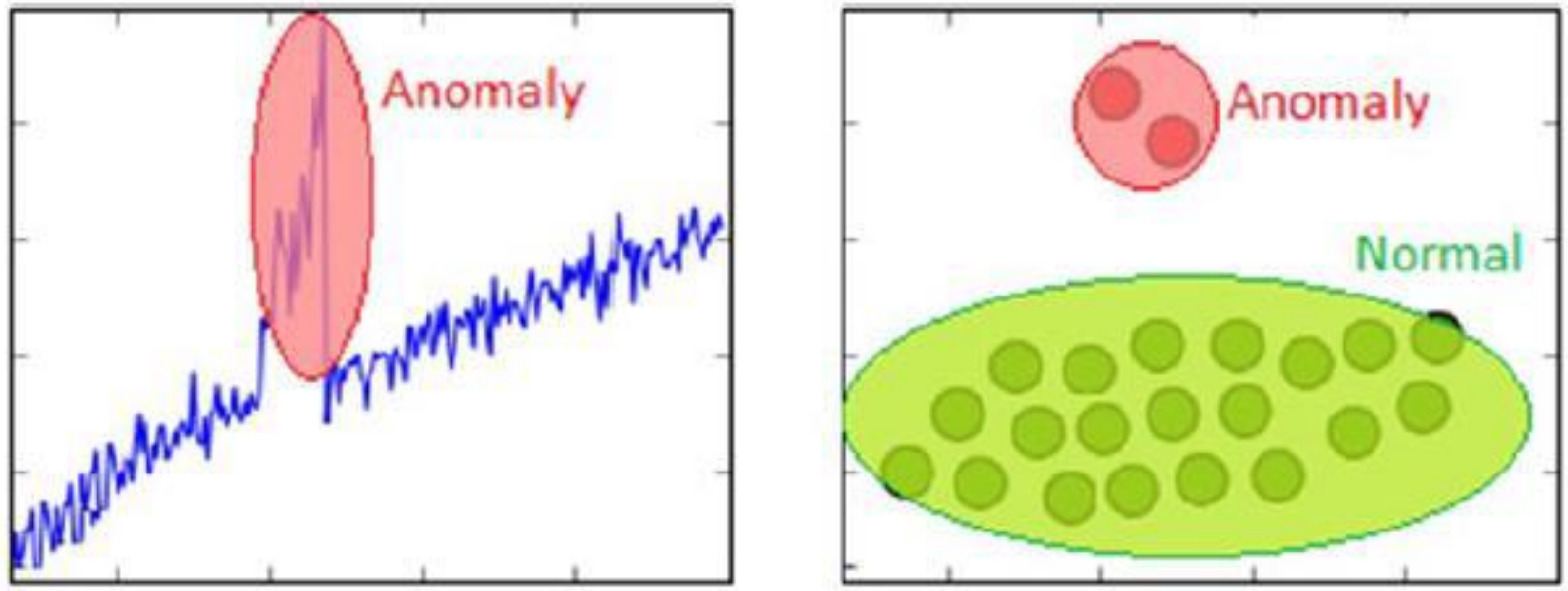
1. **Feature Selection methods:** Specific features are selected for each data sample from the original list of features and other features are discarded. No new features are generated in this process.

2. **Feature Extraction methods:** We engineer or extract new features from the original list of ~~features in the data. Thus the reduced subset of features will contain~~ newly generated features that were not part of the original feature set. PCA falls under this category.

3. **Anomaly Detection:** The process of anomaly detection is also termed as outlier detection, where we are interested in finding out occurrences of rare events or observations that typically do not occur normally based on historical data samples. Sometimes anomalies occur infrequently and are thus rare events, and in other instances, anomalies might not be rare but might occur in very short bursts over time.

4. **Unsupervised learning methods** can be used for anomaly detection such that we train the algorithm on the training dataset having normal, non-anomalous data samples. Once it learns the necessary data representations, patterns, and relations among attributes in normal samples, for any new data sample, it would be able to identify it as **anomalous or a normal data point** by using its learned knowledge. cific patterns.

