AI and Ensemble Machine Learning Algorithm using Python

by

Dr. Santosh Kumar Nanda

Asst. General Manager (Lead Data Scientist), Flytxt Mobile Solution Pvt Ltd

santoshnanda@live.in

Flytxt Mobile Solution Pvt Ltd

October 13, 2018

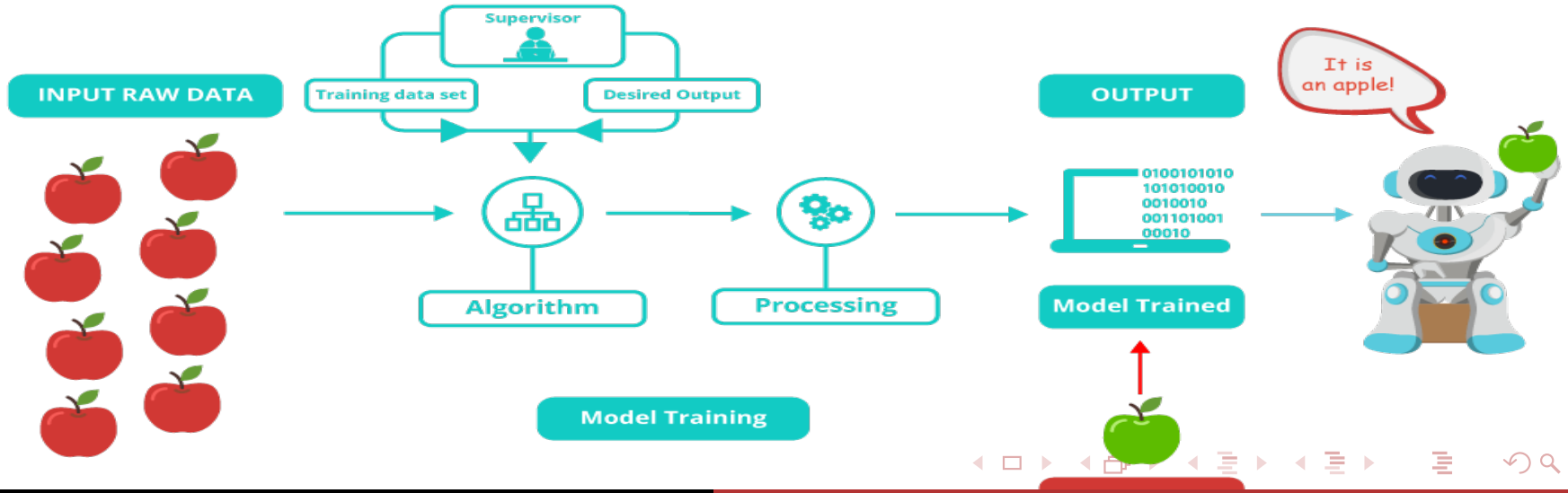
"AI is a core, transformative way by which we are rethinking how we are doing everything."

CEO, Google

Outline

Artificial intelligence (AI)

- Artificial intelligence (AI)
  1. Computers with the ability to mimic or duplicate the functions of the human brain
- Artificial intelligence systems
  1. The people, procedures, hardware, software, data, and knowledge needed to develop computer systems and machines that demonstrate the characteristics of intelligence



◇ Intelligent behavior

1. Learn from experience
2. Apply knowledge acquired from experience
3. Handle complex situations
4. Solve problems when important information is missing
5. Determine what is important
6. React quickly and correctly to a new situation
7. Understand visual images
8. Process and manipulate symbols
9. Be creative and imaginative
10. Use heuristics

Major Branches of AI

- Perceptive system
  1. A system that approximates the way a human sees, hears, and feels objects
- Vision system
  1. Capture, store, and manipulate visual images and pictures
- Robotics
  1. Mechanical and computer devices that perform tedious tasks with high precision
- Expert system
  1. Stores knowledge and makes inferences
- Learning system
  1. Computer changes how it functions or reacts to situations based on feedback
- Natural language processing
  1. Computers understand and react to statements and commands made in a "natural" language, such as English
- Neural network
  1. Computer system that can act like or simulate the functioning of the human brain

Core Structure of AI

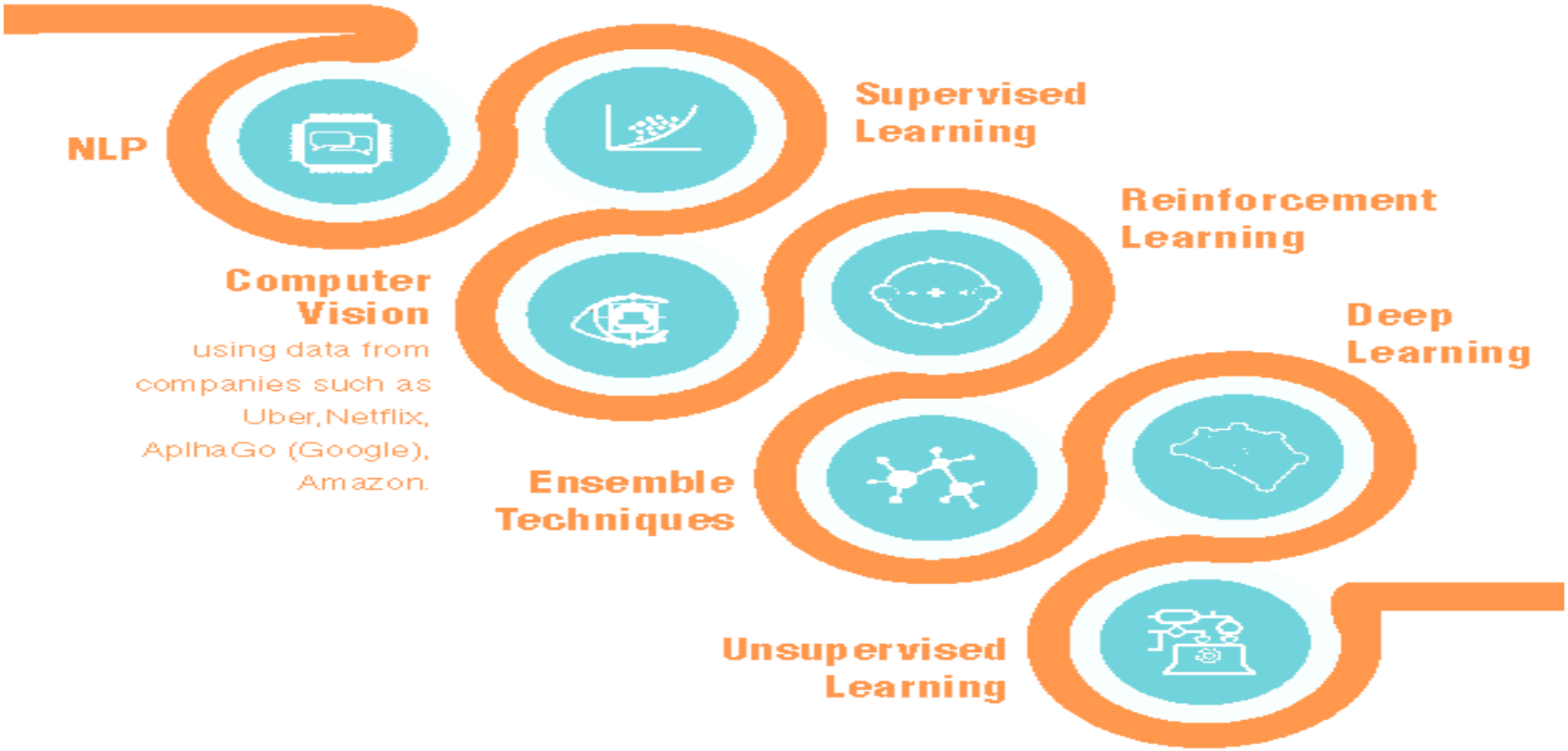


Figure: Core Structure of AI

Recent Opportunity of AI

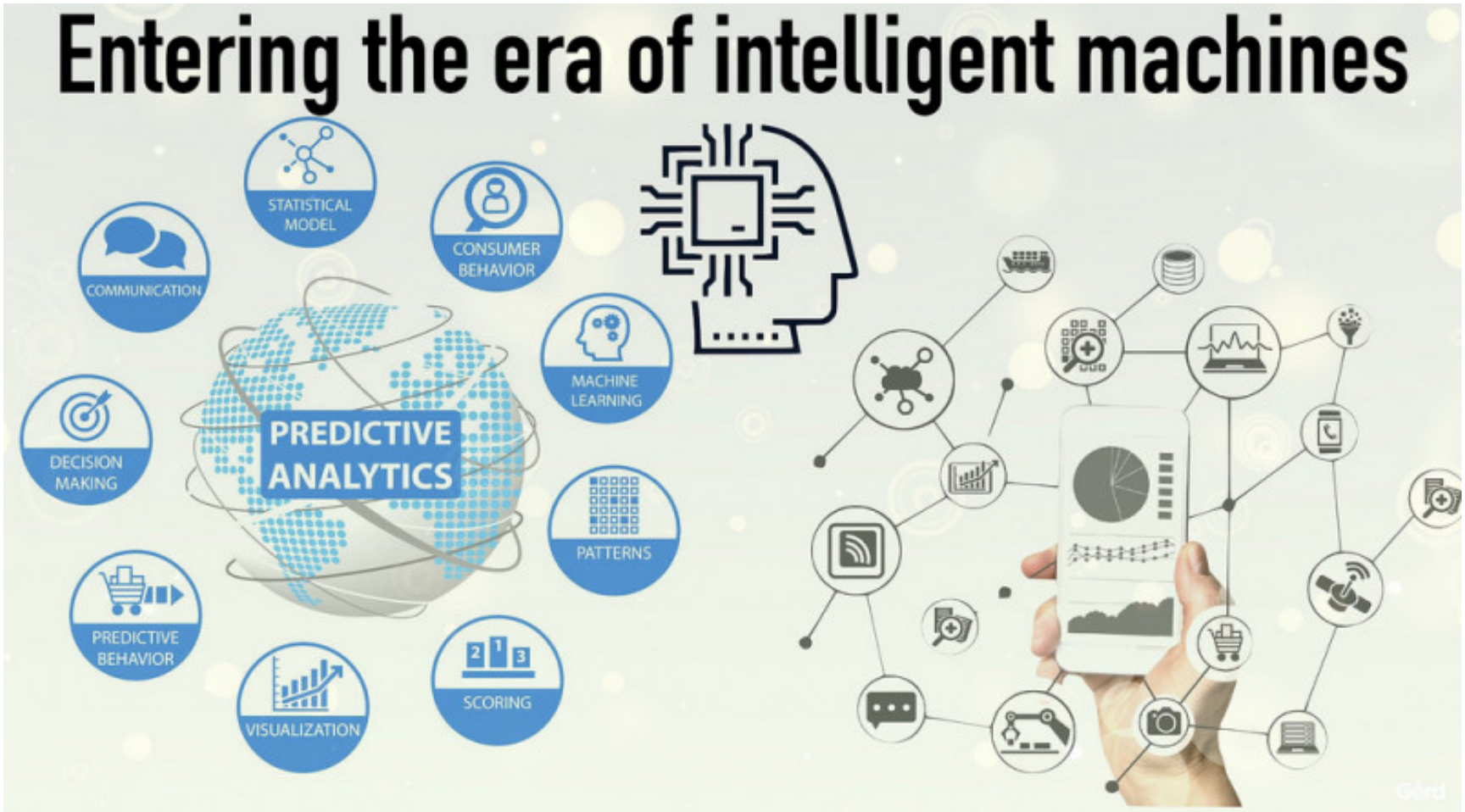
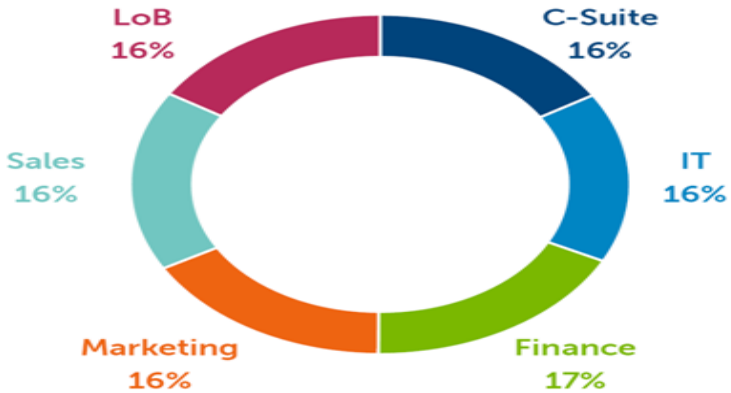


Figure: Recent Opportunity of AI

Why Data Science ?

1. The demand of internet use is more as number of users who are used smart phone, small portable high performance computing devices is also increased in last 10 years
2. Similarly in India, the growth of Smartphone penetration in India has directly resulted in a massive increase in the amount of data being generated.
3. This number is only set to grow further as high-speed internet services are becoming more accessible and easily affordable in both urban and rural areas.
4. However, Globalization, economic growth, and the availability of easy-to-use, low-cost smartphones are also playing a major role in driving greater data generation in the country
5. Therefore, the role of Data Scientist is now very important in almost of all industries

Growth of Data by Users in 2017

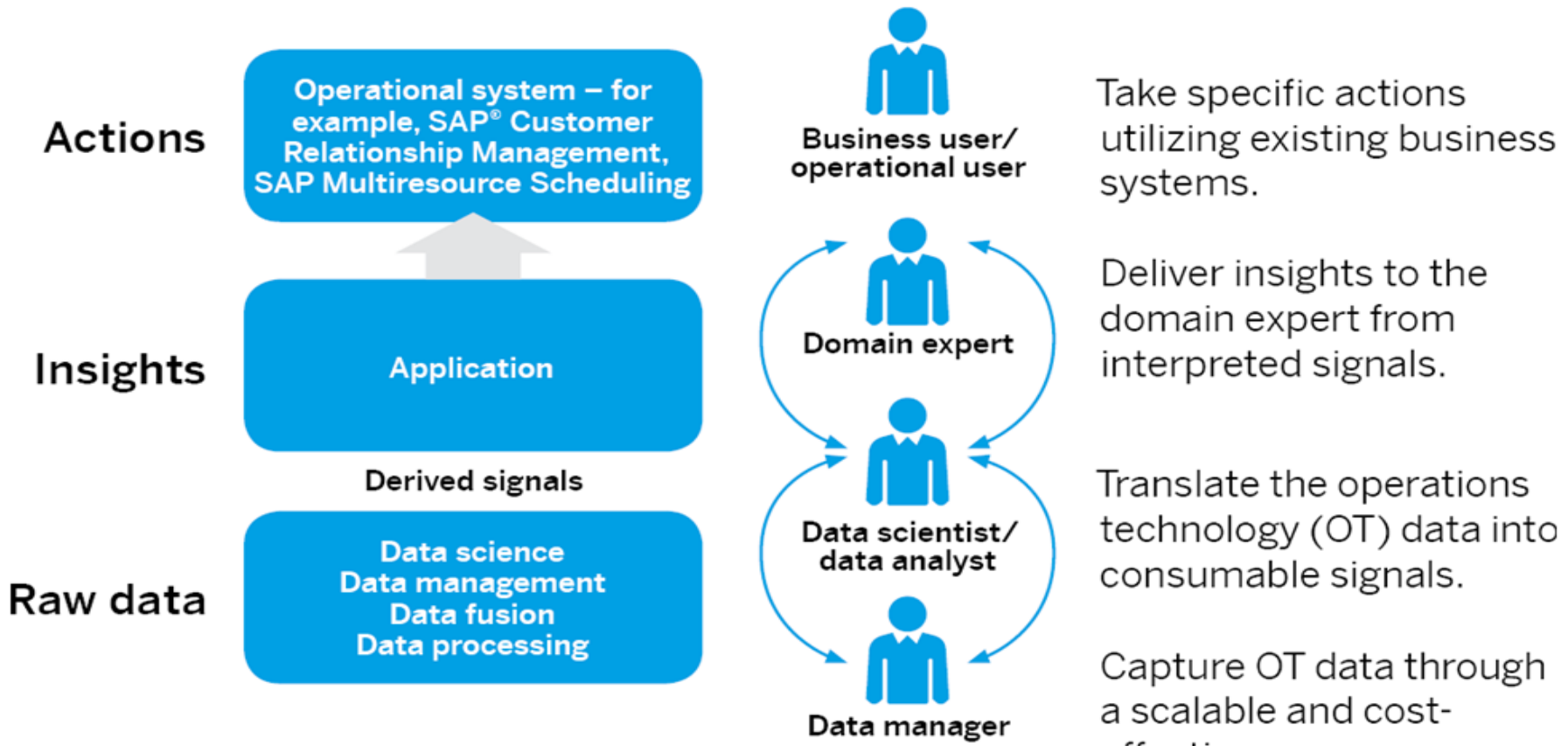


Source: Dell

Only 16% of projects are championed by IT

1. Over the last few years Data Science has really changed our concept of technology. Now, lives a lot easier as compared to 10yrs ago, and this is all because of data science.
2. Data Science has really pulled the ends between fiction and technology, for example, Linkedin, Youtube, Amzaon etc, data-science is being used everywhere.
3. Online transaction or Digitization of transaction is another factor that has driven the growth of corporate data. Compression technologies and virtualization are enabling companies to do more with their technological assets. It can be concluded that big data is no longer the sole domain of massive businesses like retail chains and financial institutions.
4. The growth of e-commerce as well as the prevalence of social media, online blogs, and e-mail exchanges has led to data coming in from multiple sources, and enterprises are looking for solutions that can extract valuable information from this data.
5. Cloud computing has also resulted in a manifold rise in the amount of data generated.

