**PROJECT IN MACHINE LEARNING**

**ON SUPERVISED**

**AND**

**UNSUPERVISED LEARNING**

NAME USN

ANKIT ANAND 1MS16IS012

DEEPIKA G 1MS16IS021  
M SAHIL 1MS16IS040

DATE:03.05.2019

SEM: 6TH

CONTENTS

1. MACHINE LEARNING

2. SUPERVISED LEARNING

-ALGORITHM

-IMPLEMENTATION

-RESULTS

3. UNSUPERVISED LEARNING

-ALGORITHM

-IMPLEMENTATION

-RESULTS

4. CONCLUSION

MACHINE LEARNING

**Machine learning** (ML) is the [scientific study](https://nam05.safelinks.protection.outlook.com/?url=https%3A%2F%2Fen.m.wikipedia.org%2Fwiki%2FBranches_of_science&data=02%7C01%7CSunny.anand%40conduent.com%7Caa9a279c30324c190b6f08d6cde7f2af%7C1aed4588b8ce43a8a775989538fd30d8%7C0%7C0%7C636922793425023961&sdata=AUTb5QUH7ZEBIcLDXE7Lvk%2F4P45rxHKCYhB6lkMg1Ko%3D&reserved=0) of [algorithms](https://nam05.safelinks.protection.outlook.com/?url=https%3A%2F%2Fen.m.wikipedia.org%2Fwiki%2FAlgorithm&data=02%7C01%7CSunny.anand%40conduent.com%7Caa9a279c30324c190b6f08d6cde7f2af%7C1aed4588b8ce43a8a775989538fd30d8%7C0%7C0%7C636922793425033969&sdata=CipnPdsvPlaO2gPb%2Fv%2Ft2YZkYLK%2Fw9reHsX64mtMzvc%3D&reserved=0) and [statistical models](https://nam05.safelinks.protection.outlook.com/?url=https%3A%2F%2Fen.m.wikipedia.org%2Fwiki%2FStatistical_model&data=02%7C01%7CSunny.anand%40conduent.com%7Caa9a279c30324c190b6f08d6cde7f2af%7C1aed4588b8ce43a8a775989538fd30d8%7C0%7C0%7C636922793425033969&sdata=LHqXgwwO3idNmjXJoTQGyA9qltSzQ6YzX20vgJ%2BDk4A%3D&reserved=0) that [computer systems](https://nam05.safelinks.protection.outlook.com/?url=https%3A%2F%2Fen.m.wikipedia.org%2Fwiki%2FComputer_systems&data=02%7C01%7CSunny.anand%40conduent.com%7Caa9a279c30324c190b6f08d6cde7f2af%7C1aed4588b8ce43a8a775989538fd30d8%7C0%7C0%7C636922793425043978&sdata=2iWufcGsiJQ1oE5LtRuG2WkPbofyfvqlBtvkcA%2BY3zY%3D&reserved=0) use to effectively perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of [artificial intelligence](https://nam05.safelinks.protection.outlook.com/?url=https%3A%2F%2Fen.m.wikipedia.org%2Fwiki%2FArtificial_intelligence&data=02%7C01%7CSunny.anand%40conduent.com%7Caa9a279c30324c190b6f08d6cde7f2af%7C1aed4588b8ce43a8a775989538fd30d8%7C0%7C0%7C636922793425043978&sdata=D9XhHcU%2FizDV5qgfADV73cK%2FwyAYbFK3v%2F%2FuEetCRWI%3D&reserved=0). Machine learning algorithms build a [mathematical model](https://nam05.safelinks.protection.outlook.com/?url=https%3A%2F%2Fen.m.wikipedia.org%2Fwiki%2FMathematical_model&data=02%7C01%7CSunny.anand%40conduent.com%7Caa9a279c30324c190b6f08d6cde7f2af%7C1aed4588b8ce43a8a775989538fd30d8%7C0%7C0%7C636922793425053986&sdata=HHt1KbJd6owBsZPqz%2BIehXsZwmP73sZNQU5ocQQd1Uo%3D&reserved=0) based on sample data, known as "[training data](https://nam05.safelinks.protection.outlook.com/?url=https%3A%2F%2Fen.m.wikipedia.org%2Fwiki%2FTraining_data&data=02%7C01%7CSunny.anand%40conduent.com%7Caa9a279c30324c190b6f08d6cde7f2af%7C1aed4588b8ce43a8a775989538fd30d8%7C0%7C0%7C636922793425053986&sdata=KRkOP8%2BbJ2uWGhPQH%2F2S51Ua5ngr01KuFwyNAtzx9b4%3D&reserved=0)", in order to make predictions or decisions without being explicitly programmed to perform the task.[[1]](https://nam05.safelinks.protection.outlook.com/?url=https%3A%2F%2Fen.m.wikipedia.org%2Fwiki%2FMachine_learning%23cite_note-1&data=02%7C01%7CSunny.anand%40conduent.com%7Caa9a279c30324c190b6f08d6cde7f2af%7C1aed4588b8ce43a8a775989538fd30d8%7C0%7C0%7C636922793425063999&sdata=2mvuh%2FuXKYUHELckBbbF2klchALbkkVgMW2gnpQArfM%3D&reserved=0)[[2]](https://nam05.safelinks.protection.outlook.com/?url=https%3A%2F%2Fen.m.wikipedia.org%2Fwiki%2FMachine_learning%23cite_note-bishop2006-2&data=02%7C01%7CSunny.anand%40conduent.com%7Caa9a279c30324c190b6f08d6cde7f2af%7C1aed4588b8ce43a8a775989538fd30d8%7C0%7C0%7C636922793425074011&sdata=4YULs6ppRQWzZINI2dtqmT8%2FPSJYhY6dL8yY3mZrqAc%3D&reserved=0):2 Machine learning algorithms are used in a wide variety of applications, such as [email