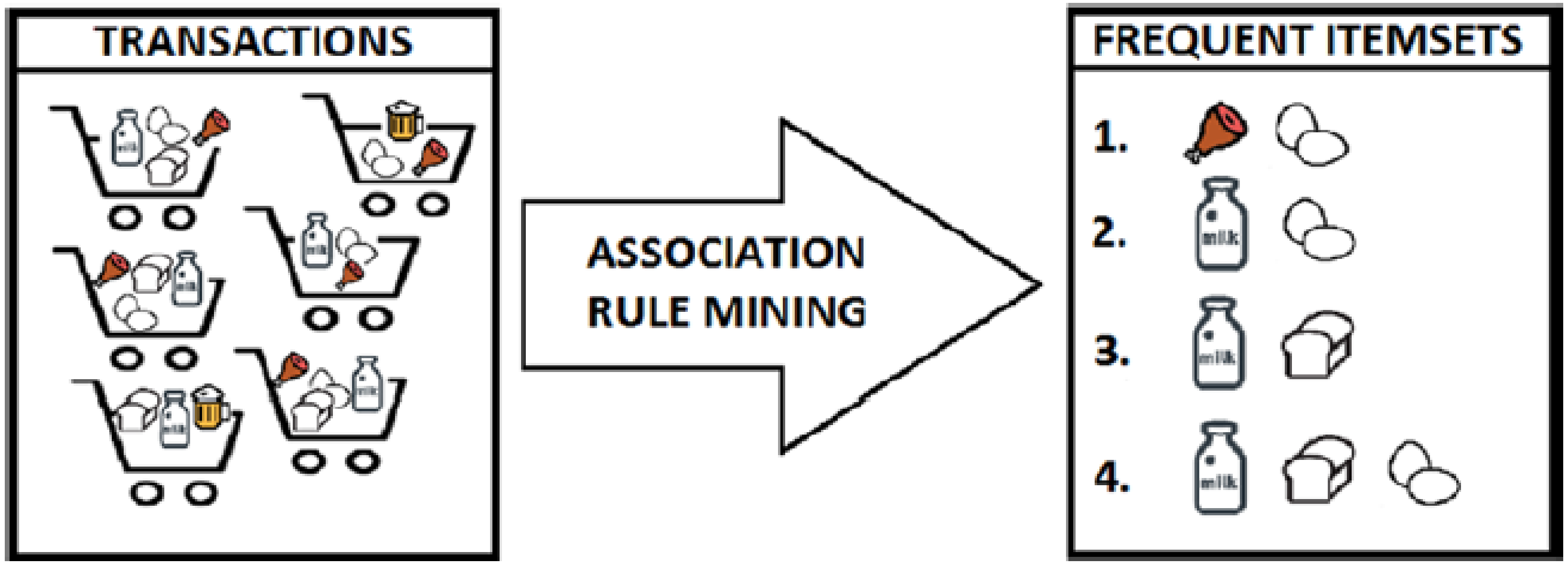
Figure depicts some typical anomaly detection based scenarios where you could apply supervised methods like one-class SVM and unsupervised methods like clustering, K-nearest neighbors, and so on to detect anomalies based on data and its features.



Anomaly detection based methods are extremely popular in real-world scenarios like detection of **security attacks or breaches, credit card fraud, manufacturing anomalies, network issues**, and many more.

Association Rule-Mining

- Typically association rule-mining is a data mining method use to examine and analyze large transactional datasets to find patterns and rules of interest.
- **Association rule-mining** is also often termed as ***market basket analysis**,* which is used to **analyze customer shopping patterns**.
- Association rules help in detecting and predicting transactional patterns based on the knowledge it gains from training transactions.
- Using this technique, we can answer questions like what items do people tend to buy together, thereby indicating frequent item sets. We can also associate or correlate products and items, i.e., **insights like people who buy beer also tend to buy chicken wings at a pub.**



- From Figure above, you can clearly see that based on different customer transactions over a period of time, we have obtained the items that are closely associated and customers tend to buy them together.
- Some of these frequent item sets are depicted like *{meat, eggs}, {milk, eggs} and so on*.
- The criterion of determining good quality association rules or frequent item sets is usually done using metrics like support, confidence, and lift.
- **This is an unsupervised method, because** we have no idea what the frequent item sets are or which items are more strongly associated with which items beforehand. Only after applying algorithms like the **apriori algorithm or FP-growth**, can we detect and predict products or items associated closely with each other and find conditional probabilistic dependencies.

Reinforcement Learning

- The reinforcement learning methods are a bit different from conventional supervised or unsupervised methods. In this context, we have an **agent** that we want to train over a period ~~of time to interact with a specific environment and improve its performance~~ over a period of time with regard to the type of actions it performs on the environment.

- Typically the agent starts with a set of strategies or policies for interacting with the environment. On observing the environment, it takes a particular action based on a rule or policy. **Based on the action, the agent gets a reward, which could be beneficial or detrimental in the form of a penalty.** It updates its current policies and strategies if needed and this iterative process continues till it learns enough about its environment to get the desired rewards.