Data Science Requirements and Predictive Maintenance Domain



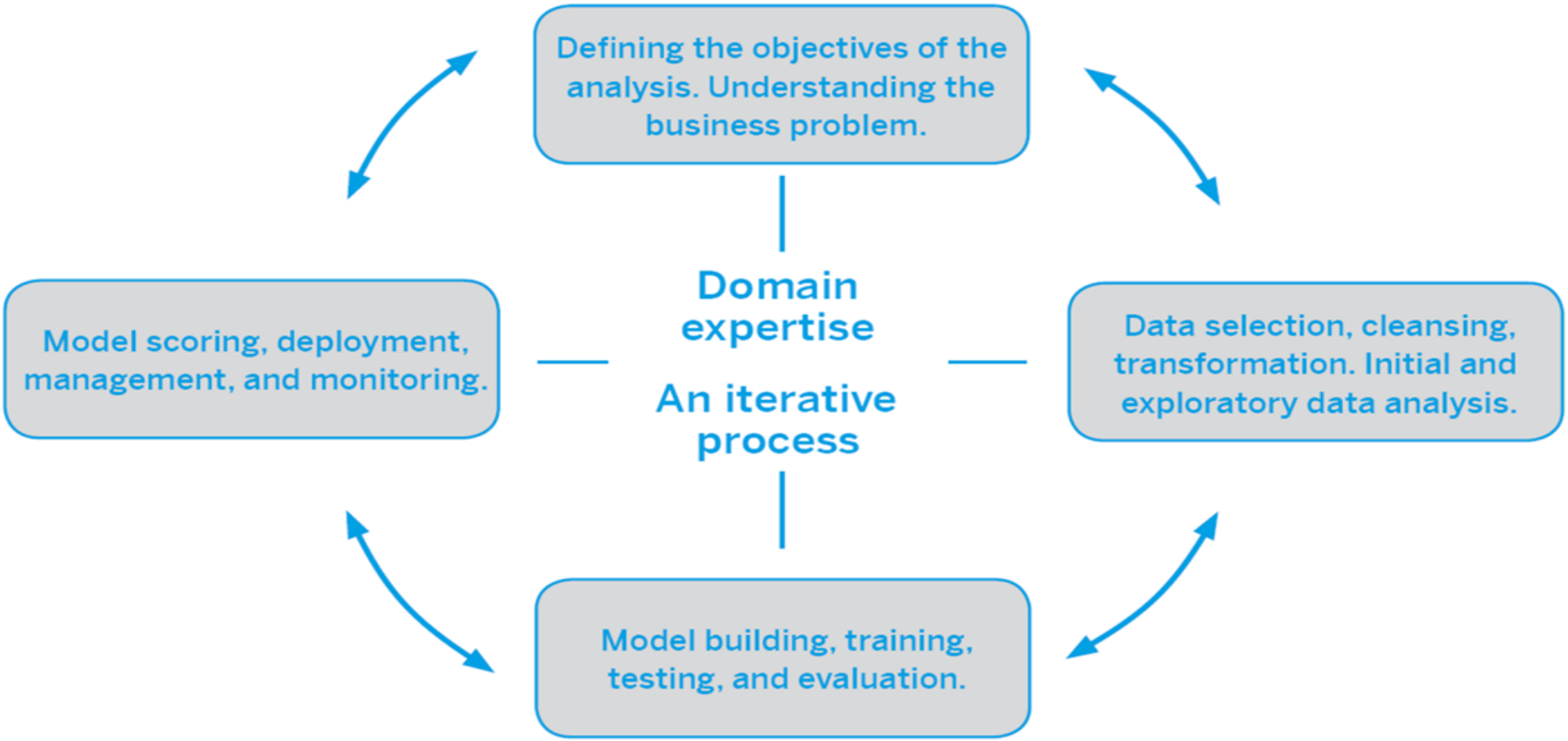


Figure: Domain Expert

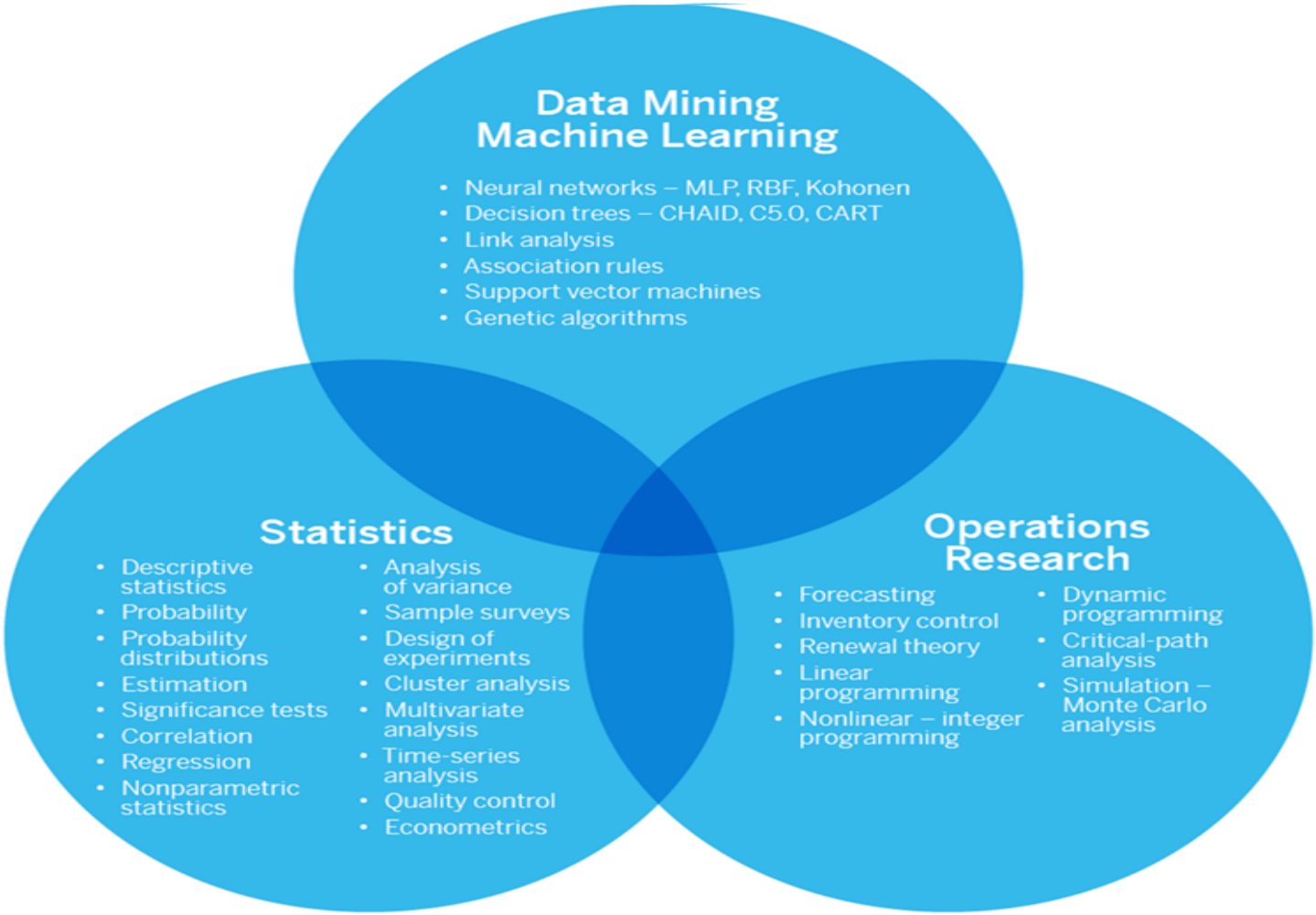
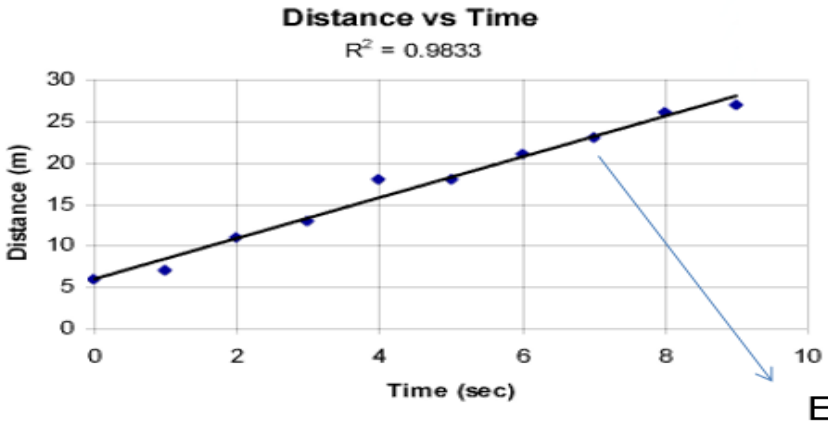
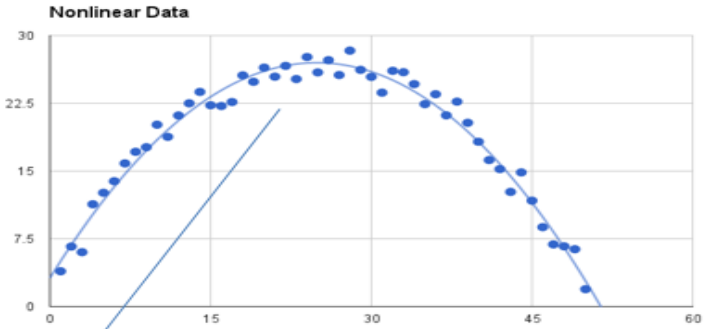


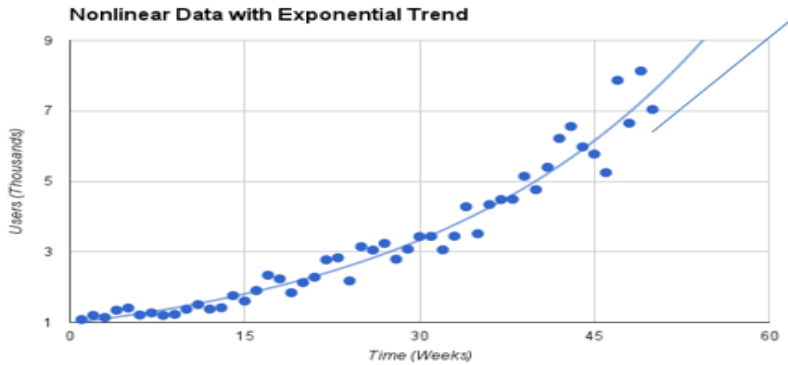
Figure: Importance of Machine Learning

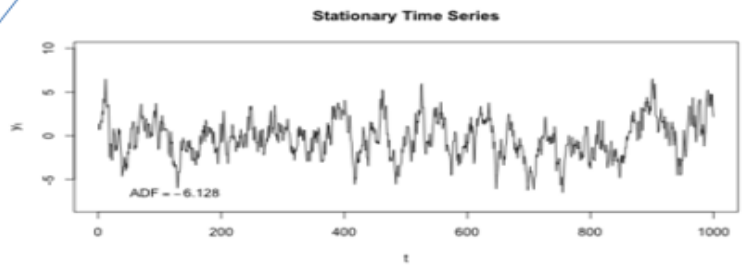
Types of Data and Data Analysis

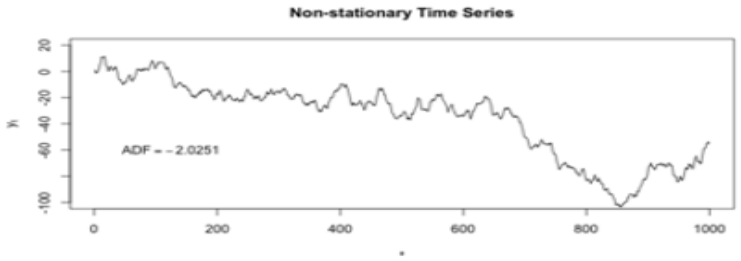




Exact trend







1. Data analysis, also known as analysis of data or data analytics, is a process of inspecting, cleansing,transforming, and modelling data with the goal of discovering useful information, suggesting conclusions, and supporting decision-making.
2. Data analysis has multiple facets and approaches, encompassing diverse techniques under a variety of names, in different business, science, and social science domains.
3. Data analysis and curation is the basis for all other quantitative methods
4. Data consistency throughout company is key (master scales, data warehouses, etc.)
5. Typically, weakest link: industry is not collecting the right data which inhibits use of analytics
6. Recent trends from description to learning
7. machine learning at several large companies

Failure of Statistical Models in Data Prediction Statistical Analysis

1. Focuses on **hypothesis testing** and **parameter estimation**
2. Fits **parsimonious statistical models** with the goal to **explain** complex relationships with fewer parameters
3. Examples: Regression, PLS, PCA, non-parametric statistics, quality control

Pattern Recognition (Data Mining) Algorithms include:

1. Trees, boosted & voted trees (forests), SVM, neural nets, ...
2. Deep learning, cognitive computing

Models are validated through

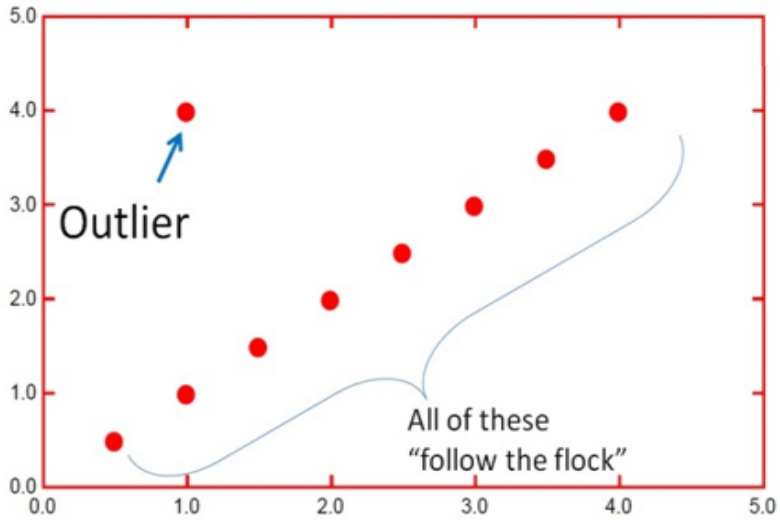
1. Statistics computed against test (hold-out) samples; Accuracy, MS Error, ROC, AUC, Lift,...
2. Established best practices

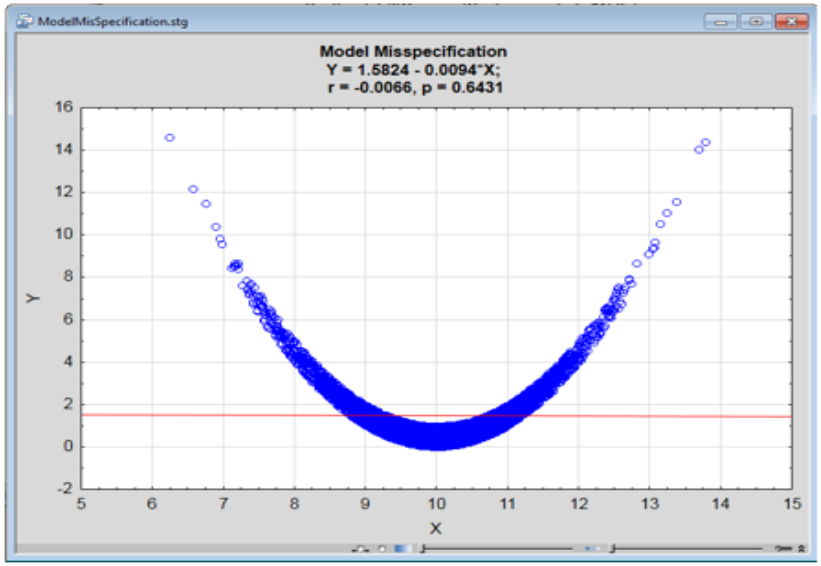
•When statistical models are misspecified, you can get no results, wrong results