filtering](https://nam05.safelinks.protection.outlook.com/?url=https%3A%2F%2Fen.m.wikipedia.org%2Fwiki%2FEmail_filtering&data=02%7C01%7CSunny.anand%40conduent.com%7Caa9a279c30324c190b6f08d6cde7f2af%7C1aed4588b8ce43a8a775989538fd30d8%7C0%7C0%7C636922793425074011&sdata=3OelI1O5phE69L9CglFkaCZ9zup6iSkIYEUUwytxC18%3D&reserved=0), and [computer vision](https://nam05.safelinks.protection.outlook.com/?url=https%3A%2F%2Fen.m.wikipedia.org%2Fwiki%2FComputer_vision&data=02%7C01%7CSunny.anand%40conduent.com%7Caa9a279c30324c190b6f08d6cde7f2af%7C1aed4588b8ce43a8a775989538fd30d8%7C0%7C0%7C636922793425084015&sdata=gecy63DA%2BMBX2D81JH4f8RwCLekj5uut4NTTGQyAGfA%3D&reserved=0), where it is infeasible to develop an algorithm of specific instructions for performing the task. Machine learning is closely related to [computational statistics](https://nam05.safelinks.protection.outlook.com/?url=https%3A%2F%2Fen.m.wikipedia.org%2Fwiki%2FComputational_statistics&data=02%7C01%7CSunny.anand%40conduent.com%7Caa9a279c30324c190b6f08d6cde7f2af%7C1aed4588b8ce43a8a775989538fd30d8%7C0%7C0%7C636922793425084015&sdata=TcXcPMLPfH%2FWuhqqHzFKFP60w%2BIViScuE4bPqnQYpjQ%3D&reserved=0), which focuses on making predictions using computers. The study of [mathematical optimization](https://nam05.safelinks.protection.outlook.com/?url=https%3A%2F%2Fen.m.wikipedia.org%2Fwiki%2FMathematical_optimization&data=02%7C01%7CSunny.anand%40conduent.com%7Caa9a279c30324c190b6f08d6cde7f2af%7C1aed4588b8ce43a8a775989538fd30d8%7C0%7C0%7C636922793425094024&sdata=FPx1NvJvQWG5%2FAQuywrr28kplRHFFOwQ6EJxGqgp820%3D&reserved=0) delivers methods, theory and application domains to the field of machine learning. [Data mining](https://nam05.safelinks.protection.outlook.com/?url=https%3A%2F%2Fen.m.wikipedia.org%2Fwiki%2FData_mining&data=02%7C01%7CSunny.anand%40conduent.com%7Caa9a279c30324c190b6f08d6cde7f2af%7C1aed4588b8ce43a8a775989538fd30d8%7C0%7C0%7C636922793425104028&sdata=o3Z4zI1dW%2BxGO6W7r2UKvViYl7V6CXDF98UVZB2t1Yk%3D&reserved=0) is a field of study within machine learning, and focuses on [exploratory data analysis](https://nam05.safelinks.protection.outlook.com/?url=https%3A%2F%2Fen.m.wikipedia.org%2Fwiki%2FExploratory_data_analysis&data=02%7C01%7CSunny.anand%40conduent.com%7Caa9a279c30324c190b6f08d6cde7f2af%7C1aed4588b8ce43a8a775989538fd30d8%7C0%7C0%7C636922793425104028&sdata=xpffBfCKkjRH7E6XJK3YQV07Ejtnp0M2NocCkGws8Js%3D&reserved=0) through [unsupervised learning](https://nam05.safelinks.protection.outlook.com/?url=https%3A%2F%2Fen.m.wikipedia.org%2Fwiki%2FUnsupervised_learning&data=02%7C01%7CSunny.anand%40conduent.com%7Caa9a279c30324c190b6f08d6cde7f2af%7C1aed4588b8ce43a8a775989538fd30d8%7C0%7C0%7C636922793425114036&sdata=ceFsru2jX4WbaDTrXGuZNRj2OIJ9TdofQ0kxReMjpUQ%3D&reserved=0).[[3]](https://nam05.safelinks.protection.outlook.com/?url=https%3A%2F%2Fen.m.wikipedia.org%2Fwiki%2FMachine_learning%23cite_note-3&data=02%7C01%7CSunny.anand%40conduent.com%7Caa9a279c30324c190b6f08d6cde7f2af%7C1aed4588b8ce43a8a775989538fd30d8%7C0%7C0%7C636922793425114036&sdata=uqJ54RBW2arr20Hz0HGOGJT7a%2FGRH6gVHe8fJZZz6%2BI%3D&reserved=0)[[4]](https://nam05.safelinks.protection.outlook.com/?url=https%3A%2F%2Fen.m.wikipedia.org%2Fwiki%2FMachine_learning%23cite_note-4&data=02%7C01%7CSunny.anand%40conduent.com%7Caa9a279c30324c190b6f08d6cde7f2af%7C1aed4588b8ce43a8a775989538fd30d8%7C0%7C0%7C636922793425124040&sdata=p4nu6RddzQx5B0Wlf%2BCjgQWECT5N41ElIumDXQaoCR0%3D&reserved=0) In its application across business problems, machine learning is also referred to as [predictive analytics](https://nam05.safelinks.protection.outlook.com/?url=https%3A%2F%2Fen.m.wikipedia.org%2Fwiki%2FPredictive_analytics&data=02%7C01%7CSunny.anand%40conduent.com%7Caa9a279c30324c190b6f08d6cde7f2af%7C1aed4588b8ce43a8a775989538fd30d8%7C0%7C0%7C636922793425124040&sdata=WxjhoYGy3%2FNQySIOhwUTBe%2B4Ld5vDZbw5%2BzWNDcWO6U%3D&reserved=0).