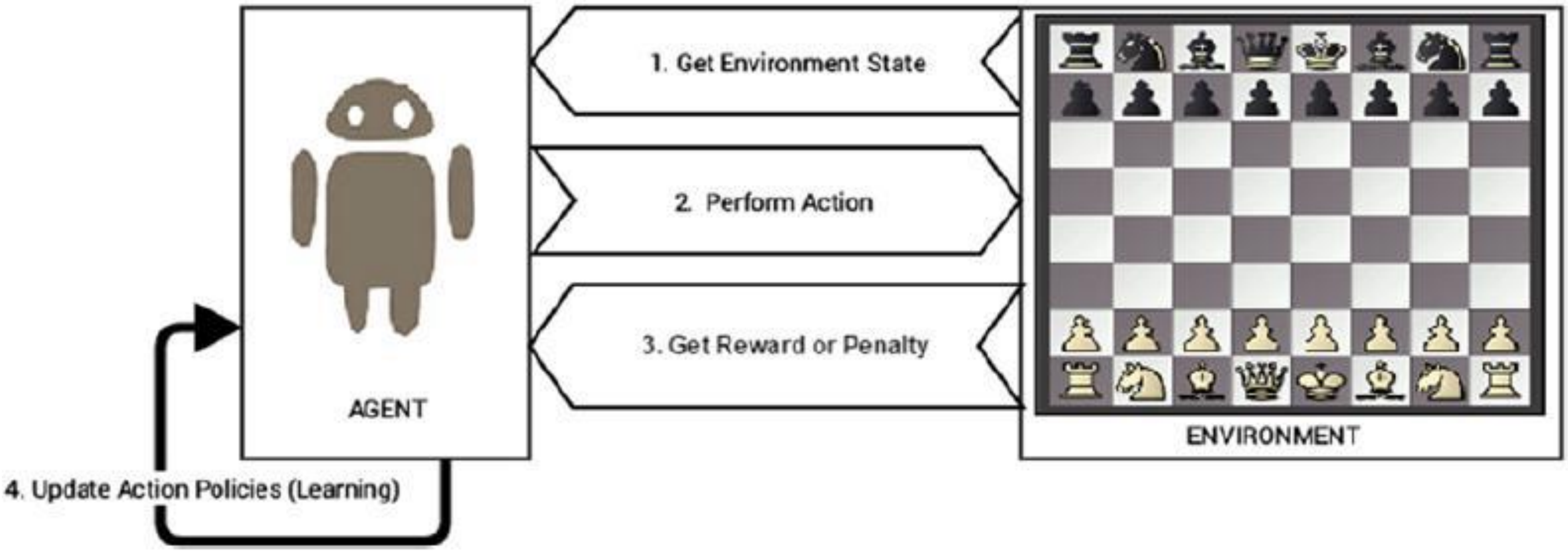
- Eg: Google Autonomous Car

The main steps of a reinforcement learning method are mentioned as follows:

1. Prepare agent with set of initial policies and strategy
2. Observe environment and current state
3. Select optimal policy and perform action
4. Get corresponding reward (or penalty)
5. Update policies if needed
6. Repeat Steps 2 - 5 iteratively until agent learns the most optimal policies

Consider a real-world problem of trying to make a robot or a machine learn to play chess. In this case the agent would be the robot and the environment and states would be the chessboard and the positions of the chess pieces.

A suitable reinforcement learning methodology is depicted in Figure.



Batch Learning

- Batch learning methods are also popularly known as offline learning methods. These are Machine Learning methods that are used in end-to-end Machine Learning systems where the ~~model is trained using all the available training data in one go. Once training~~ is done and the model completes the process of learning, on getting a satisfactory performance, it is deployed into production where it predicts outputs for new data samples.

- However, the model doesn’t keep learning over a period of time continuously with the new data. **Once the training is complete the model stops learning**. Thus, since the model trains with data in one single batch and it is usually a one-time procedure, this is known as ***batch*** or ***offline learning***.

Online Learning

- Online learning methods work in a different way as compared to batch learning methods. The **training data is usually fed in multiple incremental batches** to the algorithm. These data batches are also known as mini-batches in ML terminology.
- However, the training process does not end there unlike batch learning methods. It keeps on learning over a period of time based on new data samples which are sent to it for prediction. Basically it **predicts and learns in the process with new data** on the fly without have to re-run the whole model on previous data samples.
- Problems like **device failure** and **stock market forecasting** are two relevant scenarios.

BEWARE

- One of the major caveats in online learning methods is the fact that bad data samples can affect the model performance adversely. All ML methods work on the principle of **“Garbage In Garbage Out”.**

- Hence if you supply bad data samples to a well-trained model, it can start learning relationships and patterns that have no real significance and this ends up affecting the overall model performance. Since online learning methods keep learning based on new data samples, you should ensure proper checks are in place to notify you in case suddenly the model performance drops.

Instance Based Learning

- There are various ways to build Machine Learning models using methods that try to generalize based on input data. Instance based learning involves ML systems and methods that use the raw data points themselves to figure out outcomes for newer, previously unseen data samples instead of building an explicit model on training data and then testing it out.
- A simple example would be a **K-nearest neighbor algorithm**.
- Assuming **k = 3**.
- The ML method knows the representation of the data from the features, including its dimensions, position of each data point, and so on. For any new data point, it will use a similarity measure (like cosine or Euclidean distance) and ***find the three nearest input data points*** to this new data point. Once that is decided, we simply take a majority of the outcomes for those three training points and predict or assign it as the outcome label/response for this new data point.
- Thus, instance based learning works by looking at the input data points and using a similarity metric to generalize and predict for new data points.