Obviously, relationships can be
- •Nonlinear
- •non-montonic

•Non-additive, or
with partial interactions

Relationships can be "difficult"
- E.g., consider a linear model to predict Hits and Misses, when relationships are like this…
- Machine learning will detect and model this relationship, and provide insight into the process



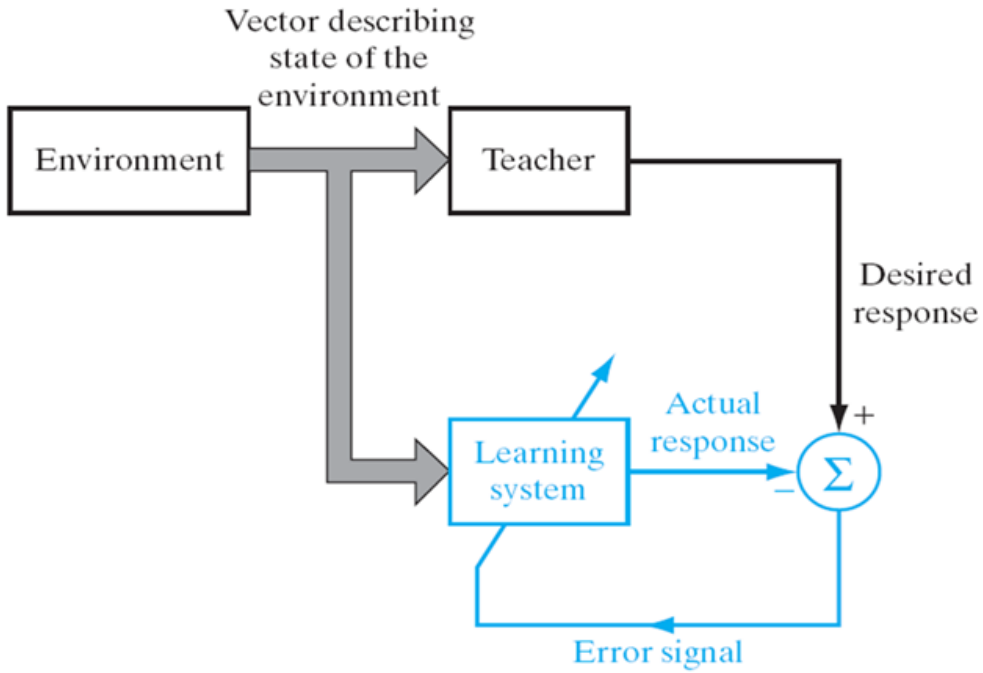


Machine Learning : Why and Applications

1. Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that machines should be able to learn and adapt through experience
2. Many machine learning algorithms have been around for a long time, the ability to automatically apply complex mathematical calculations to big data – over and over, faster and faster – is a recent development. Here are a few widely publicized examples of machine learning applications familiar with:
3. Self-driving Google car? The essence of machine learning.
4. Online recommendation offers such as those from Amazon and Netflix? Machine learning applications for everyday life.
5. Knowing what customers are saying about you on Twitter? Machine learning combined with linguistic rule creation.
6. Fraud detection ? One of the more obvious, important uses in our world today.

Machine Learning : Types

Learning with a teacher is also referred to as supervised learning



In supervised learning, the learning process takes place under the tutelage of a teacher. However, in the paradigm known as learning without a teacher, as the name, there is no teacher to oversee the learning process

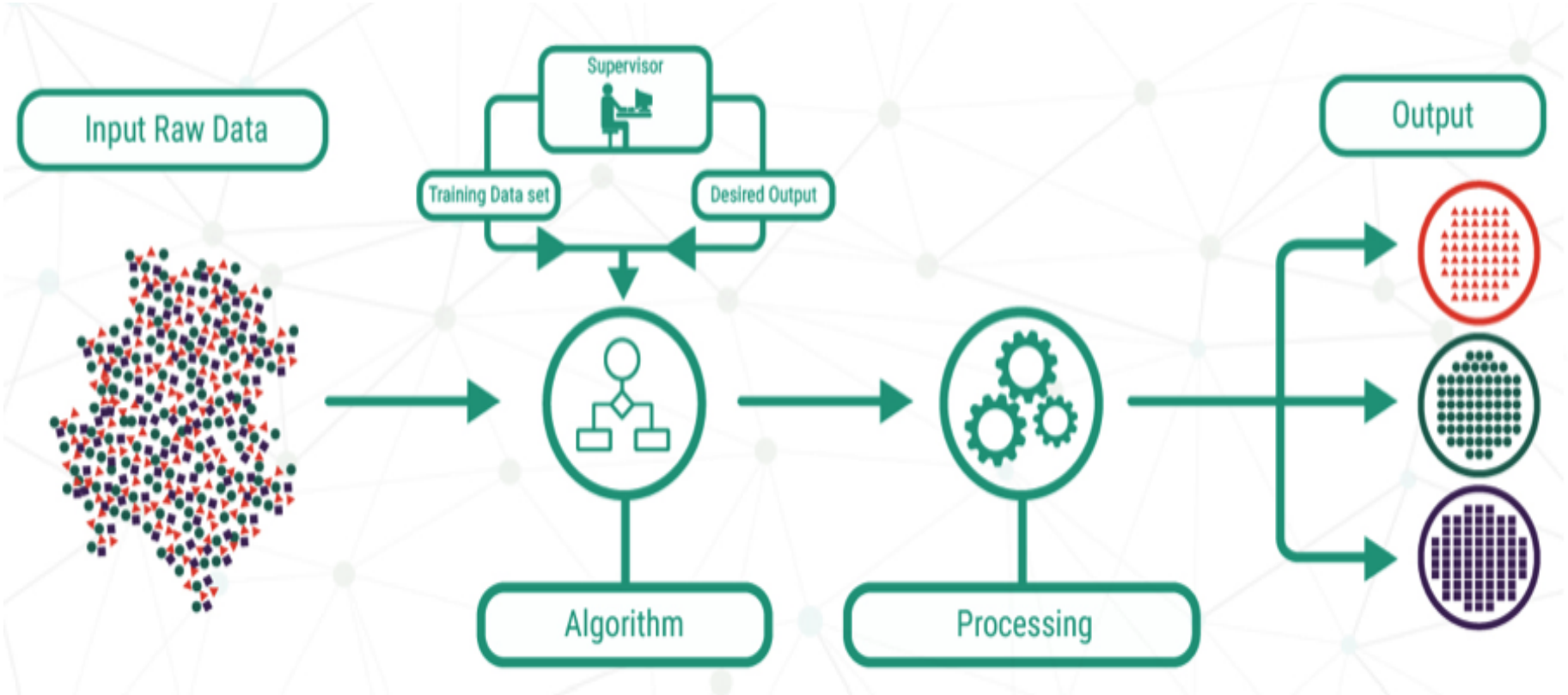
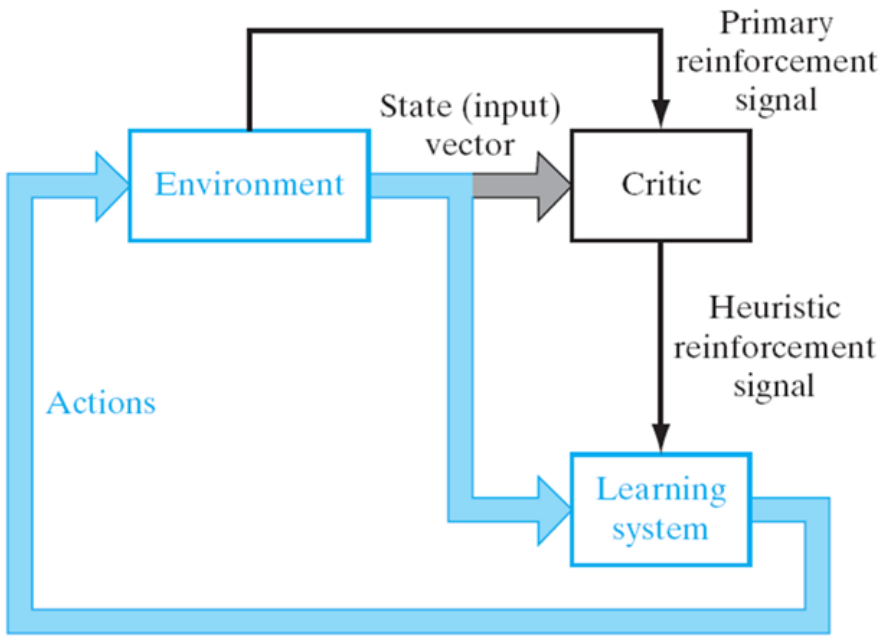


Figure: Supervised Learning

Learning with a critic is also referred to as reinforcement learning



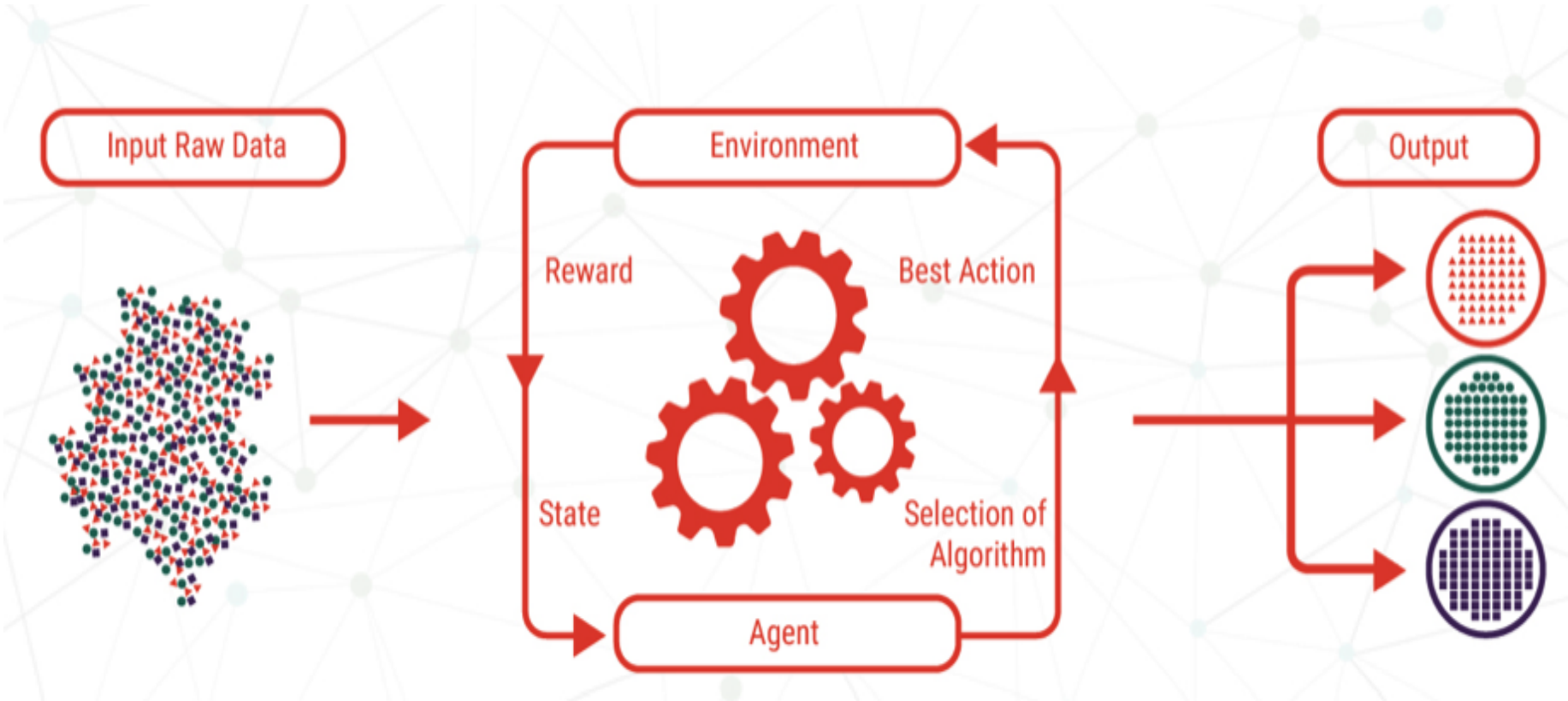
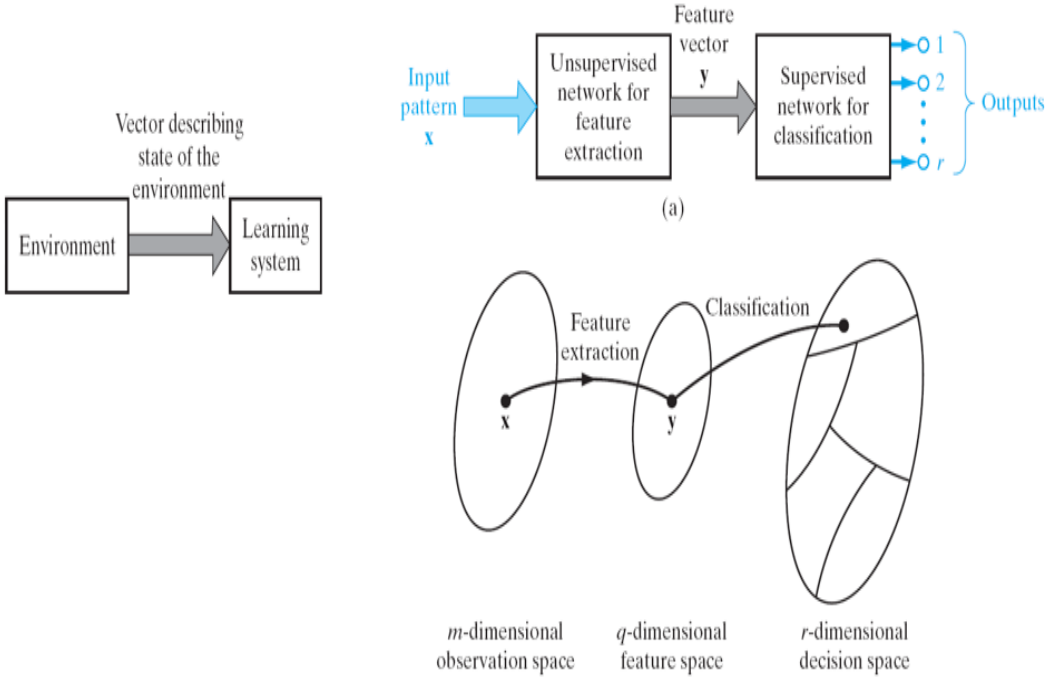


Figure: Reinforcement Learning

Unsupervised, or self-organized, learning by it self (without teacher and non-available of critic)



To perform unsupervised learning, we may use a competitive-learning rule

1. Kalman Filter
2. K-mean Clustering
3. Principle of Component Analysis

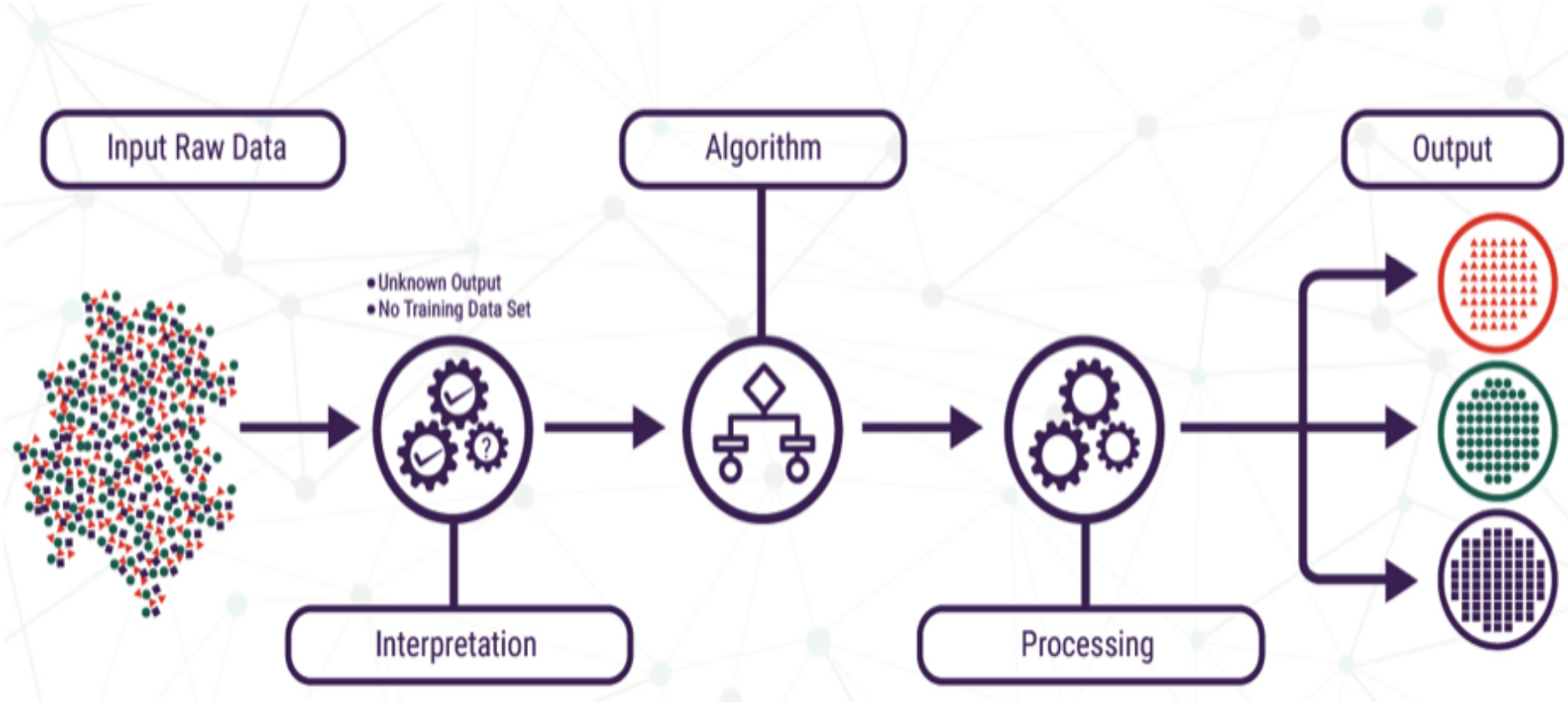


Figure: Un-Supervised Learning

- Predictive maintenance or condition monitoring
- Warranty reserve estimation
- Propensity to buy
- Demand forecasting
- Process optimization
- Telematics