TYPES OF MACHINE LEARNING

1 .SUPERVISED LEARNING

2. UNSUPERVISED LEARNING

3. REINFORCEMENT LEARNING

Supervised learning

* Supervised learning, in the context of artificial intelligence (AI) and machine learning, is a type of system in which both input and desired output data are provided. Input and output data are labelled for classification to provide a learning basis for future data processing.
* Supervised machine learning systems provide the learning algorithms with known quantities to support future judgments. Chatbots, self-driving cars, facial recognition programs, expert systems and robots are among the systems that may use either supervised or unsupervised learning. Supervised learning systems are mostly associated with retrieval-based AI but they may also be capable of using a generative learning model.
* Training data for supervised learning includes a set of examples with paired input subjects and desired output (which is also referred to as the supervisory signal).
* In supervised learning for image processing, for example, an AI system might be provided with labelled pictures of vehicles in categories such as cars and trucks. After a sufficient amount of observation, the system should be able to distinguish between and categorize unlabeled images, at which time training can be said to be complete.

**Advantages:**

* Supervised learning models have some advantages over the unsupervised approach, but they also have limitations. The systems are more likely to make judgments that humans can relate to, for example, because humans have provided the basis for decisions. However, in the case of a retrieval-based method, supervised learning systems have trouble dealing with new information. If a system with categories for cars and trucks is presented with a bicycle, for example, it would have to be incorrectly lumped in one category or the other. If the AI system was generative, however, it may not know what the bicycle is but would be able to recognize it as belonging to a separate category.