Model Based Learning

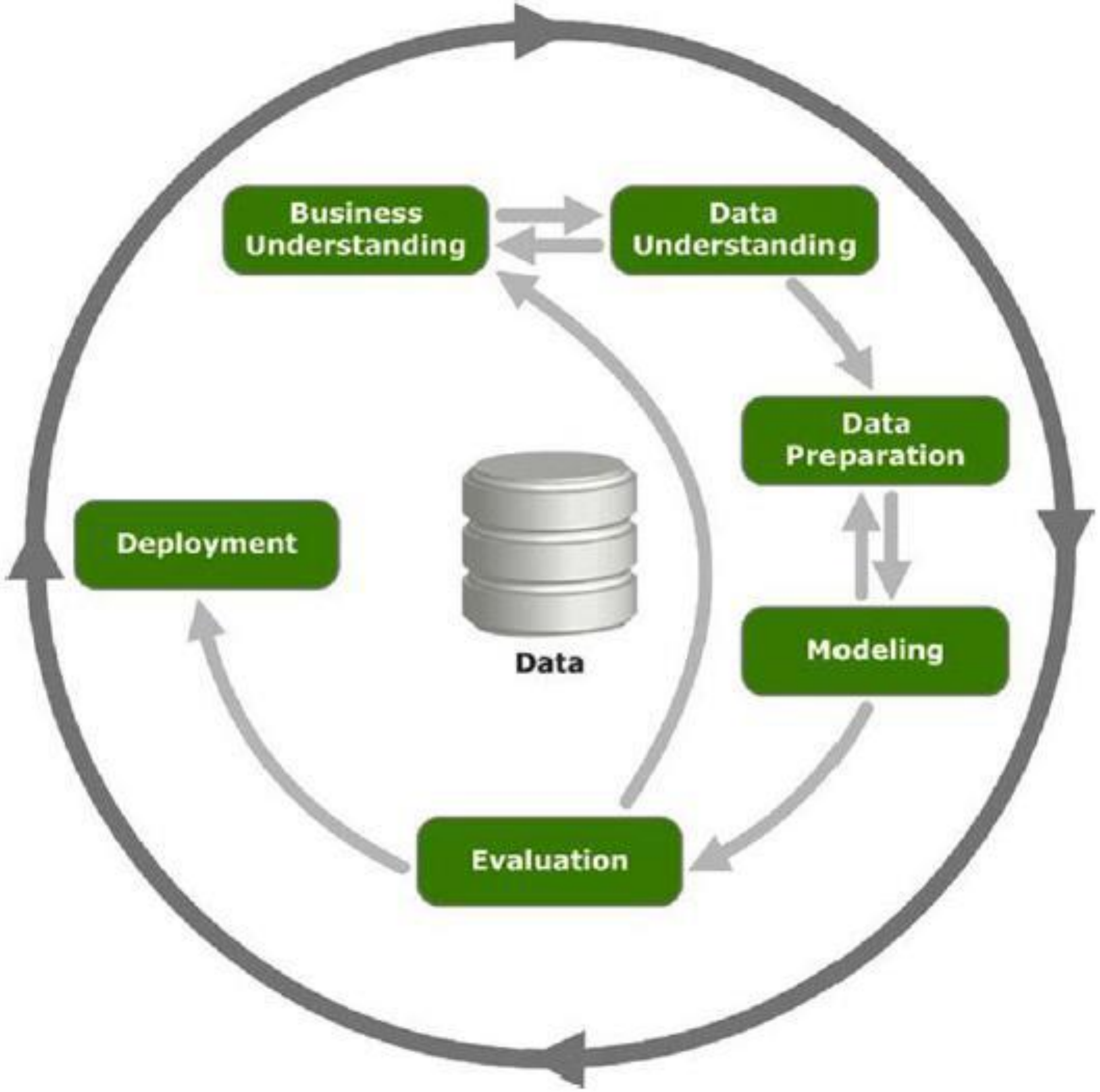
The model based learning methods are a more traditional ML approach toward generalizing based on training data. Typically an iterative process takes place where the input data is used to extract features and models are built based on various model parameters (known as ***hyperparameters***).

These **hyperparameters** are optimized based on various model validation techniques to select the model that generalizes best on the training data and some amount of validation and test data (split from the initial dataset).

Finally, the best model is used to make predictions or decisions as and when needed.

The CRISP-DM Process Model

- The CRISP-DM model stands for **CR**oss **I**ndustry **S**tandard **P**rocess for **D**ata **M**ining. More popularly known by the acronym itself, CRISP-DM is a tried, tested, and robust industry standard process model followed for data mining and analytics projects.
- The CRISP-DM model tells us that for building an end-to-end solution for any analytics project or system, there are a total of six major steps or phases, some of them being iterative.
- **Just like we have a software development lifecycle with several major phases or steps for a software development project, we have a data mining or analysis lifecycle in this scenario.**



Business Understanding

- This is the initial phase before kick starting any project in full flow. However this is one of the most important
- phases in the lifecycle!
- *The main objective here starts with understanding the business context and requirements for the problem to be solved at hand.*
- Definition of business requirements is crucial to convert the business problem into a data mining or analytics problem and to set expectations and success criteria for both the customer as well as the solution task force.
- The final deliverable from this phase would be a detailed plan with the major milestones of the project and expected timelines along with success criteria, assumptions, constraints, caveats, and challenges.