Manufacturing

- Predictive inventory planning
- Recommendation engines
- Upsell and cross-channel marketing
- Market segmentation and targeting
- Customer ROI and lifetime value

Retail

- Alerts and diagnostics from real-time patient data
- Disease identification and risk stratification
- Patient triage optimization
- Proactive health management
- Healthcare provider sentiment analysis

Healthcare and Life Sciences

- Aircraft scheduling
- Dynamic pricing
- Social media – consumer feedback and interaction analysis
- Customer complaint resolution
- Traffic patterns and congestion management

Travel and Hospitality

- Risk analytics and regulation
- Customer Segmentation
- Cross-selling and up-selling
- Sales and marketing campaign management
- Credit worthiness evaluation

Financial Services

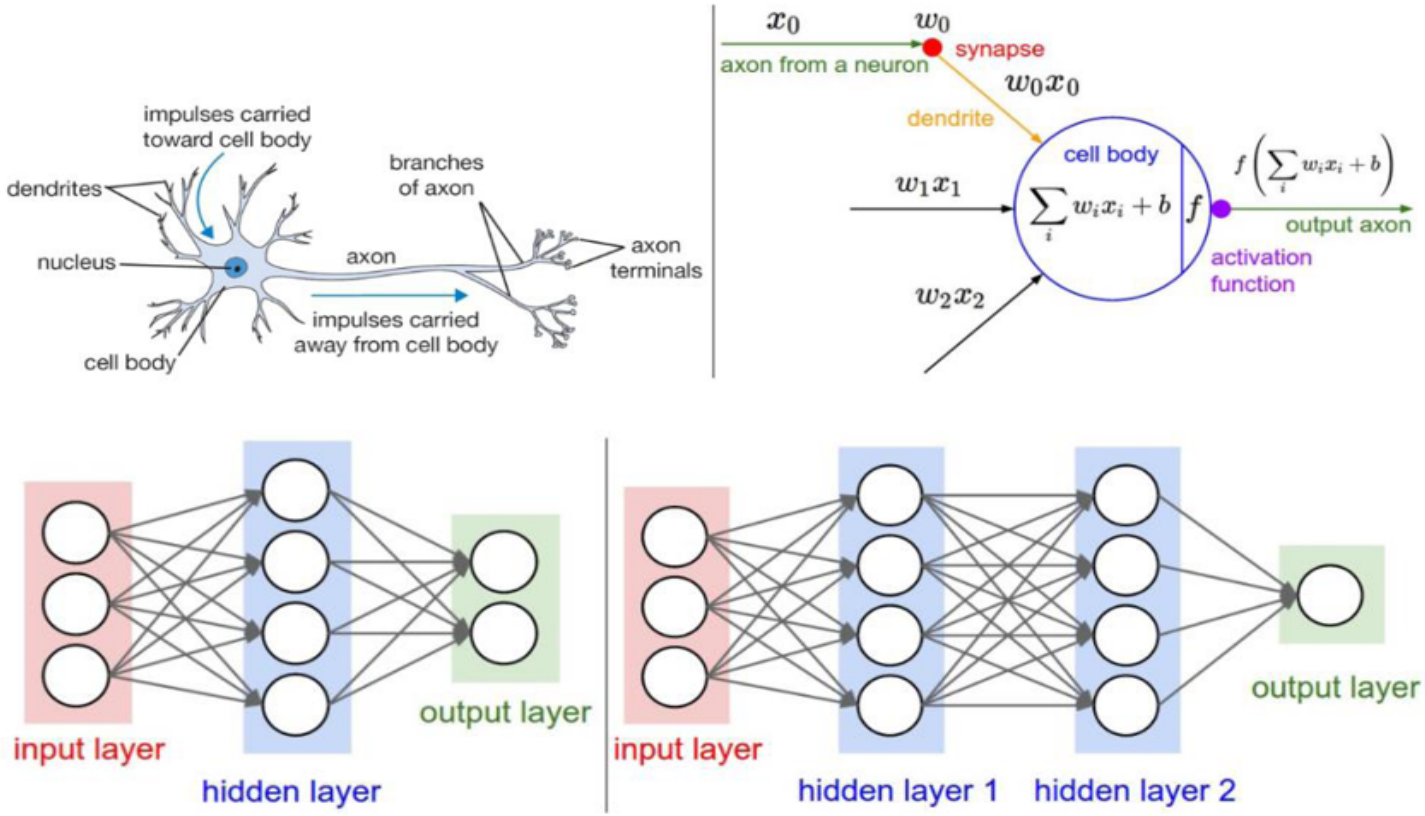
- Power usage analytics
- Seismic data processing
- Carbon emissions and trading
- Customer-specific pricing
- Smart grid management
- Energy demand and supply optimization

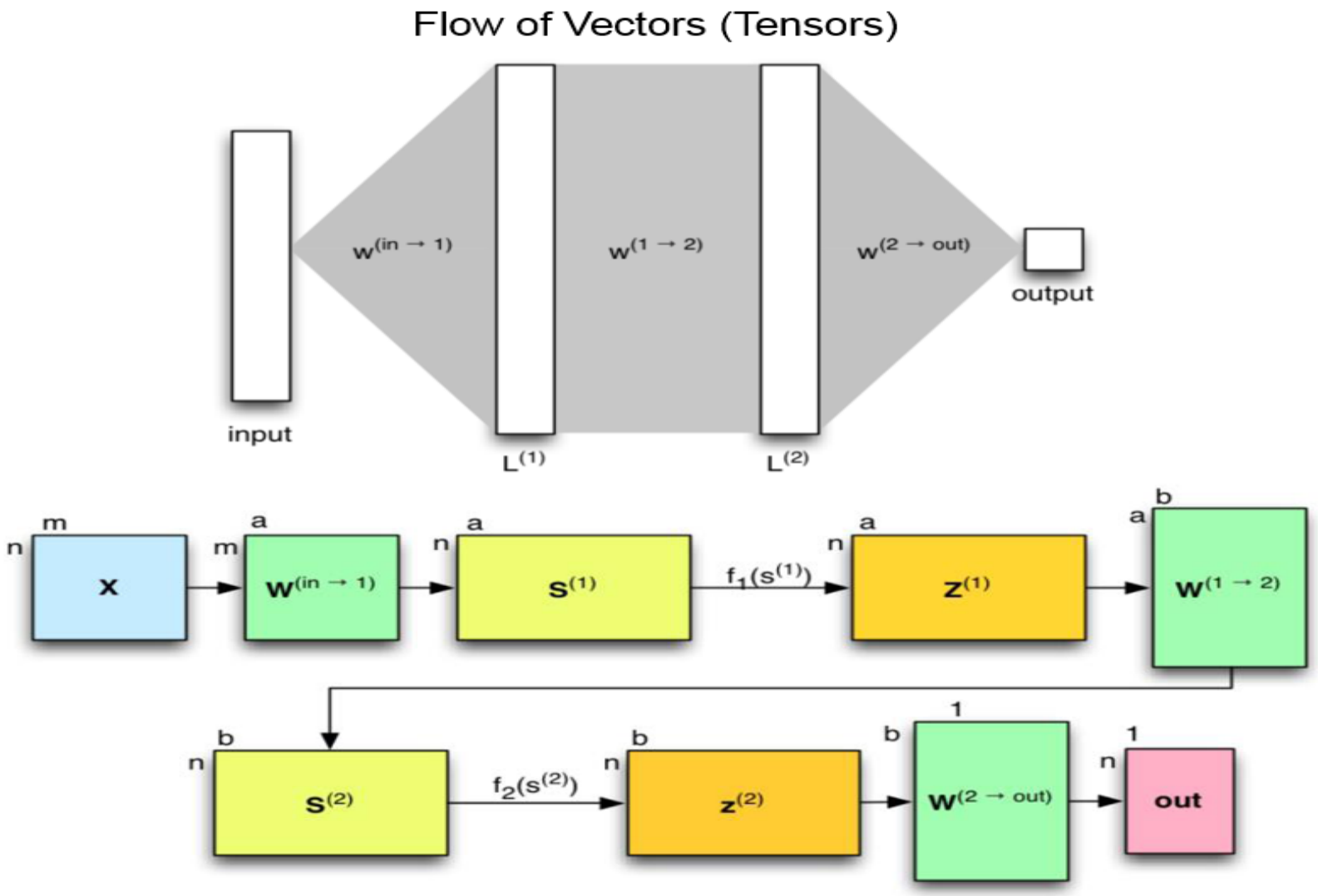
Energy, Feedstock, and Utilities

Deep Learning becoming popular

Neural nets express some output as a nonlinear function (sets of equations) of some inputs

Biological Inspiration, Neurons




Flow of Vectors (Tensors)
input
$w^{(in \to 1)}$
$L^{(1)}$
$w^{(1 \to 2)}$
$L^{(2)}$
$w^{(2 \to out)}$
output
X
$w^{(in \to 1)}$
$s^{(1)}$
$f_1(s^{(1)})$
$z^{(1)}$
$w^{(1 \to 2)}$
$s^{(2)}$
$f_2(s^{(2)})$
$z^{(2)}$
$w^{(2 \to out)}$
out

BASIC TO PYTHON PROGRAM

BASIC TO PYTHON PROGRAM

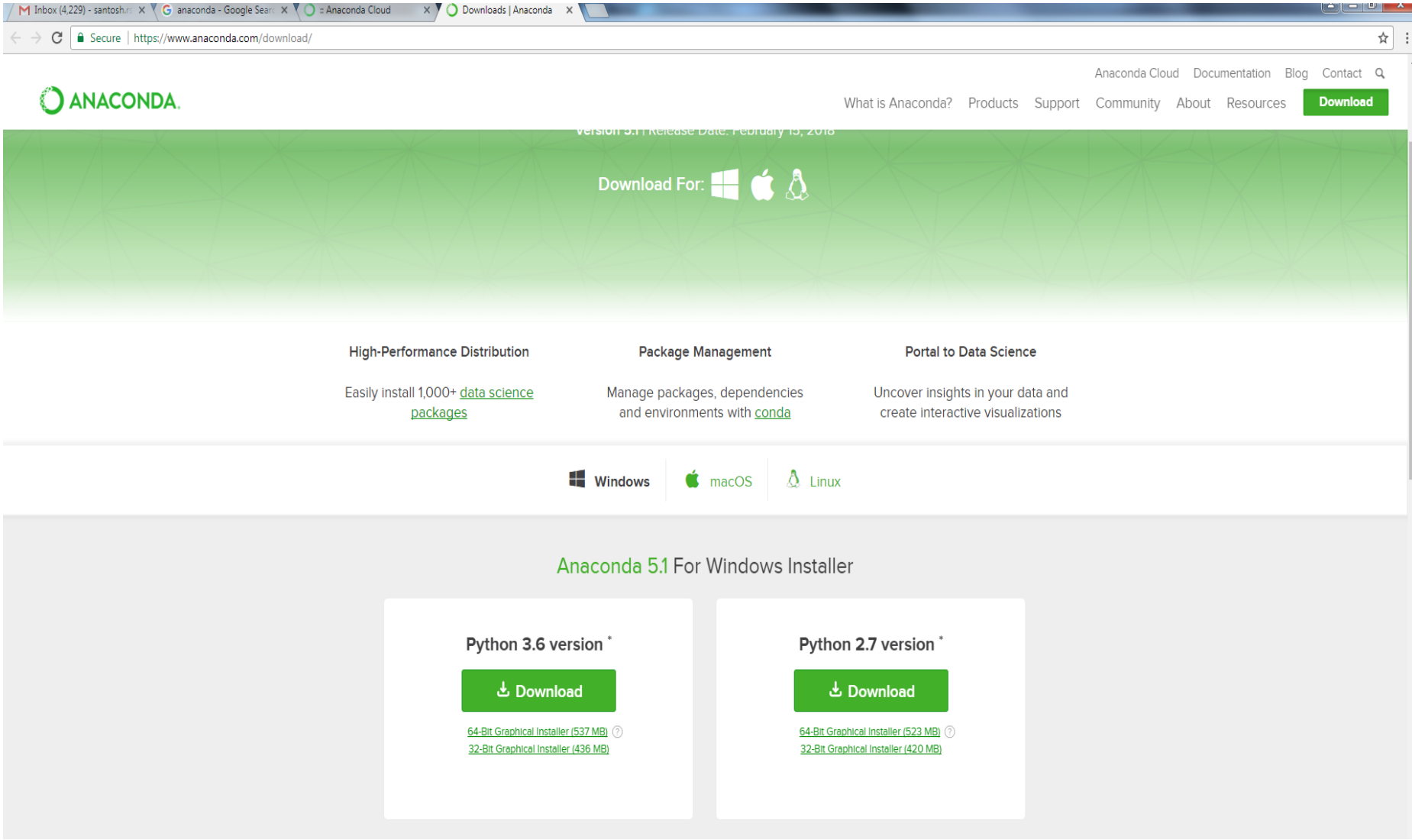


Figure: Anaconda Installation

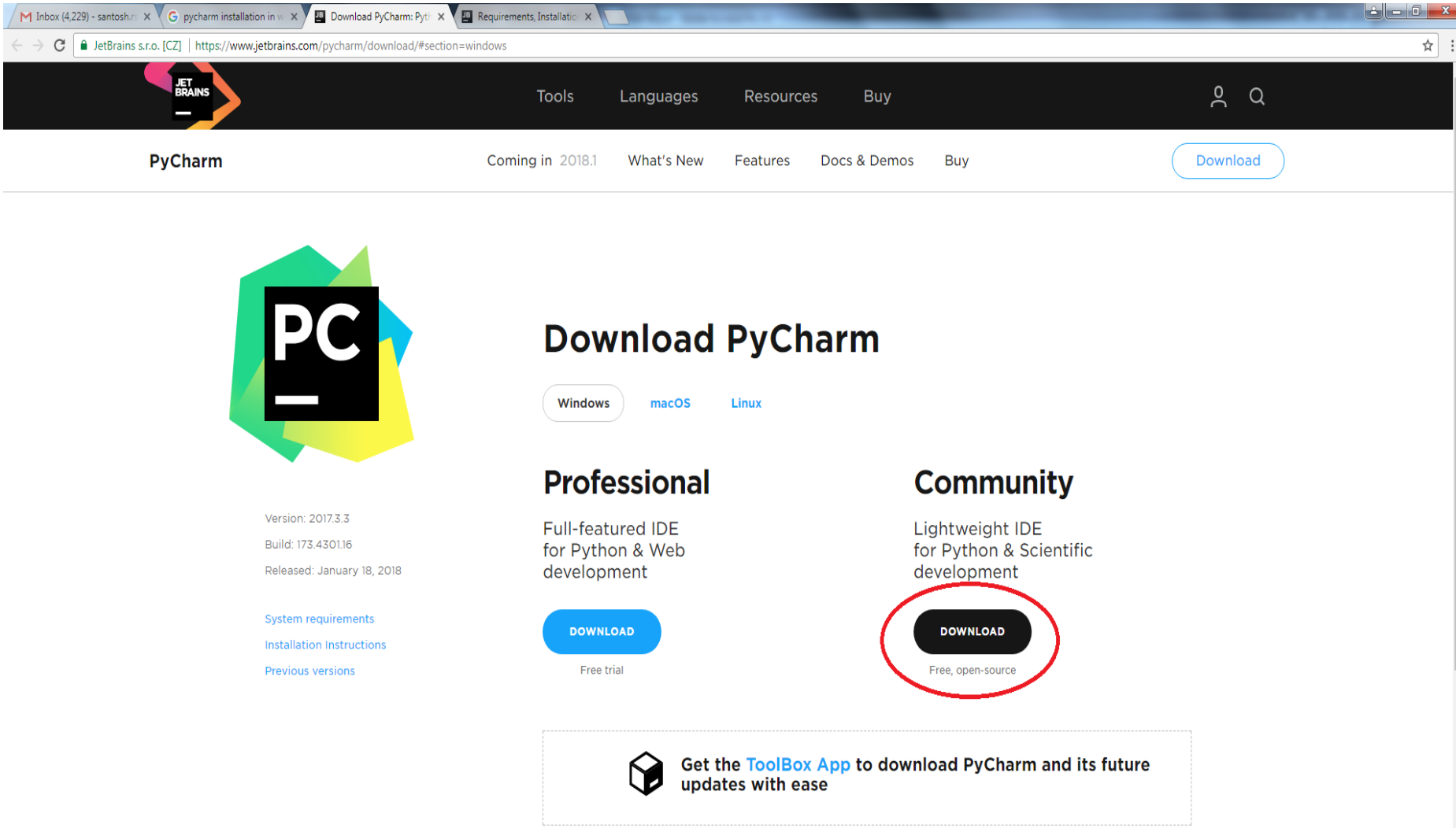


Figure: Pycharm Installation

Welcome to Python 3.6's help utility! If this is your first time using Python, you should definitely check out the tutorial on the Internet at `http://docs.python.org/3.6/tutorial/`. Enter the name of any module, keyword, or topic to get help on writing Python programs and using Python modules. To quit this help utility and return to the interpreter, just type "quit".