**Supervised learning Algorithm:(Single layer Perceptrons):**

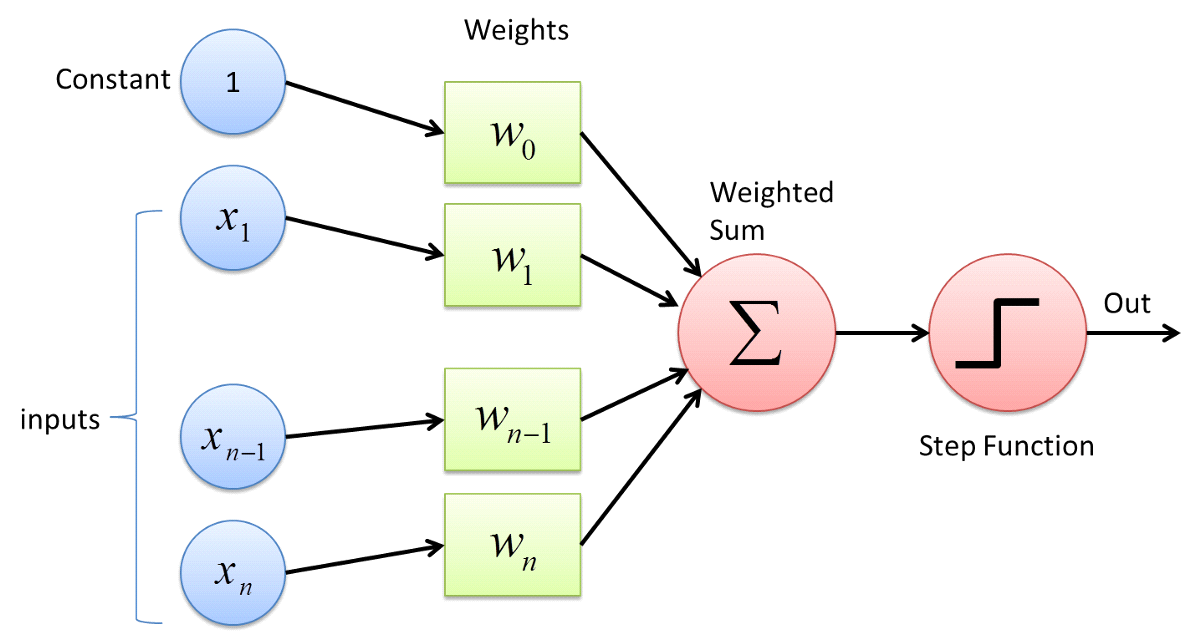
**-What is Single layer Perceptrons??**

Perceptron is a single layer neural network and a multi-layer perceptron is called Neural Networks.

Perceptron is a linear classifier (binary). Also, it is used in supervised learning. It helps to classify the given input data

The perceptron consists of 4 parts .

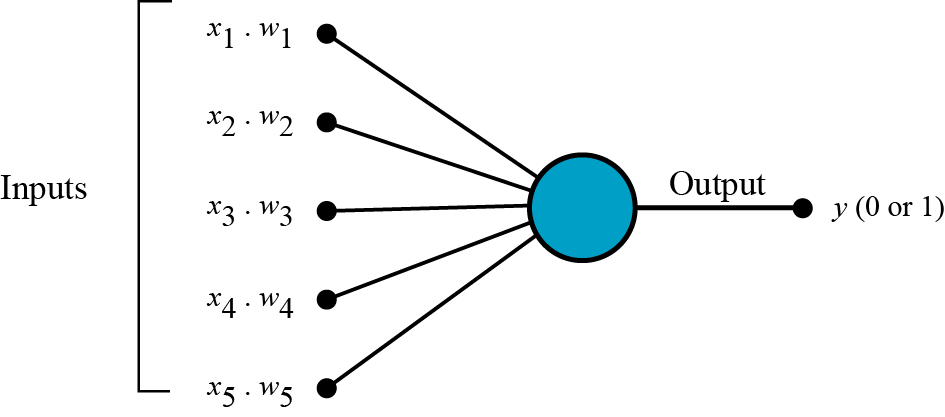
* Input values or One input layer
* Weights and Bias
* Net sum
* Activation Function



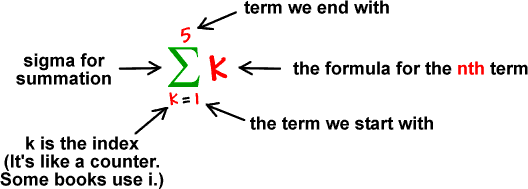
**How does it work?**

The perceptron works on these simple steps

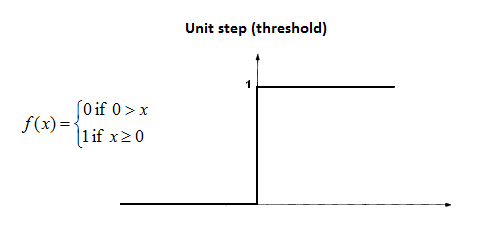
* All the inputs **x** are multiplied with their weights **w**.Let’s call it **k.**

****

b. **Add** all the multiplied values and call them **Weighted Sum.**

****

c. **Apply** that weighted sum to the correct [activation function](https://medium.com/towards-data-science/activation-functions-neural-networks-1cbd9f8d91d6). For Example : Unit Step Activation Function.



**Where we use Perceptron?**

Perceptron is usually used to classify the data into two parts.

**Learning Algorithm:**

* Initialize weights at random
* For each training pair/pattern ( x, ytarget )

- Compute output y

- Compute error, δ=(ytarget – y)

- Use the error to update weights as follows:

∆w = w – wold = η \* δ\*x or wnew = wold + η \* δ\*x

where η is called the learning rate or step size and it determines how smoothly the learning process is taking place.

* Repeat 2 until convergence (i.e. error δ is zero)
* The Perceptron Learning Rule is then given by wnew = wold + η \* δ\*x where δ=(ytarget – y)

**IMPLEMENTED CODE**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import random

from sklearn.metrics import confusion\_matrix,accuracy\_score,precision\_score,recall\_score,f1\_score

from mlxtend.plotting import plot\_decision\_regions

from sklearn.decomposition import PCA

w = []

data = pd.read\_csv('slp.csv')

b = []

y\_acc = []

x\_epoc = []

data = data.iloc[:,[2,3,4]]

for i in range(len(data.index)):

b.append(-1)

data.insert(loc=0,column='bais',value=b)

def initilize\_weights(data):

no\_of\_weigths = len(data.columns)

for i in range( no\_of\_weigths-1):

w.append( round(random.uniform(-0.05,0.05),2))

def accuracy(predvalues,target,i):

print('Accuracy score: {}'.format(accuracy\_score(target, predvalues)))