Data Understanding

- The second phase in the CRISP-DM process involves taking a deep dive ~~into the data available and understanding it in further detail~~ before starting the process of analysis.
- *This involves collecting the data, describing the various attributes, performing some exploratory analysis of the data, and keeping tabs on data quality.*
- This phase should not be neglected because *bad data or insufficient knowledge about available data can have cascading adverse effects in the later stages in this process.*

Data Preparation

- The third phase in the CRISP-DM process takes place after gaining enough knowledge on the business problem and relevant dataset.
- *Data preparation is mainly a set of tasks that are performed to clean, wrangle, curate, and prepare the data before running any analytical or Machine Learning methods and building models.*
- We will briefly discuss some of the major tasks under the data preparation phase in this section.
- **An important point to remember here is that data preparation usually is the most time consuming phase in the data mining lifecycle and often takes 60% to 70% time in the overall project.**
- *However this phase should be taken very seriously because, like we have discussed multiple times before, bad data will lead to bad models and poor performance and results.*

Modeling

- The fourth phase in the CRISP-DM process is the core phase in the ~~process where most of the analysis takes place with regard to using~~ clean, formatted data and its attributes to build models to solve business problems.
- This is an iterative process, as depicted in Figure earlier, along with model evaluation and the preceding steps leading up to modeling.
- *The basic idea is to build multiple models iteratively trying to get to the best model that satisfies our success criteria, data mining objectives, and business objectives.*

Evaluation

- The fifth phase in the CRISP-DM process takes place once we have the final models from the modeling phase that satisfy necessary success criteria with respect to our data mining goals and have the desired performance and results with regard to model evaluation metrics like accuracy.

- The evaluation phase involves carrying out a detailed assessment and review of the final models and the results which are obtained from them.

- Some of the main points that are evaluated in this section are as follows.
  - *Ranking final models based on the quality of results and their relevancy based on alignment with business objectives*
  - *Any assumptions or constraints that were invalidated by the models*
  - *Cost of deployment of the entire Machine Learning pipeline from data extraction and processing to modeling and predictions*
  - *Any pain points in the whole process? What should be recommended? What should be avoided?*
  - *Data sufficiency report based on results*
  - *Final suggestions, feedback, and recommendations from solutions team and SMEs*

- **Based on the report formed from these points, after a discussion, the team can decide whether they want to proceed to the next phase of model deployment or a full reiteration is needed, starting from business and data understanding to modeling.**

Deployment

- The final phase in the CRISP-DM process is all about deploying your selected models to production and making sure the transition from development to production is seamless.
- Usually most organizations follow a standard path-to-production methodology. A proper plan for deployment is built based on resources required, servers, hardware, software, and so on. *Models are validated, saved, and deployed on necessary systems and servers.*
- A plan is also put in place for regular monitoring and maintenance of models to continuously evaluate their performance, check for results and their validity, and retire, replace, and update models as and when needed.