>>> *help*()

List and Tuples

A data structure is a collection of data elements (such as numbers or characters—or even other data structures) that is structured in some way, for example, by numbering the elements. Python has six built-in types of sequences, but let's concentrate on two of the most common ones—lists and tuples. The main difference between these is that you can change a list, but you can't change a tuple.

```
>>> animals = ['cat', 'dog']
>>> animals
['cat', 'dog']
>>> animals.append('mat')
>>> animals
['cat', 'dog', 'mat']
>>> animals[2] = 'bat'
>>> animals
['cat', 'dog', 'bat']
```

Tuple is :

```
>>> point = (3,7)
>>> point
(3, 7)
>>> point[1]
7
>>> point[1] = 4
Traceback (most recent call last):
  File "", line 1, in
TypeError: 'tuple' object does not support item assignment
```

Array, String, Insert, Remove

Array

```
a = np.arange(1, 10)
a = np.arange(5)
a = np.arange(2.1, 5.4, 0.1)
a = np.arange(0, 10, 1.5)
b = np.linspace(0, 10, 7)
```

String

```
s = '123'
pie = '3.141592653589' x
 = int(s) + 1 y = float(pie) + 1
 z_bad = int(s) + int(pie)
z_good = int(s) + int(float(pie))
```

The list Function Because strings can't be modified in the same way as lists, often it can be useful to create a list from a string. You can do this with the list function:1

```
>>> list('Hello')
>>> x = [1, 1, 1]
>>> x[1] = 2
>>> x [1, 2, 1]
```

Deleting Elements Deleting elements from a list is easy too; you can simply use the del statement:

```
>>> names = ['Alice', 'Beth', 'Cecil', 'Dee-Dee', 'Earl']
>>> del names[2]
>>> names ['Alice', 'Beth', 'Dee-Dee', 'Earl']
```

Assigning to Slices Slicing is a very powerful feature, and it is made even more powerful by the fact that you can assign to slices:

```
>>> name = list('Perl')
>>> name ['P', 'e', 'r', 'l']
>>> name[2:] = list('ar')
>>> name ['P', 'e', 'a', 'r']
```

append The append method is used to append an object to the end of a list:

```
>>> lst = [1, 2, 3]
>>> lst.append(4)
>>> lst [1, 2, 3, 4]
```

Insert The insert method is used to insert an object into a list:

```
>>> numbers = [1, 2, 3, 5, 6, 7]
>>> numbers.insert(3, 'four')
>>> numbers [1, 2, 3, 'four', 5, 6, 7]
```

pop The pop method removes an element (by default the last one) from the list and returns it:

```
>>> x = [1, 2, 3]
 >>> x.pop()
3
>>> x
[1, 2]
>>> x.pop(0)
1
>>> x
[2]
```

Using pop, you can implement a common data structure called a stack. A stack like this works just like a stack of plates. You can put plates on top, and you can remove plates from the top. The last one you put into the stack is the first one to be removed. (This principle is called Last-In, First-Out, or LIFO.)

remove The remove method is used to remove the first occurrence of a value:

```
>>> x = ['to', 'be', 'or', 'not', 'to', 'be']
>>> x.remove('be')
>>> x ['to', 'or', 'not', 'to', 'be']
```

LOOPs

For Loop

```
for x in range(5):
    print(x)
# Prints out 3,4,5
for x in range(3, 6):
    print(x)
# Prints out 3,5,7
for x in range(3, 8, 2):
    print(x)
```

Time on Small ranges

```
import time
#use time.time() on Linux
start = time.clock()
for x in range(1000):
    pass
stop = time.clock()
print stop-start
start = time.clock()
for x in xrange(1000):
    pass
stop = time.clock()
print stop-start
```

IF-ELSE Statement

```
import numpy as np
from math import *
# If statement
x=float(input('Value: '))
if x==10:
    print('X is a 10')
elif x > 10:
    print ('X is larger than 10')
else :
    print('X is less than 10...')
    print(x)
########################
value = int(input("Please enter an integer in the range 0...5: "))
print('Program starts.............')
if value < 0 :
    print("Too small")
elif value == 0:
     print("zero")
elif value == 1:
    print("one")
elif value == 2:
    print("two")
elif value == 3:
    print("three")
elif value == 4:
    print("four")
elif value == 5:
    print("five")
else:
    print("Too large")
print("Program complete.....................")
```

WHILE Statement

```
# Prints out 0,1,2,3,4 and then it prints "count value reached 5"
import numpy as np
count=0
while(count<5):
    print(count)
    count +=1
else:
    print("count value reached %d" %(count))

# Prints out 1,2,3,4
for i in range(1, 10):
    if(i%5==0):   ### % is the remainder
        break
    print(i)
else:
    print("this is not printed because for loop is terminated because of break "
          "but not due to fail in condition")
```

Define Function and Calling Function

```
import numpy as np
def plus(a,b):
    sum = a + b
    return sum
def mult (a,b):
    mul=a*b
    return mul
def hsqr (a,b):
    dh=a*a+b*b+2*a*b
    return dh
```

```
import numpy as np
from sknfun import plus,mult,hsqr
a=float(input('a:'))
b=float(input('b:'))
sd=plus(a,b)
print('Sum of two numbers..............')
print(sd)
df=mult (a,b)
print('Multiplication of two numbers..............')
print(df)
gh=hsqr(a,b)
print('Square of two numbers..............')
print(gh)
```

First Program

```
import numpy as np
from math import *
'''
This Python file created by Dr. Santosh Kumar Nanda
Date: 20/2/2018
Version 1.1
'''
###### For hexadecimal value
bb=hex(1024)
###### For binary value
vv=bin(1024)
print('.............Hexadecimal value............')
print(bb)
print('.............Binary value............')
print(vv)
#######  Addition ###########
a=float(input('a: '))
b=float(input('b: '))
c=a+b
print('.............Addition of two value............')
print(c)
dd=a-b
print('.............Substraction of two value............')
print(dd)
vs=sqrt(a)
print('............Sqaure Root of a value..........')
print(vs)
##########  Import numpy as mathematic operation ########
from numpy import sqrt, exp
vz=sqrt(2)
vd=exp(3)
print('Square and expontetial from numpy.........')
print('square root.........')
print(vz)
print('expontial.........')
print(vd)
```

Extract Data from Excel and data visualization

```
import numpy as np
import matplotlib.pyplot as plt
from openpyxl import load_workbook

wb = load_workbook('skn_demo.xlsx')
sheet_1=wb.get_sheet_by_name('Sheet1')
xx=np.zeros(sheet_1.max_row)
yy=np.zeros(sheet_1.max_row)
zz=np.zeros(sheet_1.max_row)
for i in range(0, sheet_1.max_row):
    xx[i] =sheet_1.cell(row=i + 1, column=1).value
    yy[i] =sheet_1.cell(row=i + 1, column=2).value

########  Data Visulaization ###############
plt.figure(5)
plt.plot(xx, yy, label='Data-1')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Plot')
plt.legend(loc='best', fontsize='small')
plt.savefig('Xl_demo.png')
plt.show()
```

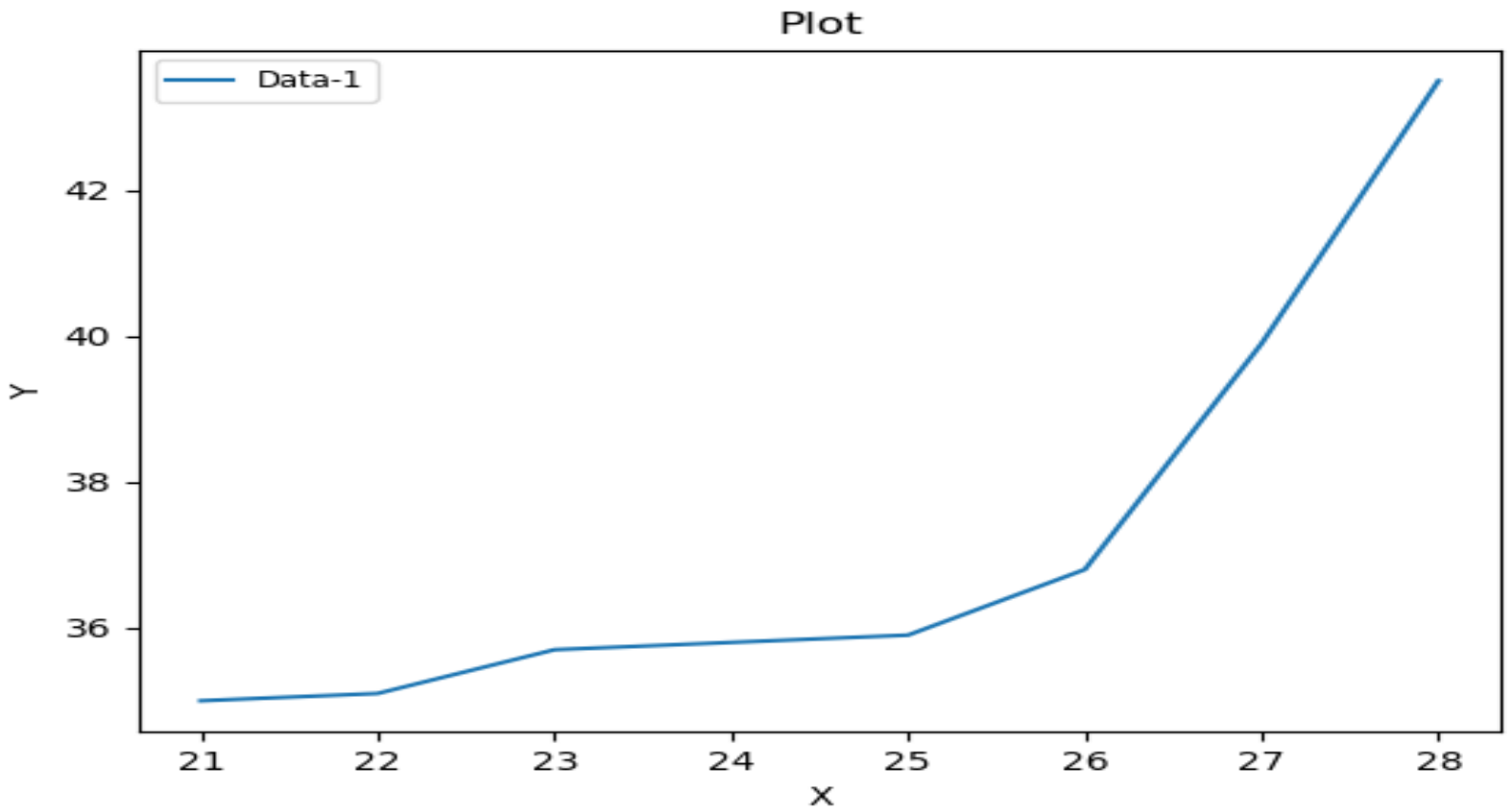


Figure: Data visualization

Create Excel to store the data

```
import openpyxl
wb=openpyxl.Workbook()
sheet=wb.get_active_sheet()
sheet.title='Sheet#1'
#####################
xb=np.linspace(1,10,10)
yb=xb*xb
########## Add title
sheet.cell(row=1, column=1).value='X values'
sheet.cell(row=1, column=2).value='Y values'
######
for i in range (0,len(xb)):
    sheet.cell(row=i+2, column=1).value=xb[i]
    sheet.cell(row=i + 2, column=2).value = yb[i]
#########
wb.save('write_simple.xlsx')
```

VISUALIZATION DATA- GRAPHICS

VISUALIZATION DATA- GRAPHICS

Single Plot

```
import numpy as np
import matplotlib.pyplot as plt
t = np.arange(0.0, 2.0, 0.01)
s = 1 + np.sin(2*np.pi*t)
plt.figure(1)
plt.plot(t, s, color="green", linewidth=1.5, label='Sine-Function')
plt.xlabel('time (s)')
plt.ylabel('voltage (mV)')
plt.title('Plot of Sine-Function')
plt.grid(True)
plt.axis([np.min(t),np.max(t),np.min(s),np.max(s)])
plt.legend (loc='best')
plt.savefig("test1.png")
plt.show()
```

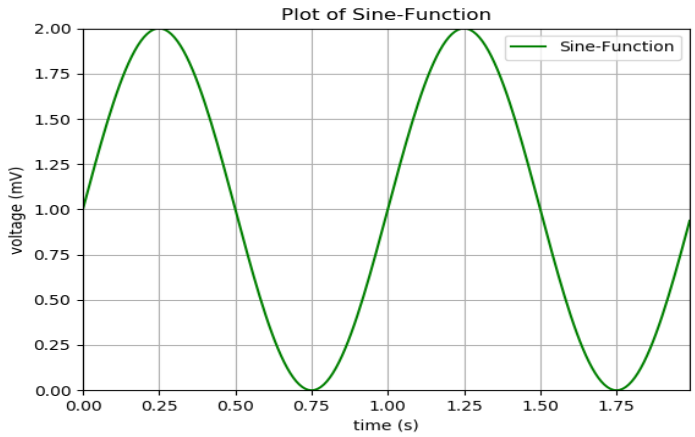


Figure: Data visualization

Multiple plot

```
# simple_plot.py [get code]
import numpy as np, matplotlib.pyplot as plt
t = np.arange(0.0, 2.0, 0.01)
s = 1 + np.sin(2*np.pi*t)
s1 = 1 + np.cos(2*np.pi*t)
plt.figure(1)
plt.plot(t, s, color="green", linewidth=1.5, label='Sine-Function')
plt.plot(t, s1, color="red", linewidth=1.5, label='Cosine-Function')
plt.xlabel('time (s)')
plt.ylabel('voltage (mV)')
plt.title('Plot of Sine-Function')
plt.grid(True)
plt.axis([np.min(t),np.max(t),np.min(s),np.max(s)])
plt.legend (loc='best')
#plt.savefig("test1.pdf")
plt.savefig("test2.png")
plt.show()
```

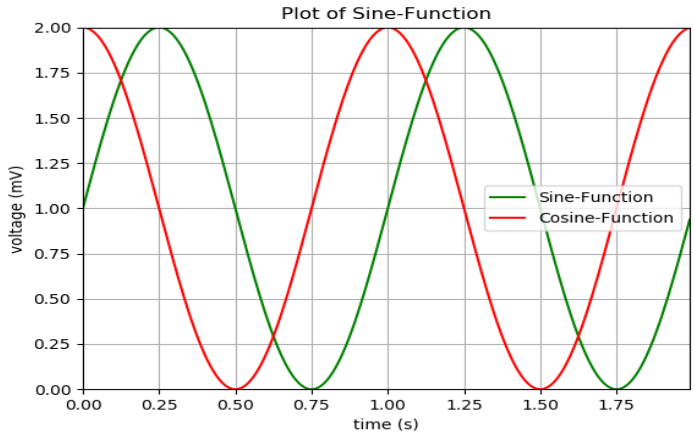


Figure: Data visualization

```
import numpy as np
import matplotlib.pyplot as plt

# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)

# red dashes, blue squares and green triangles
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
plt.savefig("test3.png")
plt.show()
```