#for early stop

y\_acc.append(accuracy\_score(target, predvalues))

x\_epoc.append(i)

plt.subplot(2,1,1)

plt.plot(x\_epoc,y\_acc)

plt.title("early stop")

plt.xlabel("epocs")

plt.ylabel("accuracy")

# print('Precision score: {}'.format(precision\_score(target, predvalues)))

# print('Recall score: {}'.format(recall\_score(target, predvalues)))

#print('F1 score: {}'.format(f1\_score(target, predvalues)))

def update\_weigths(w,n,ypred,t,dataarray,j):

for i in range(len(w)):

w[i] = w[i]- (n\*(ypred-t)\*dataarray[j][i])

def confusionmatrix(predvalues,t):

print("confusion matrix")

cm = confusion\_matrix(t,predvalues)

print(cm)

print("\n")

def plot\_decision(X,predvalues,target):

#dimensionality reduction for plotting

pca = PCA(n\_components=1)

principalComponents = pca.fit\_transform(X)

principalDf = pd.DataFrame(data = principalComponents

, columns = ['principal component 1'])

print(principalDf)

plt.subplot(2,1,2)

from matplotlib.colors import ListedColormap

X\_set, y\_set = X,target

X1, X2 = np.meshgrid(np.arange(X\_set[:, 0].min() - 1, X\_set[:, 0].max() + 1,step=0.1),

np.arange(X\_set[:, 1].min() - 1, X\_set[:, 1].max() + 1,step=0.1))

plt.contourf(X1, X2, slp.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j)

predvalues = []

class slp:

def train\_model(epocs,learning\_rate):

y=0

target = np.array(data['Purchased'])

dataarray= np.array(data)

for i in range(epocs):

print("starting with epocs ---->"+" "+str(i+1) )

print("\n")

predvalues = []

for k in range(len(data.index)):

for j in range( len(data.columns)-1):

y = y + (dataarray[k][j]\*w[j])

#print("activation value:" +" "+str(y))

if y>0:

ypred = 1

else:

ypred = 0

predvalues.append(ypred)

if ypred != target[k]:

update\_weigths(w,learning\_rate,ypred,target[k],dataarray,k)

print("updated weightd are :")

print(w)

#i is the nth epoc

print("predicted values for epoc"+" "+ str(i+1)+" is:")

print(predvalues)

accuracy(predvalues,target,i)

confusionmatrix(predvalues,target)

#plot\_decision(dataarray[:,1:3],predvalues,target)

def predict(x):

predictedvalues = []

for i in x:

y=0

y = y+(-1\*w[0])

for j in range(len(i)):

y = y + (i[j]\*w[j+1])

if y>0:

predictedvalues.append(1)

else:

predictedvalues.append(0)

predictedvalues = np.array(predictedvalues)

return predictedvalues

initilize\_weights(data)

slp.train\_model(epocs=13,learning\_rate=0.25)

#testing

ypredicted = slp.predict(np.array([[27,84000]]))

print(ypredicted)

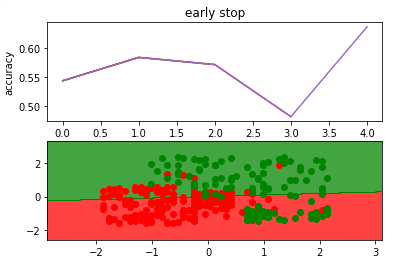
**Result: For train data:**

After 5 epochs the accuracy is 65%

confusion matrix

[198 11]

[105 6]



**For test data:**

The accuracy is 78%

confusion matrix

[32 16]

[ 2 30]

UNSUPERVISED LEARNING

Unsupervised learning is the training of an artificial intelligence (AI) algorithm using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance.

In unsupervised learning, an AI system may group unsorted information according to similarities and differences even though there are no categories provided. AI systems capable of unsupervised learning are often associated with generative learning models, although they may also use a retrieval-based approach (which is most often associated with supervised learning). Chatbots, self-driving cars, facial recognition programs, expert systems and robots are among the systems that may use either supervised or unsupervised learning approaches.

In unsupervised learning, an AI system is presented with unlabeled, uncategorised data and the system’s algorithms act on the data without prior training. The output is dependent upon the coded algorithms. Subjecting a system to unsupervised learning is one way of testing AI.

Unsupervised learning algorithms can perform more complex processing tasks than supervised learning systems. However, unsupervised learning can be more unpredictable than the alternate model. While an unsupervised learning AI system might, for example, figure out on its own how to sort cats from dogs, it might also add unforeseen and undesired categories to deal with unusual breeds, creating clutter instead of order.

**Types of Unsupervised Learning**

**Clustering**

Any business needs to focus on understanding customers: who they are and what’s driving their purchase decisions?