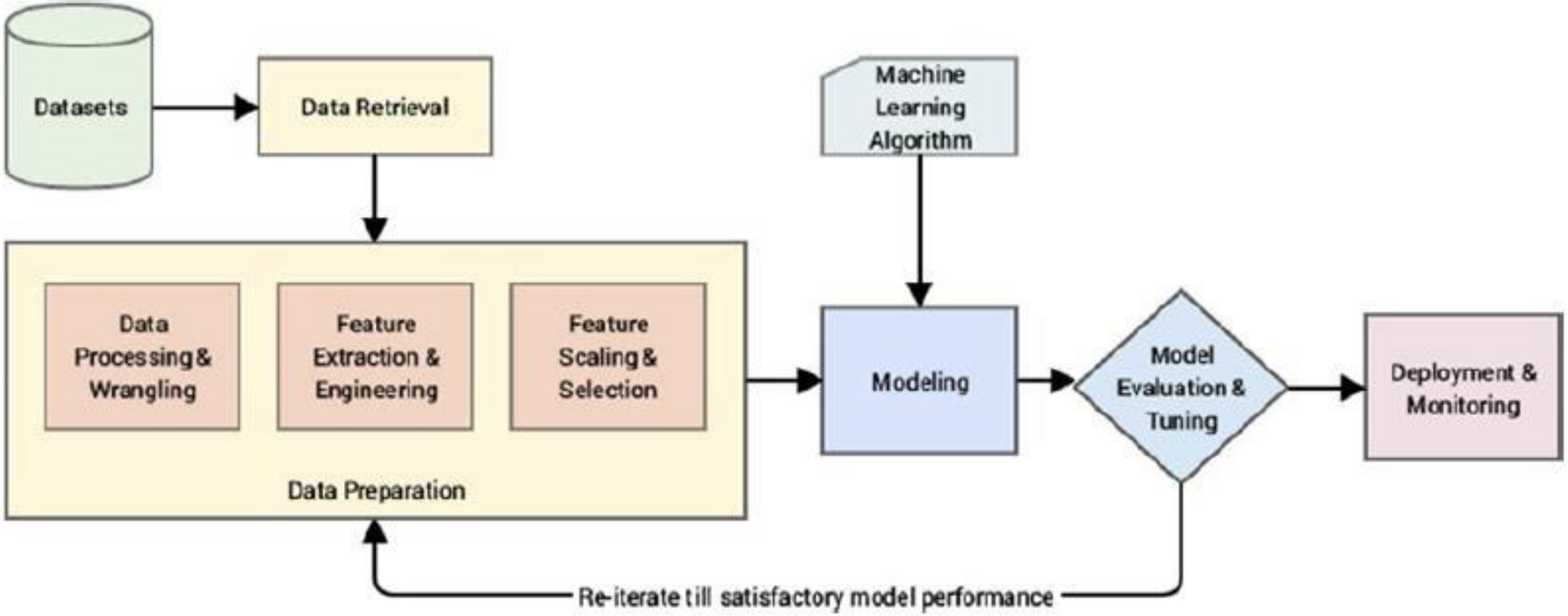
Building Machine Intelligence

Machine intelligence can be built using ***non-traditional computing*** approaches like **Machine Learning**.

This is done by using **Machine Learning pipelines** (based on the CRISP-DM model), which will help us solve real-world problems by building machine intelligence using a structured process.

❑Machine Learning Pipelines

The best way to solve a real-world Machine Learning or analytics problem is to use a Machine Learning pipeline starting from getting your data to transforming it into information and insights using Machine Learning algorithms and techniques.


Datasets
Data Retrieval
Machine Learning Algorithm
Data Processing & Wrangling
Feature Extraction & Engineering
Feature Scaling & Selection
Data Preparation
Modeling
Model Evaluation & Tuning
Deployment & Monitoring
Re-iterate till satisfactory model performance

The major steps in the pipeline are briefly mentioned here.

1. **Data retrieval**: This is mainly data collection, extraction, and acquisition from various data sources and data stores.

2. **Data preparation**: In this step, we pre-process the data, clean it, wrangle it, and manipulate it as needed. Next steps involved extracting, engineering, and selecting features/attributes from the data.
   - **Data processing and wrangling:** Mainly concerned with data processing, cleaning, munging, wrangling and performing initial descriptive and exploratory data analysis(wrangling and munging are similar).
   - **Feature extraction and engineering:** Here, we extract important features or attributes from the raw data and even create or engineer new features from existing features.
   - **Feature scaling and selection:** Data features often need to be normalized and scaled to prevent Machine Learning algorithms from getting biased. Besides this, often we need to select a subset of all available features based on feature importance and quality. This process is known as feature selection(scaling and normalization are similar).

Data Wrangling

The process of data wrangling or data munging involves data processing, cleaning, normalization, and formatting.

Data in its raw form is rarely consumable by Machine Learning methods to build models. Hence we need to process the data based on its form, clean underlying errors and inconsistencies, and format it into more consumable formats for ML algorithms.

Following are the main tasks relevant to data wrangling.

1. **Handling missing values (remove rows, impute missing values)**
2. **Handling data inconsistencies (delete rows, attributes, fix inconsistencies)**
3. **Fixing incorrect metadata and annotations**
4. **Handling ambiguous attribute values**
5. **Curating and formatting data into necessary formats (CSV, Json, relational)**

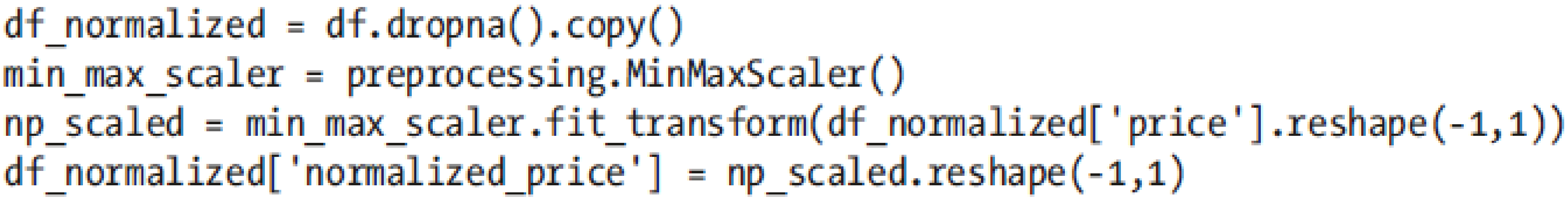
Normalizing Values :

Attribute normalization is the process of standardizing the range of values of attributes. Machine learning algorithms in many cases utilize distance metrics, attributes or features of different scales/ranges which might adversely affect the calculations or bias the outcomes.

Normalization is also called feature scaling.