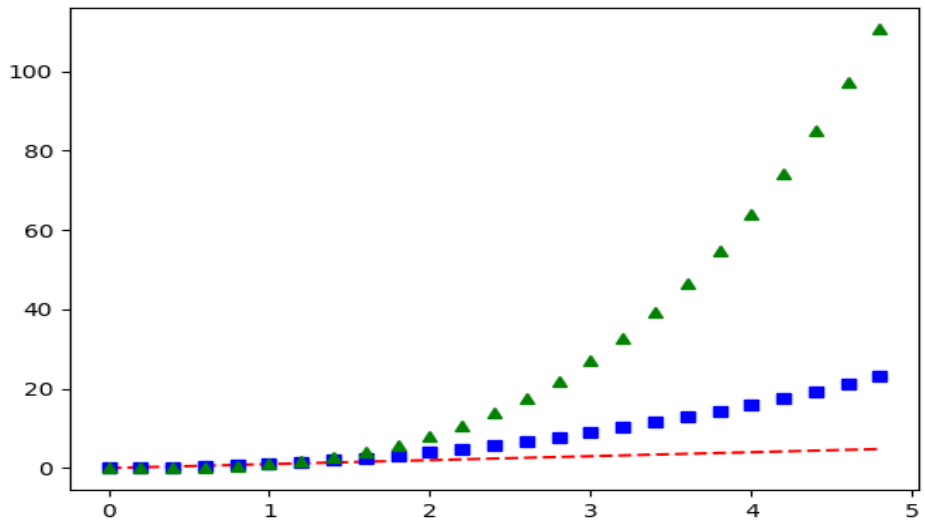


Figure: Data visualization

Subplot

```
import numpy as np
import matplotlib.pyplot as plt
def f(t):
    return np.exp(-t) * np.cos(2*np.pi*t)

t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)

plt.figure(2)
plt.subplot(211)
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')

plt.subplot(212)
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
plt.savefig("test4.png")
plt.show()
```

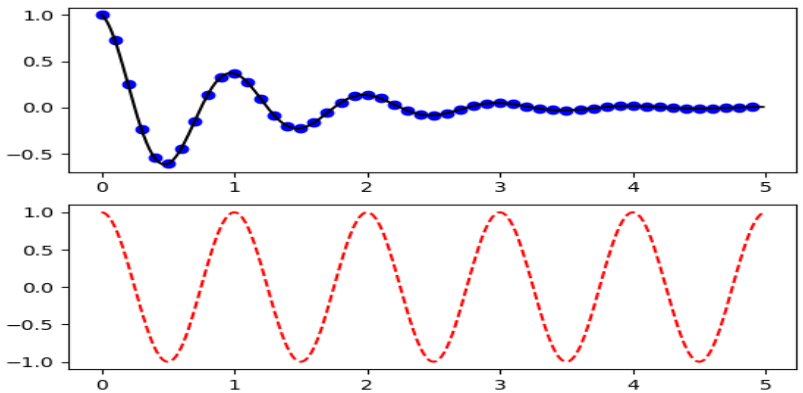


Figure: Data visualization

Surface Plot

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
import numpy as np
fig = plt.figure(3)
ax = fig.gca(projection='3d')
# Make data.
X = np.arange(-5, 5, 0.25)
Y = np.arange(-5, 5, 0.25)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R)
# Plot the surface.
surf = ax.plot_surface(X, Y, Z, cmap=cm.coolwarm,
                       linewidth=0, antialiased=False)
# Customize the z axis.
ax.set_zlim(-1.01, 1.01)
ax.zaxis.set_major_locator(LinearLocator(10))
ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))

# Add a color bar which maps values to colors.
fig.colorbar(surf, shrink=0.5, aspect=5)
ax.set_xlabel('X Label')
ax.set_ylabel('Y Label')
ax.set_zlabel('Z Label')
plt.savefig("test5.png")

plt.show()
```

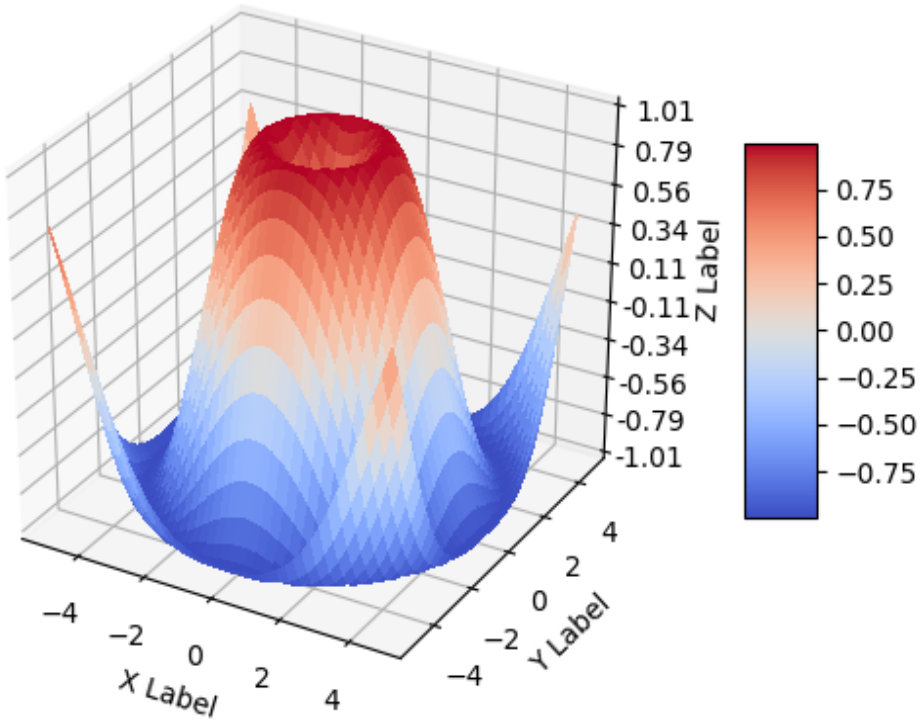


Figure: Data visualization–Surface Plot

Projectile Motion

```
# projectile.py
# Dr. S.K.Nanda
"""
Calculate how long an object is in the air when thrown from a specified height
with a range of initial speeds assuming constant acceleration due to gravity:
    0.5 * g * t**2 - v0 * t - y0 = 0
"""
import numpy as np

#%% Initialization of variables.
initial_speed = 0.0         # v0 = initial vertical speed of ball in [m/s]
impact_time = 0.0           # t = time of impact in [s] (computed in loop)

#%% Initialization of parameters.
g = 9.8066                  # gravitational acceleration in [m/s^2]
initial_height = 2.0        # y0 = height ball is thrown from in [m]
speed_increment = 5.0       # how much to increase speed in [m/s] for each iteration
cutoff_time = 10.0          # stop computing after impact time exceeds cutoff

#%% Calculate and display impact time.  Increment initial speed each step.
#   Repeat until impact time exceeds cutoff.
while impact_time < cutoff_time:
    # Use quadratic equation to solve kinematic equation for impact time:
    impact_time = (np.sqrt(initial_speed**2 + 2 * g * initial_height) + initial_speed) / g
    print("speed= {} m/s; time= {:.1f} s".format(initial_speed, impact_time))
    initial_speed += speed_increment
print("Calculation complete.")
```

Machine Learning

Machine Learning

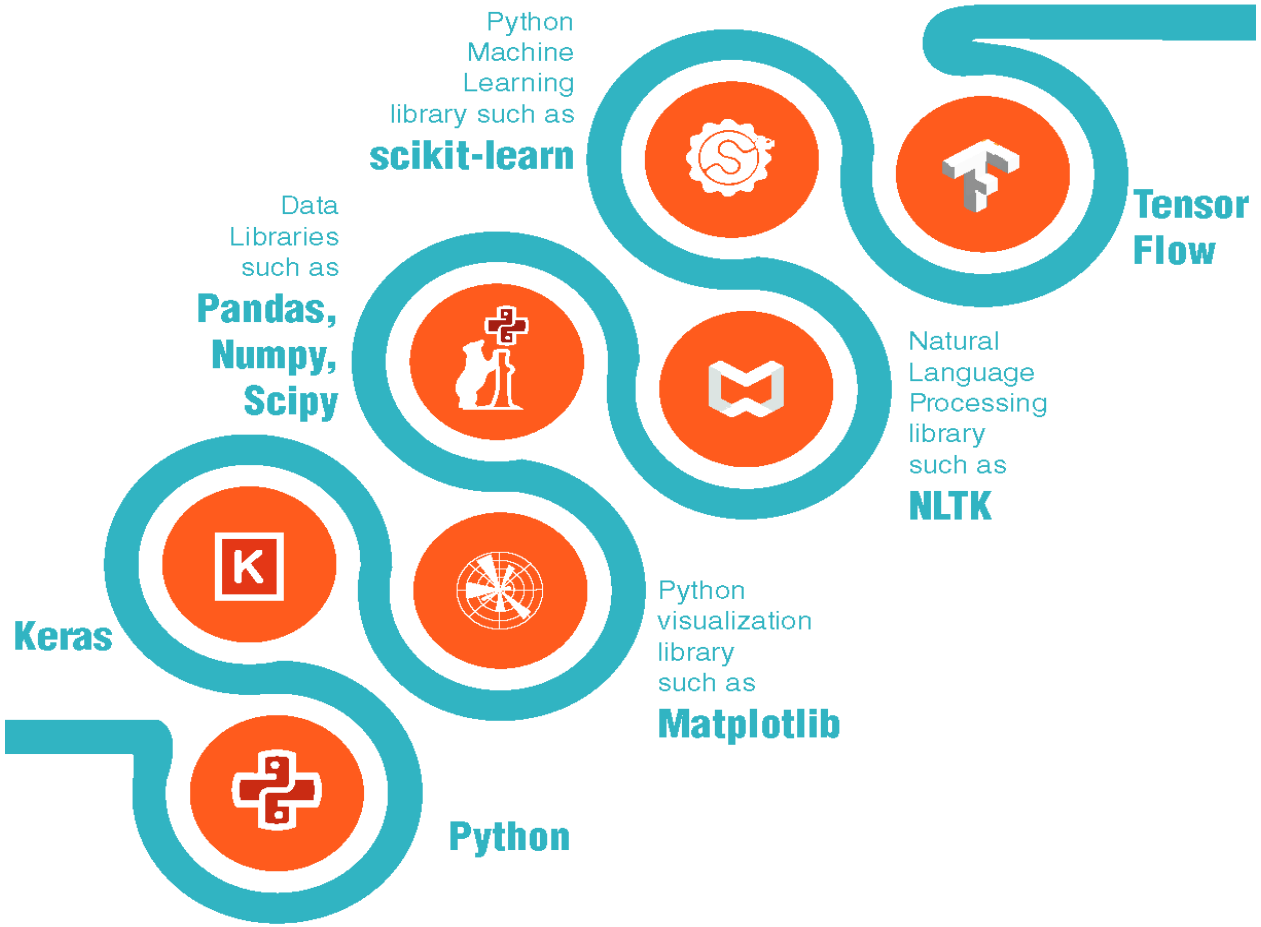


Figure: Machine Learning Tool

- supervised learning, in which the data comes with additional attributes that we want to predict (Click here to go to the scikit-learn supervised learning page).This problem can be either:
  - classification: samples belong to two or more classes and we want to learn from already labeled data how to predict the class of unlabeled data. An example of classification problem would be the handwritten digit recognition example, in which the aim is to assign each input vector to one of a finite number of discrete categories.
  - regression: if the desired output consists of one or more continuous variables, then the task is called regression. An example of a regression problem would be the prediction of the length of a salmon as a function of its age and weight.
- unsupervised learning, in which the training data consists of a set of input vectors x without any corresponding target values. The goal in such problems may be to discover groups of similar examples within the data, where it is called clustering, or to determine the distribution of data within the input space, known as density estimation.
- Reinforcement Learning Method

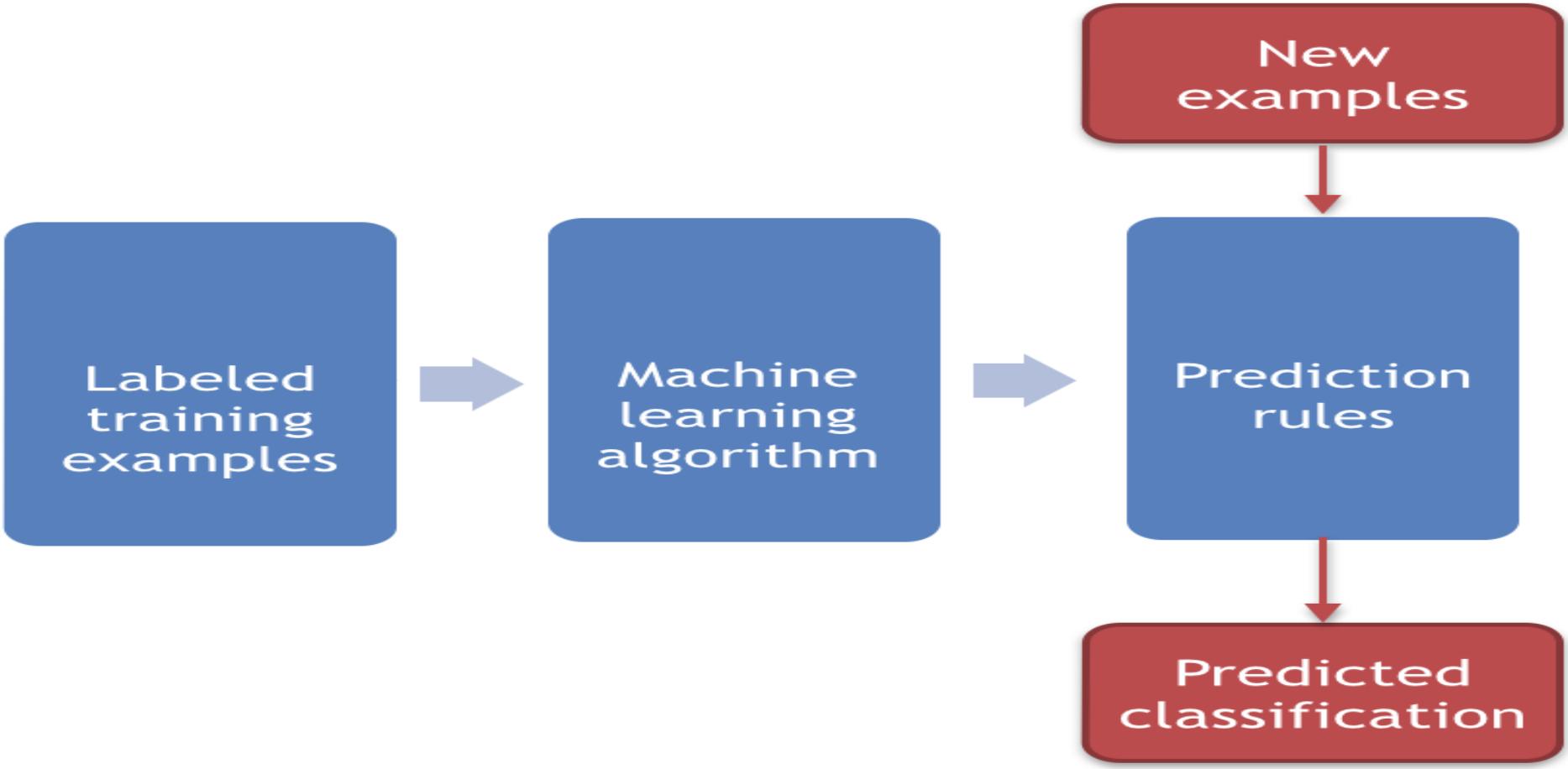


Figure: Typical supervised learning problem

Scikit-learn provides a rich environment with state of the art implementations of many well known machine learning algorithms, while maintaining an easy to use interface tightly integrated with the Python language. To start with scikit-learn, visit `http://scikit-learn.org/stable/index.html` and click on examples. You will find all the gallery of machine learning at `http://scikit-learn.org/stable/autoexamples/index.html`. Then, to go further into classification, go to the home page `http://scikit-learn.org/stable/supervisedlearning.htmlsupervised-learning` , then click on decision. Scikit-learn depends on two other Python packages, NumPy and SciPy. For plotting and interactive development, you should also install matplotlib, IPython and the Jupyter notebook. We recommend using one of the following pre-packaged Python distributions, which will provide the necessary packages:

- Anaconda (`https://store.continuum.io/cshop/anaconda/`): a Python distribution made for large-scale data processing, predictive analytics, and scientific computing. Anaconda comes with NumPy, SciPy, matplotlib, IPython, Jupyter notebooks, and scikit-learn. Anaconda is available on Mac OS X, Windows, and Linux.