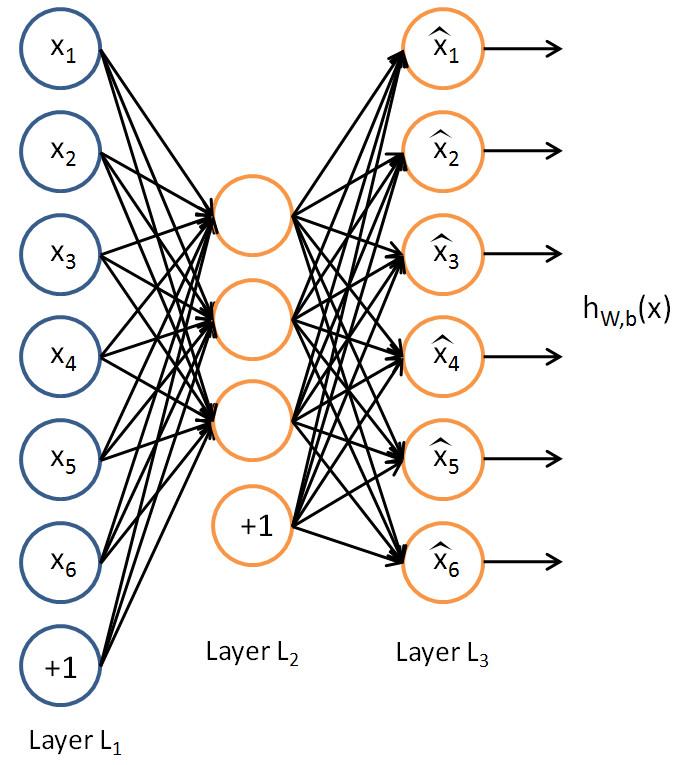
You’ll usually have different groups of users that can be split across a few criteria. These criteria can be as simple, such as age and gender, or as complex as persona and purchase process. Unsupervised learning can help you accomplish this task automatically.

Clustering algorithms will run through your data and find these natural clusters if they exist. For your customers, that might mean one cluster of 30-something artists and another of millennials who own dogs. You can typically modify how many clusters your algorithms looks for, which lets you adjust the granularity of these groups. There are a few different types of clustering you can utilize:

**K-Means Clustering** – Clustering your data points into a number (K) of mutually exclusive clusters. A lot of the complexity surrounds how to pick the right number for K.

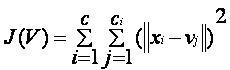
Hierarchical Clustering – clustering your data points into parent and child clusters. You might split your customers between younger and older ages, and then split each of those groups into their own individual clusters as well.

Probabilistic Clustering – clustering your data points into clusters on a probabilistic scale.



**ALGORITHM**: **k-means clustering algorithm**

K-means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed apriori. The main idea is to define k centers, one for each cluster. These centers should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest center. When no point is pending, the first step is completed and an early group age is done. At this point we need to re-calculate k new centroids as barycenter of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new center. A loop has been generated. As a result of this loop we may notice that the k centers change their location step by step until no more changes are done or in other words centers do not move any more. Finally, this algorithm aims at minimizing an objective function know as squared error function given by:



where,

‘||xi - vj||’ is the Euclidean distance between xi and vj.

‘ci’ is the number of data points in ith cluster.

‘c’ is the number of cluster centers.

**Algorithmic steps for k-means clustering**

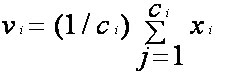
Let X = {x1,x2,x3,……..,xn} be the set of data points and V = {v1,v2,…….,vc} be the set of centers.

1) Randomly select ‘c’ cluster centers.

2) Calculate the distance between each data point and cluster centers.

3) Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers..

4) Recalculate the new cluster center using:



where, ‘ci’ represents the number of data points in ith cluster.

5) Recalculate the distance between each data point and new obtained cluster centers.

6) If no data point was reassigned then stop, otherwise repeat from step 3).

**Advantages**

1) Fast, robust and easier to understand.

2) Relatively efficient: O(tknd), where n is # objects, k is # clusters, d is # dimension of each object, and t is # iterations. Normally, k, t, d << n.

3) Gives best result when data set are distinct or well separated from each other.

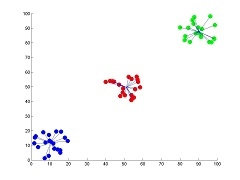


Fig I: Showing the result of k-means for 'N' = 60 and 'c' = 3

Note: For more detailed figure for k-means algorithm please refer to k-means figure sub page.

Disadvantages

1) The learning algorithm requires apriori specification of the number of cluster centers.

2) The use of Exclusive Assignment - If there are two highly overlapping data then k-means will not be able to resolve that there are two clusters.

3) The learning algorithm is not invariant to non-linear transformations i.e. with different representation of data we get

different results (data represented in form of cartesian co-ordinates and polar co-ordinates will give different results).