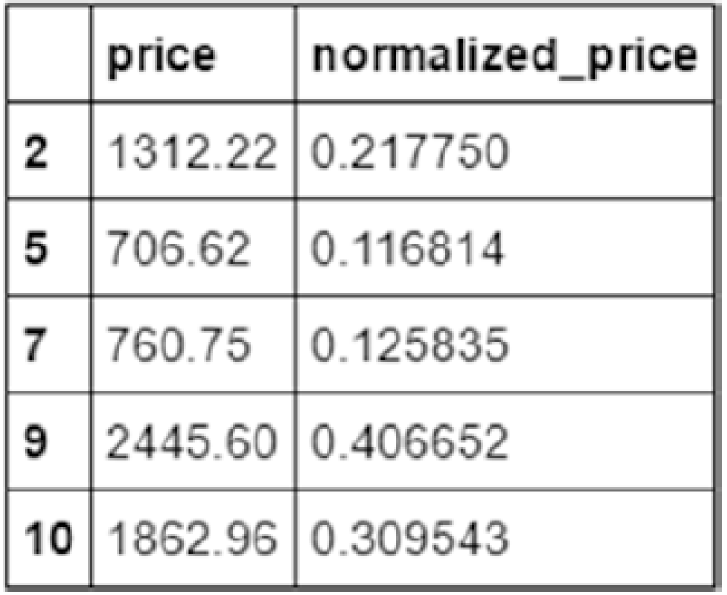
There are various ways of scaling/normalizing features, we may choose a normalization technique based upon the feature, algorithm and use case at hand.

The following snippet showcases a quick example of using a **min-max scaler**, available from the **preprocessing module of sklearn**, which **rescales** attributes to the desired given range.



```
df_normalized = df.dropna().copy()
min_max_scaler = preprocessing.MinMaxScaler()
np_scaled = min_max_scaler.fit_transform(df_normalized['price'].reshape(-1,1))
df_normalized['normalized_price'] = np_scaled.reshape(-1,1)
```

Original and normalized values for price



	price	normalized_price
2	1312.22	0.217750
5	706.62	0.116814
7	760.75	0.125835
9	2445.60	0.406652
10	1862.96	0.309543

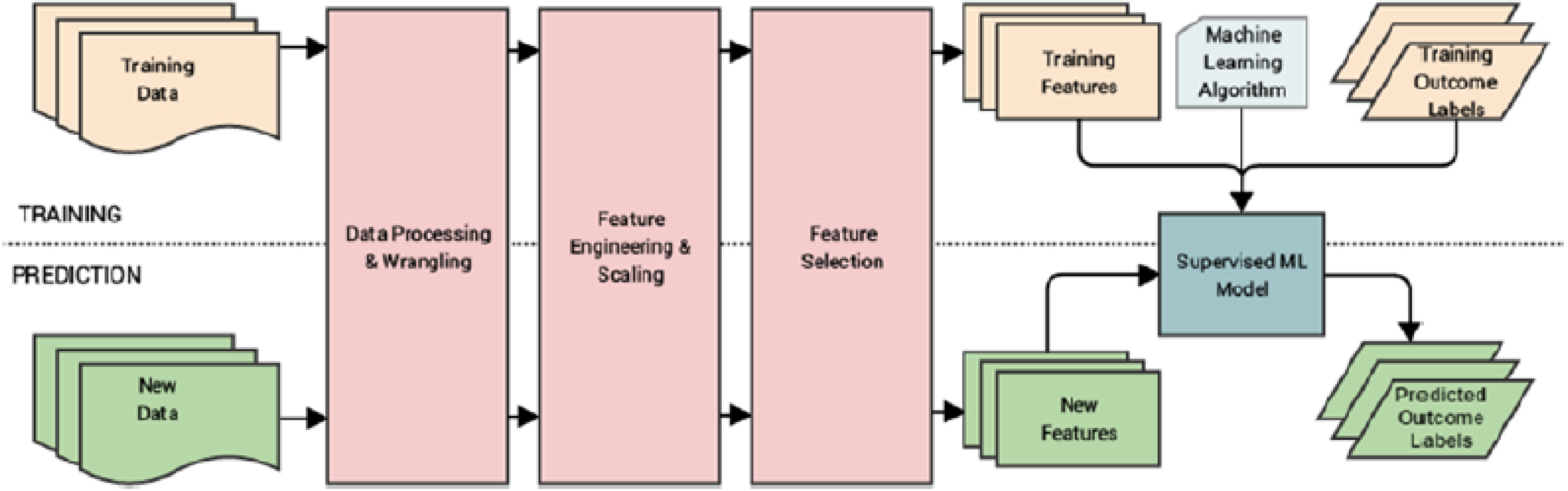
3. **Modeling:** In the process of modeling, we usually feed the data features to a Machine Learning method or algorithm and train the model, typically to optimize a specific cost function in most cases with the objective of reducing errors and generalizing the representations learned from the data.

4. **Model evaluation and tuning:** Built models are evaluated and tested on validation datasets and, based on metrics like accuracy, F1 score, and others, the model performance is evaluated. Models have various parameters that are tuned in a process called hyperparameter optimization to get models with the best and optimal results.

5. **Deployment and monitoring:** Selected models are deployed in production and are constantly monitored based on their predictions and results.

Supervised Machine Learning Pipeline

By now we know that supervised Machine Learning methods are all about working with supervised labeled data to train models and then predict outcomes for new data samples. You can clearly see the two phases of model training and prediction highlighted. **Based on our earlier generic ML pipeline**, ***Figure*** shows a standard **supervised ML pipeline.**



Un-Supervised Machine Learning Pipeline

Unsupervised Machine Learning is all about extracting patterns, relationships, associations, and clusters from data. The processes related to feature engineering, scaling and selection are similar to supervised learning. However there is no concept of pre-labeled data here. Hence the unsupervised Machine Learning pipeline would be slightly different in contrast to the supervised pipeline.