```
# use numpy own format for one matrix
import numpy as np
import scipy.io
data=np.random.randn(2,1000)
np.save('pop.npy', data)
data1 = np.load('pop.npy')
# use text format for one matrix
np.savetxt('pop2.txt', data)
data2 = np.loadtxt('pop2.txt')
# use matlab format for several matrices ({'var1':matrix1,'var2',...})
scipy.io.savemat('pop3.mat',{'data':data})
data3=scipy.io.loadmat('pop3.mat')
```

Time Series Plot

```
import numpy
from numpy.random import randn
import matplotlib.pyplot as plt
npoints=100
x=[i for i in range(npoints)]
y1 = randn(npoints)
y2 = randn(npoints)
y3 = randn(npoints)
y4 = randn(npoints)
plt.subplot(2,1,1)
plt.plot(x,y1)
plt.plot(x,y2,':')
plt.grid()
plt.xlabel('time')
plt.ylabel('series 1 and 2')
plt.axis('tight')
plt.subplot(2,1,2)
plt.plot(x,y3)
plt.plot(x,y4,'r')
plt.grid()
plt.xlabel('time')
plt.ylabel('series 3 and 4')
plt.axis('tight')
```

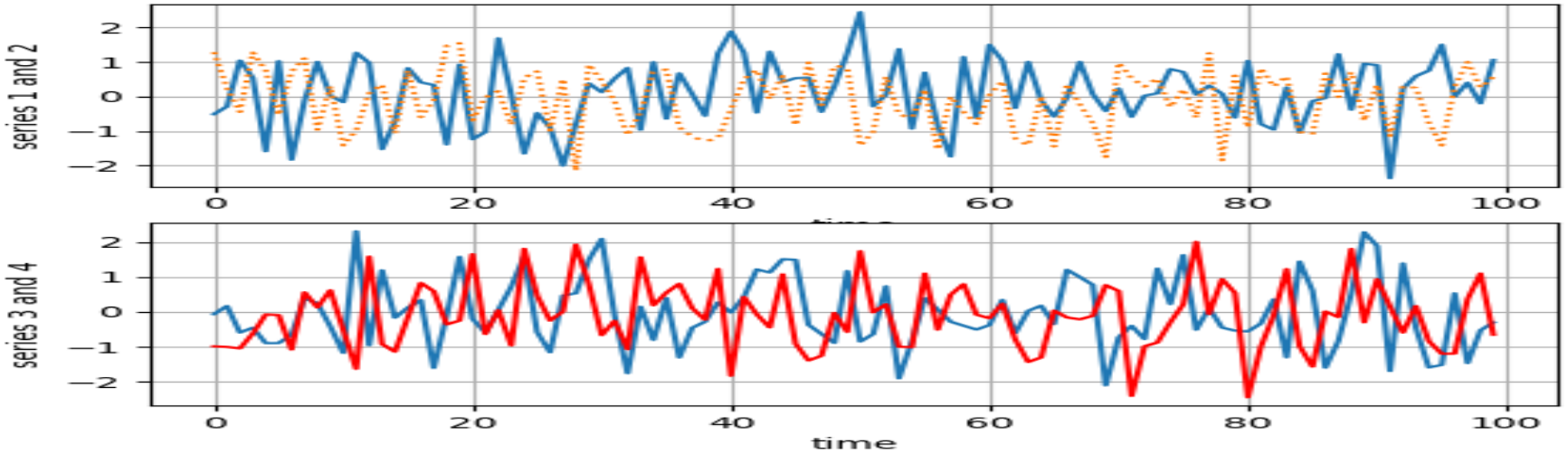


Figure: Time Series Plot

```
import matplotlib.pyplot as plt
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes = [15, 30, 45, 10]
colors = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral']
explode = (0, 0.1, 0, 0)
plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%',
        shadow=True, startangle=90)
plt.axis('equal')
plt.savefig('skn2.png')
plt.show()
```

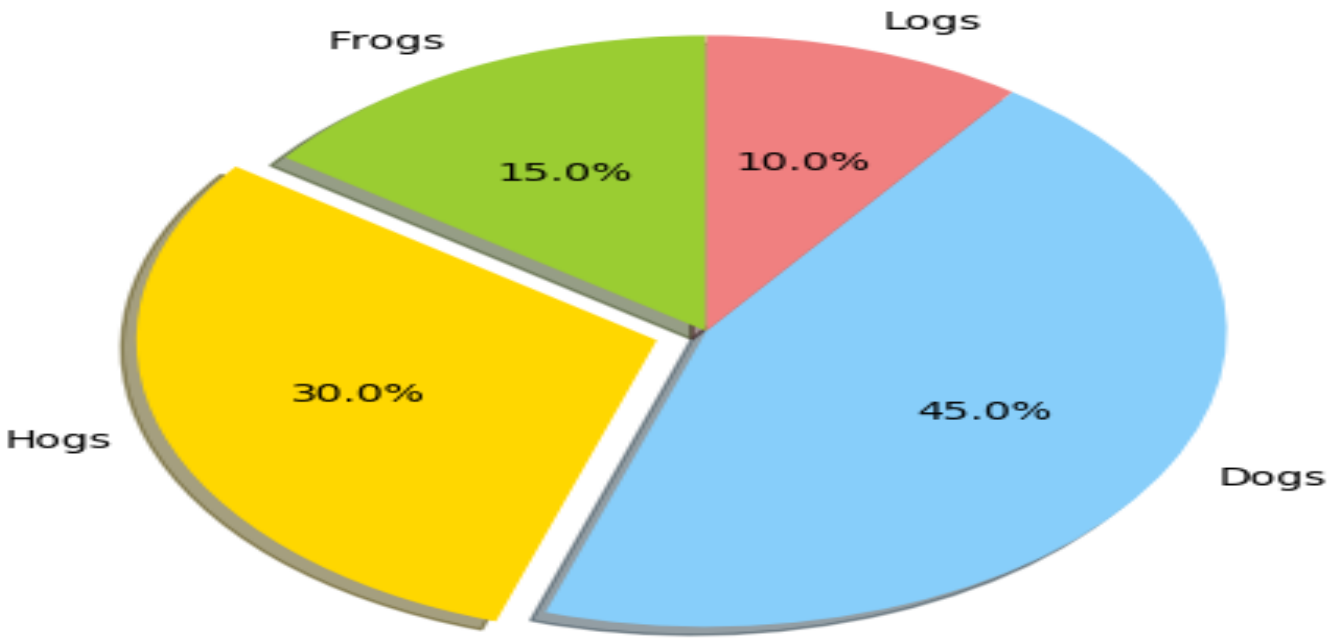


Figure: Pi-Chart

```
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
iris = load_iris()
iris.keys()
print(iris['DESCR'][:193] + "\n...")
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(iris['data'], iris['target'],
random_state=0)
fig, ax = plt.subplots(3, 3, figsize=(15, 15))
plt.suptitle("iris_pairplot")
for i in range(3):
    for j in range(3):
        ax[i, j].scatter(X_train[:, j], X_train[:, i + 1], c=y_train, s=60)
        ax[i, j].set_xticks(())
        ax[i, j].set_yticks(())
        if i == 2:
            ax[i, j].set_xlabel(iris['feature_names'][j])
        if j == 0:
            ax[i, j].set_ylabel(iris['feature_names'][i + 1])
        if j > i:
            ax[i, j].set_visible(False)

plt.savefig('sknirs.png')

plt.show()
```

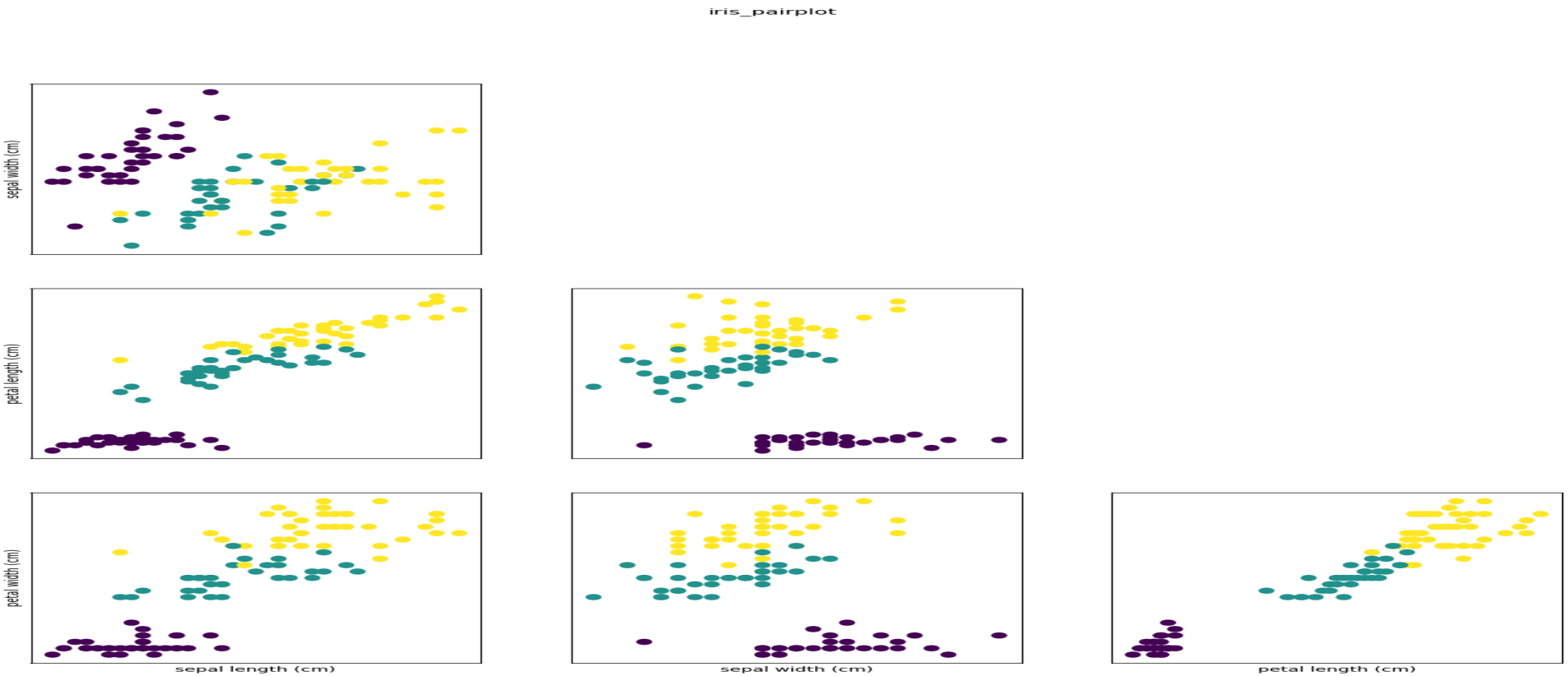


Figure: Irris data

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
iris = load_iris()
iris.keys()
knn = KNeighborsClassifier(n_neighbors=1)
X_train, X_test, y_train, y_test = train_test_split(iris['data'], iris['target'],
random_state=0)
knn.fit(X_train, y_train)
pp=knn.score(X_test, y_test)
print(pp)
```

```
from sklearn.datasets import load_breast_cancer
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neighbors import KNeighborsRegressor
from sklearn import neighbors
import numpy as np
cancer = load_breast_cancer()
cancer.keys()
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
cancer.data, cancer.target, stratify=cancer.target, random_state=66)
training_accuracy = []
test_accuracy = []
# try n_neighbors from 1 to 10.
neighbors_settings = range(1, 11)
for n_neighbors in neighbors_settings:
# build the model
    clf = KNeighborsClassifier(n_neighbors=n_neighbors)
    clf.fit(X_train, y_train)
# record training set accuracy
    training_accuracy.append(clf.score(X_train, y_train))
# record generalization accuracy
    test_accuracy.append(clf.score(X_test, y_test))
plt.plot(neighbors_settings, training_accuracy, label="training accuracy")
plt.plot(neighbors_settings, test_accuracy, label="test accuracy")
plt.legend()
plt.show()

########################  Nearest Neighbor Regression ####################
np.random.seed(0)
X = np.sort(5 * np.random.rand(40, 1), axis=0)
T = np.linspace(0, 5, 500)[:, np.newaxis]
y = np.sin(X).ravel()

# Add noise to targets
y[::5] += 1 * (0.5 - np.random.rand(8))
```