4) Euclidean distance measures can unequally weight underlying factors.

5) The learning algorithm provides the local optima of the squared error function.

6) Randomly choosing of the cluster center cannot lead us to the fruitful result. Pl. refer Fig.

7) Applicable only when mean is defined i.e. fails for categorical data.

8) Unable to handle noisy data and outliers.

9) Algorithm fails for non-linear data set.

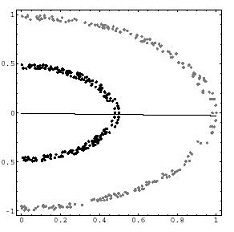
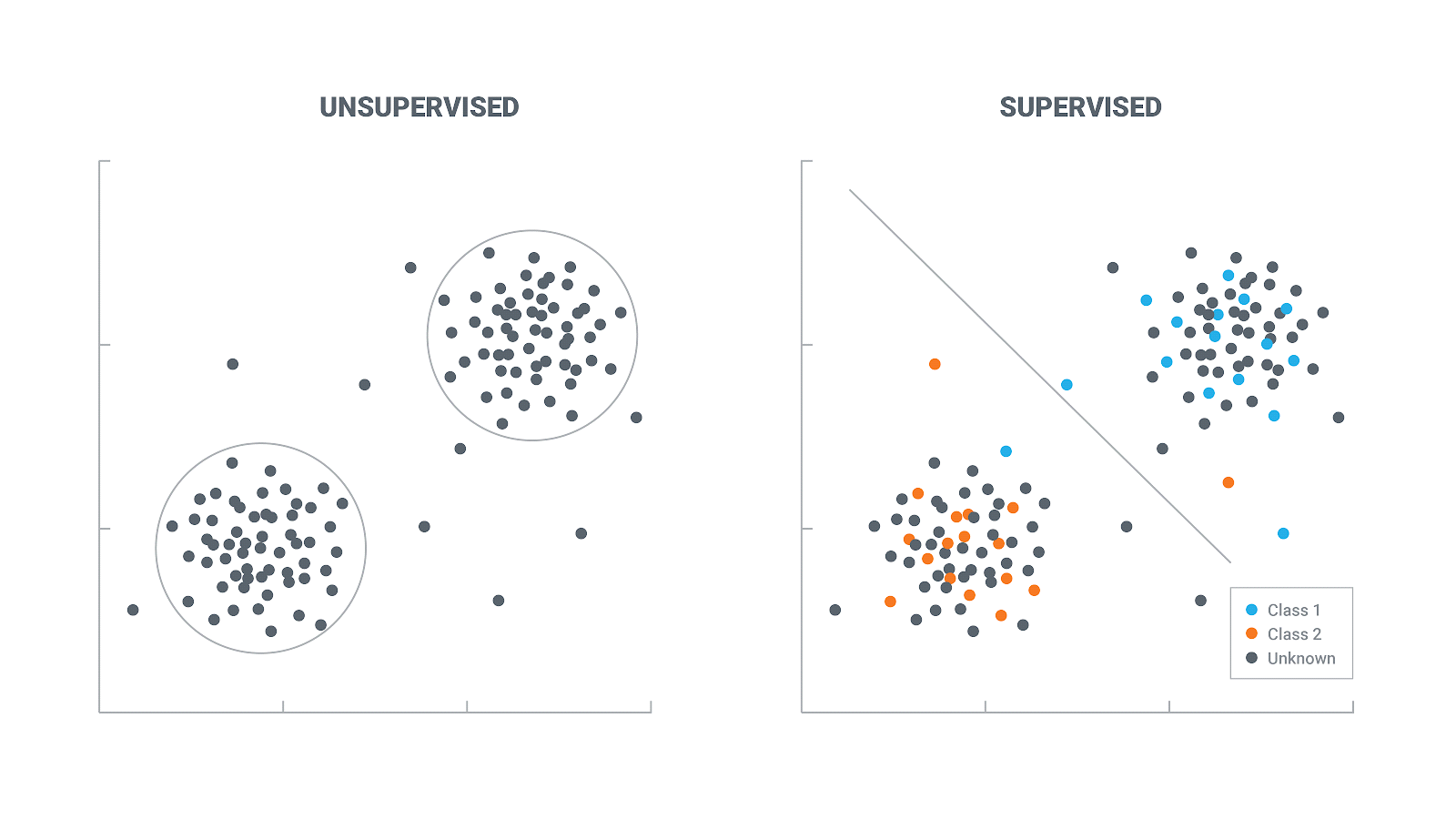


Fig II: Showing the non-linear data set where k-means algorithm fails

**SUPERVISED VS UNSUPERVISED GRAPHICAL REPRESENTATION**

****

**IMPLEMENTED CODE**

import matplotlib.pyplot as plt

from matplotlib import style

style.use('ggplot')

import numpy as np

import pandas as pd

from sklearn import preprocessing

#defining c;lass

class K\_Means:

def \_\_init\_\_(self,k=2,tol=0.001,max\_iter=300):

self.tol=tol

self.k=k

self.max\_iter=max\_iter

def fit(self,data):

self.centroids=np.array([data[i] for i in range(self.k)])

self.centroids={}

prev\_centroids={}

for i in range(self.k):

self.centroids[i]=data[i]

prev\_centroids[i]=data[i]