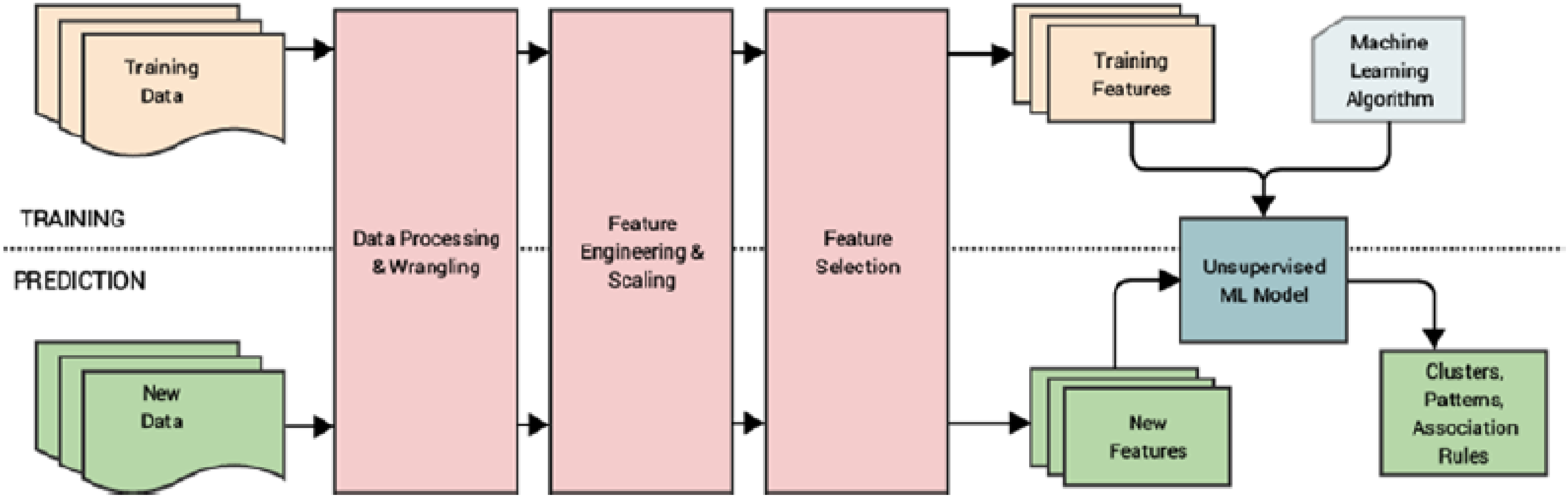

Training Data
TRAINING
PREDICTION
New Data
Data Processing & Wrangling
Feature Engineering & Scaling
Feature Selection
Training Features
Machine Learning Algorithm
Unsupervised ML Model
New Features
Clusters, Patterns, Association Rules

Figure clearly depicts that no supervised labeled data is used for training the model.

Challenges in Machine Learning

Machine Learning is a rapidly evolving, fast-paced, and exciting field with a lot of prospect, opportunity, and scope. However it comes with its own set of challenges, due to the complex nature of Machine Learning methods, its dependency on data, and not being one of the more traditional computing paradigms.

The following points cover some of the main challenges in Machine Learning.

i. Data quality issues lead to problems, especially with to data processing and feature extraction.

ii. Data acquisition, extraction, and retrieval is an extremely tedious and time consuming process.

iii. Lack of good quality and sufficient training data in many scenarios.

iv. Formulating business problems clearly with well-defined goals and objectives.

v. Feature extraction and engineering, especially hand-crafting features, is one of the most difficult yet important tasks in Machine Learning.

vi. Overfitting or underfitting models can lead to the model learning poor representations and relationships from the training data leading to detrimental performance.

vii. The curse of dimensionality: too many features can be a real hindrance.

viii. Complex models can be difficult to deploy in the real world.

Real-World Case Study: Predicting Student Grant Recommendations

- Let’s take a step back from what we have learned so far! The main objective here was to gain a solid grasp over the entire Machine Learning landscape, understand crucial concepts, build on the basic foundations, and understand how to execute Machine Learning projects with the help of Machine Learning pipelines with the CRISP-DM process model being the source of all inspiration.
- Let’s put all this together to take a very basic real-world case study by building a supervised Machine Learning pipeline on a toy dataset. Our major objective is as follows. Given that you have several students with multiple attributes like grades, performance, and scores, can you build a model based on past historical data to predict the chance of the student getting a recommendation grant for a research project?
- This will be a quick walkthrough with the main intent of depicting how to build and deploy a real-world Machine Learning pipeline and perform predictions. This will also give you a good hands-on experience to get started with Machine Learning. Do not worry too much if you don’t understand the details of each and every line of code; the Trainer shall cover all the tools, techniques, and frameworks used here in detail.

Refer NB: 2 Logistic Regression CaseStudy_1_ml_Predicting Student Grant Recommendations.ipynb

Thank You