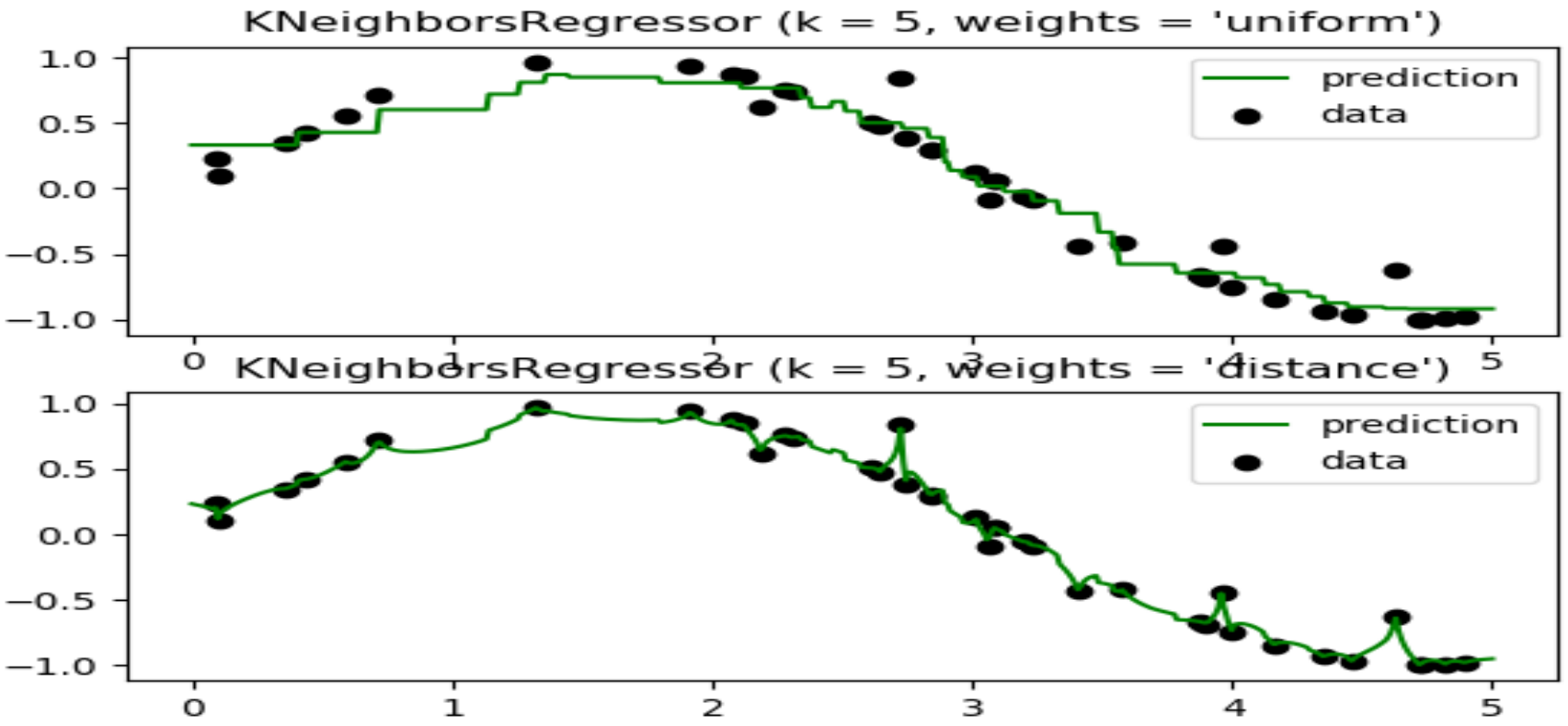


Figure: Irris data

Neural Network

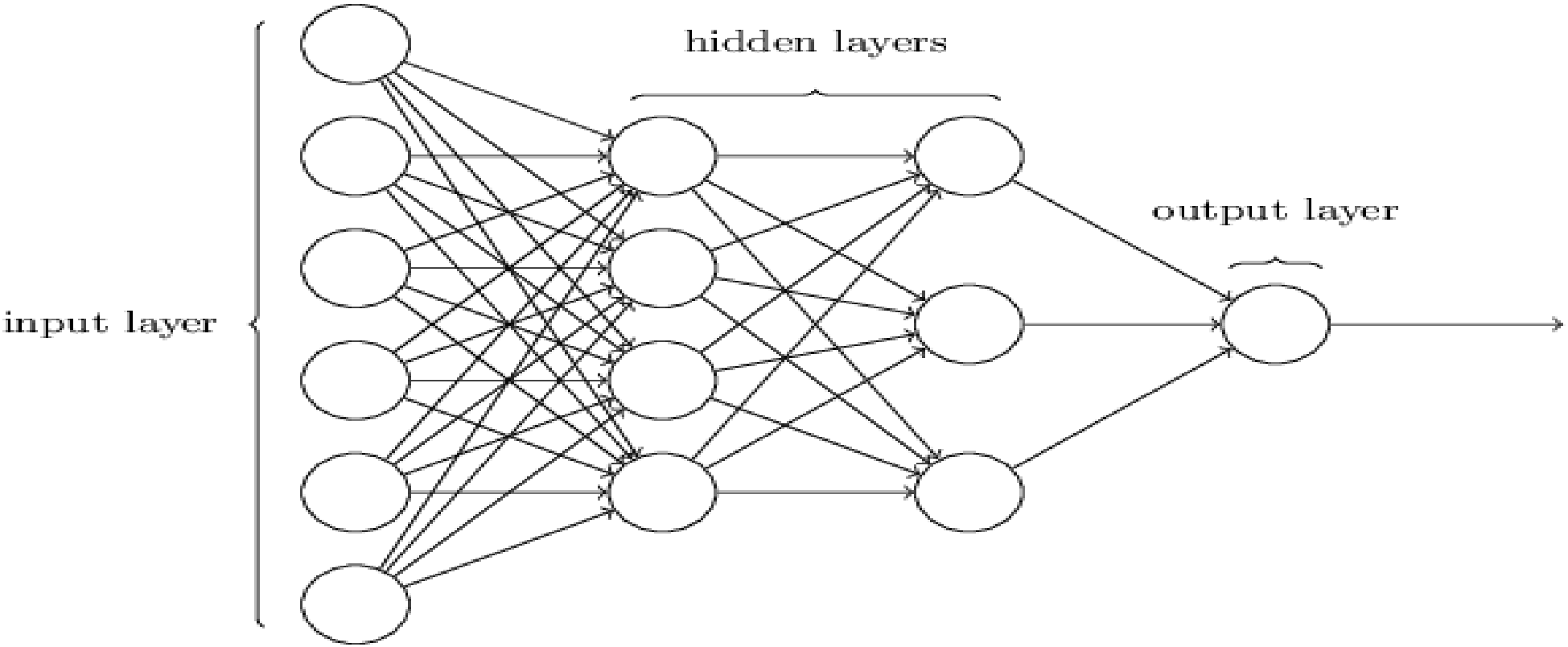


Figure: MLP

```
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import MinMaxScaler
import numpy as np
from sklearn.model_selection import train_test_split

MLPClassifier(activation='relu', alpha=1e-05, batch_size='auto',
beta_1=0.9, beta_2=0.999, early_stopping=False,
epsilon=1e-08, hidden_layer_sizes=(5, 2), learning_rate='constant',
learning_rate_init=0.001, max_iter=200, momentum=0.9,
nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True,
solver='lbfgs', tol=0.0001, validation_fraction=0.1, verbose=False,
warm_start=False)

from sklearn.datasets import make_moons

X, y = make_moons(n_samples=100, noise=0.25, random_state=3)
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, random_state=42)
mlp = MLPClassifier(solver='lbfgs', alpha=1e-5,hidden_layer_sizes=(15,), random_state=1).fit(X_train, y_train)


plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train, s=60)
plt.show()
```

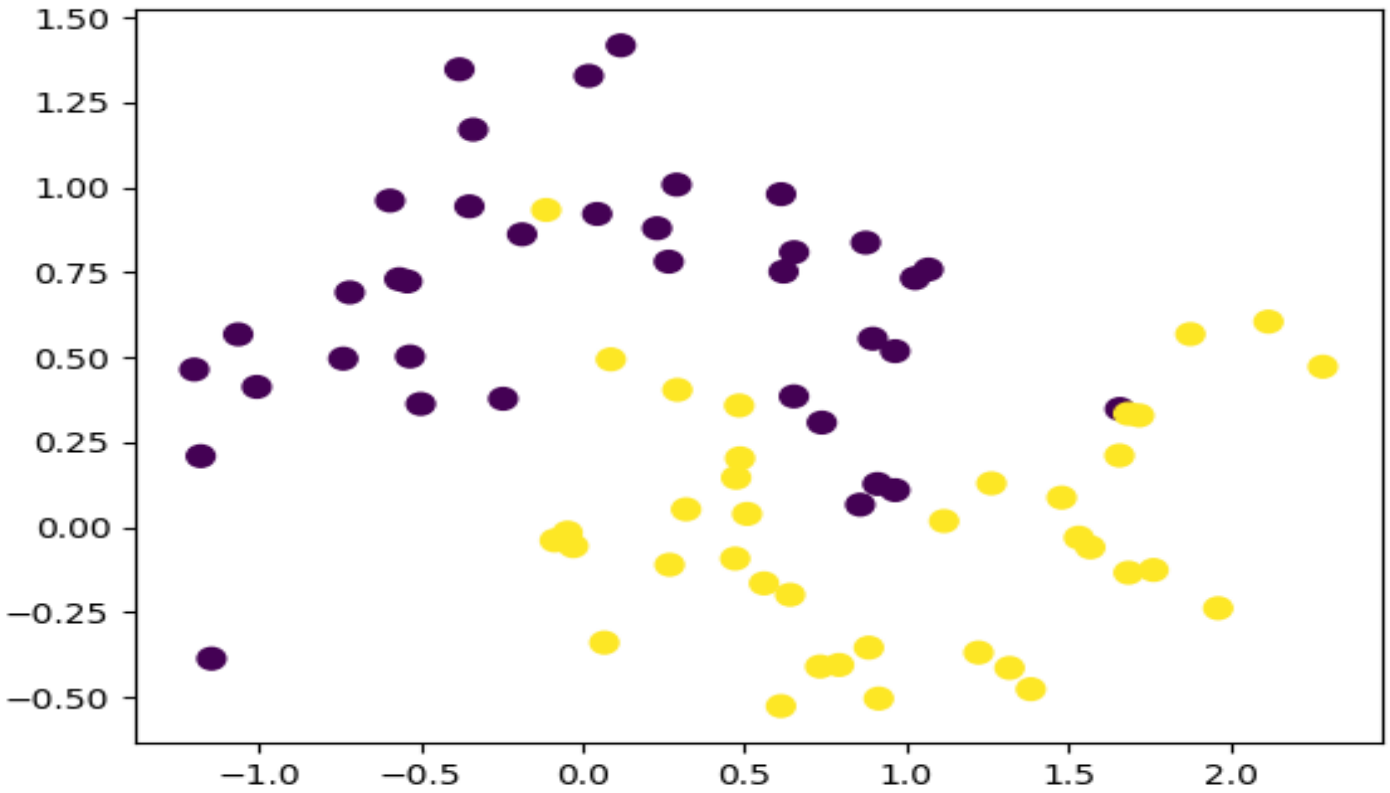


Figure: MLP

```
from sklearn.cluster import KMeans
from sklearn import datasets
import matplotlib.pyplot as plt
from pylab import *
iris = datasets.load_iris()
X, y = iris.data, iris.target
k_means = KMeans(n_clusters=3, random_state=0) # Fixing the RNG in kmeans
k_means.fit(X)
y_pred = k_means.predict(X)
plt.scatter(X[:, 0], X[:, 1], c=y_pred)
plt.savefig("kmean1.png")
plt.xlabel('Samples')
plt.ylabel('Data')
plt.title('Kmean Cluster')
plt.show()
```

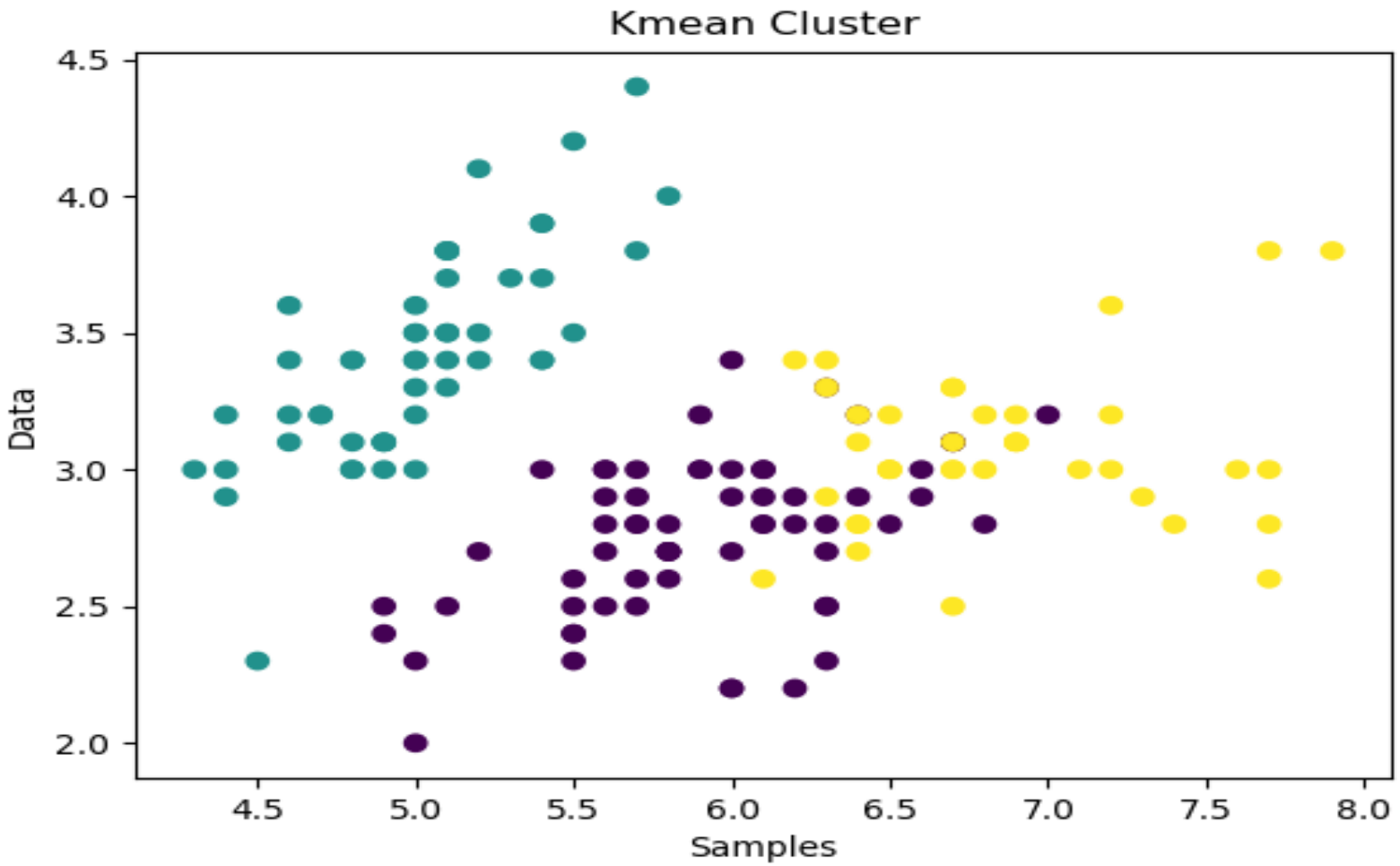
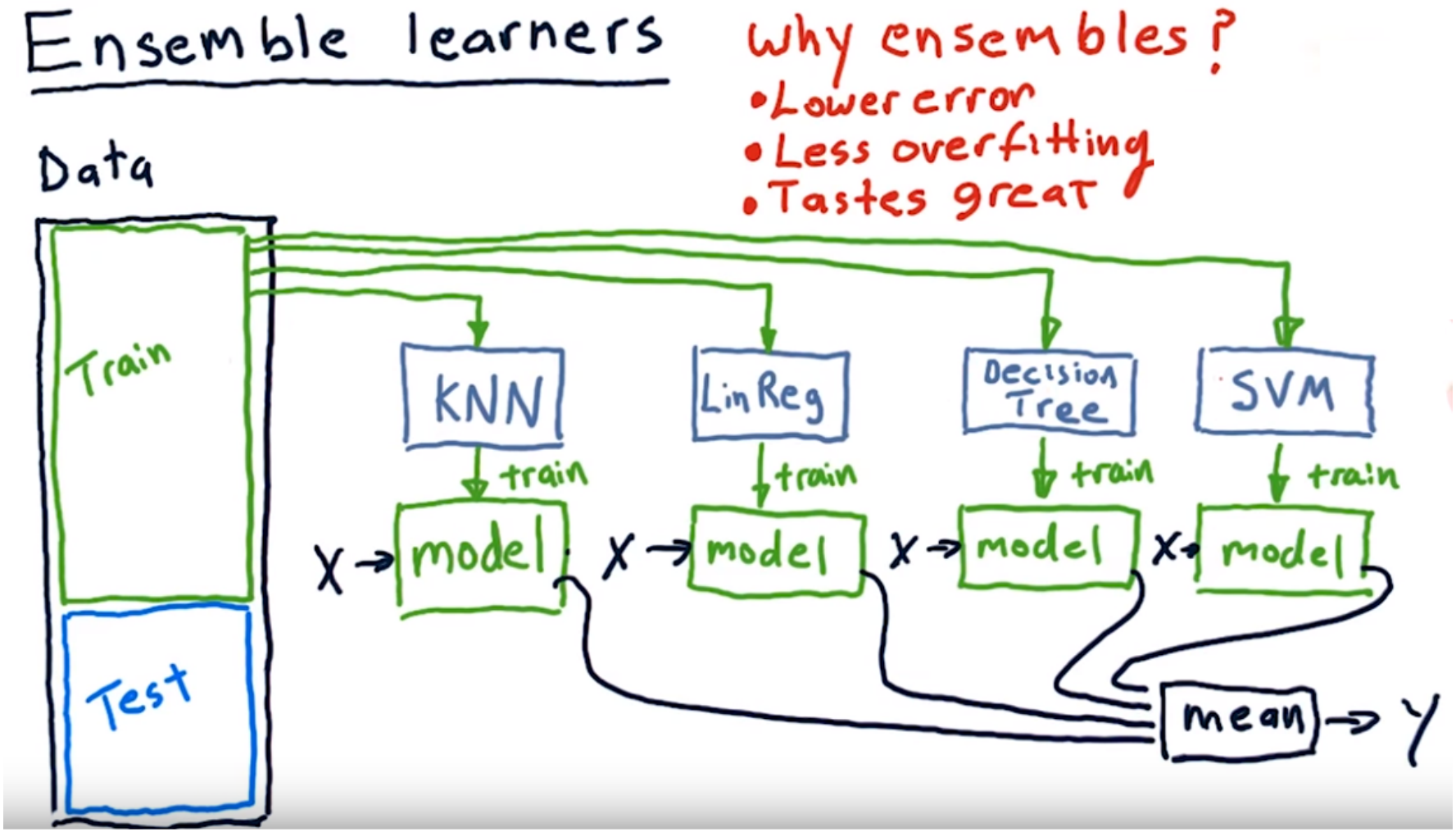


Figure: K-mean

Ensemble Machine Learning



Ensemble Machine Learning

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn. ensemble import RandomForestClassifier, BaggingClassifier, AdaBoostClassifier, VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
data = pd.read_csv('mnist.csv')
df_x = data.iloc[:,1:]
df_y = data.iloc[:,0]
x_train, x_test, y_train, y_test = train_test_split(df_x, df_y, test_size=0.2, random_state=4)
#descision tree
dt = DecisionTreeClassifier()
dt.fit(x_train,y_train)
dt.score(x_test,y_test)
dt.score(x_train,y_train)
#Random Forest - Ensemble of Descision Trees
rf = RandomForestClassifier(n_estimators=20)
rf.fit(x_train,y_train)
rf.score(x_test,y_test)
#Bagging
bg = BaggingClassifier(DecisionTreeClassifier(), max_samples= 0.5, max_features = 1.0, n_estimators = 20)
bg.fit(x_train,y_train)
bg.score(x_test,y_test)
bg.score(x_train,y_train)
#Boosting - Ada Boost
adb = AdaBoostClassifier(DecisionTreeClassifier(),n_estimators = 5, learning_rate = 1)
adb.fit(x_train,y_train)
adb.score(x_test,y_test)
adb.score(x_train,y_train)
# Voting Classifier - Multiple Model Ensemble
lr = LogisticRegression()
dt = DecisionTreeClassifier()
svm = SVC(kernel = 'poly', degree = 2 )
```

```
evc = VotingClassifier( estimators= [('lr',lr),('dt',dt),('svm',svm)], voting = 'hard')
evc.fit(x_train.iloc[1:4000],y_train.iloc[1:4000])
evc.score(x_test, y_test)
```


THANK
YOU