#for i in range(self.max\_iter):

for i in range(self.max\_iter):

self.classifications={}

for i in range(self.k):

self.classifications[i]=[]

for featureset in x:

distances=[np.linalg.norm(featureset-self.centroids[centroid]) for centroid in self.centroids]

classification=distances.index(min(distances))

self.classifications[classification].append(featureset)

prev\_centroids=dict(self.centroids)

for classification in self.classifications:

self.centroids[classification]=np.average(self.classifications[classification],axis=0)#find centroid for all value

optimized=True

for c in self.centroids:

original\_centroid=prev\_centroids[c]

current\_centroid=self.centroids[c]

if np.sum((current\_centroid-original\_centroid)/original\_centroid\*100.0)>self.tol:

print(np.sum((current\_centroid-original\_centroid)/original\_centroid\*100.0))

optimized=False

if optimized:

break

def predict(self,data):

distances=[np.linalg.norm(data-self.centroids[centroid]) for centroid in self.centroids]

classification=distances.index(min(distances))

return classification

df=pd.read\_csv('train.csv')

df.drop(['Name'],1,inplace=True)

df.convert\_objects(convert\_numeric=True)

print(df.head())

df.fillna(0,inplace=True)

def handle\_non\_numerical\_data(dataset):

columns=df.columns.values

for column in columns:

text\_digit\_vals={}

def convert\_to\_int(val):

return text\_digit\_vals[val]

if df[column].dtype!=np.int64 and df[column].dtype!=np.float64:

column\_contents=df[column].values.tolist()

unique\_elements=set(column\_contents)

x=0

for unique in unique\_elements:

if unique not in text\_digit\_vals:

text\_digit\_vals[unique]=x

x=x+1

df[column]=list(map(convert\_to\_int,df[column]))

return df

df=handle\_non\_numerical\_data(df)

print(df.head())

df.drop(['Ticket'],1,inplace=True)

x=np.array(df.drop(['Survived'],1).astype(float))

x=preprocessing.scale(x)

y=np.array(df['Survived'])

clf=K\_Means()

clf.fit(x)

correct=0

for i in range(len(x)):

predict\_me=np.array(x[i].astype(float))

predict\_me=predict\_me.reshape(-1,len(predict\_me))

prediction=clf.predict(predict\_me)

if(prediction == y[i]):

correct+=1

print(correct/len(x))

clf=K\_Means()

clf.fit(x)

for centroid in clf.centroids:

plt.scatter(clf.centroids[centroid][0],clf.centroids[centroid][1],marker="o",color="k",s= 150,linewidth=5)

for classification in clf.classifications:

for featureset in clf.classifications[classification]:

plt.scatter(featureset[0],featureset[1],marker="x",s=150,linewidth=5)

plt.show()

**RESULTS**

**PassengerId Survived Pclass Sex Age SibSp Parch**

0 1 0 3 male 22.0 1 0

1 2 1 1 female 38.0 1 0

2 3 1 3 female 26.0 0 0

3 4 1 1 female 35.0 1 0

4 5 0 3 male 35.0 0 0

**Ticket Fare Cabin Embarked**

0 A/5 21171 7.2500 NaN S

1 PC 17599 71.2833 C85 C

2 STON/O2. 3101282 7.9250 NaN S

3 113803 53.1000 C123 S

4 373450 8.0500 NaN S

**PassengerId Survived Pclass Sex Age SibSp Parch Ticket Fare** \

0 1 0 3 0 22.0 1 0 493 7.2500

1 2 1 1 1 38.0 1 0 100 71.2833

2 3 1 3 1 26.0 0 0 525 7.9250

3 4 1 1 1 35.0 1 0 263 53.1000

4 5 0 3 0 35.0 0 0 451 8.0500

**Cabin Embarked**

0 0 3

1 133 2

2 0 3

3 97 3

4 0 3

**ACCURACY**

0.6790123456790124

0.6801346801346801

0.6812570145903479

0.6812570145903479

0.6823793490460157

0.6835016835016835

0.6846240179573513

0.6857463524130191

0.6868686868686869

SS0.6879910213243546

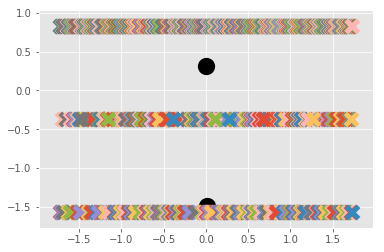
0.6891133557800224

0.6902356902356902

0.691358024691358

0.6924803591470258

**RESULTING CLUSTERS**

****