## 1. Describe supervised and unsupervised learning and provide one specific example for both types of learning. For each example state the problem and describe the evaluation plan (training, validation and test sets).

*Supervised learning* is a machine learning approach that’s defined by its use of labeled datasets. The machine is trained or taught using data that is labelled. After that, the machine is provided with a new set of data into classifying data or predicting outcomes accurately. Using labeled inputs and outputs, the model can measure its accuracy and learn over time.

There are two main areas where supervised machine learning comes in handy: classification problems and regression problems.

**Classification**

Classification refers to taking an input value and mapping it to a discrete value. In classification problems, our output typically consists of classes or categories. This could be things like trying to predict what objects are present in an image (an apple/ an orange/a banana) or whether a transaction is going to fraud or not.

**Regression**

Regression is related to continuous data (value functions). In Regression, the predicted output values are real numbers. It deals with problems such as predicting the price of a house or the trend in the stock price at a given time, etc.

[*Unsupervised learning*](https://www.ibm.com/cloud/learn/unsupervised-learning) uses machine learning algorithms to analyze and cluster unlabeled data sets. These algorithms discover hidden patterns in data without the need for human intervention.

In unsupervised learning algorithms are provided with data that does not contain any labels or explicit instructions on what to do with it. The goal is for the learning algorithm to find structure in the input data on its own.

In other words, unsupervised Learning is a kind of self-learning where the algorithm can find previously hidden patterns in the unlabeled datasets and give the required output without any interference.

Identifying these hidden patterns helps in clustering, association, and detection of anomalies and errors in data.

Unsupervised Learning has been split up majorly into 2 types:

**Clustering**

Clustering isa data mining technique which groups unlabeled data based on their similarities or differences. Clustering algorithms are used to process raw, unclassified data objects into groups represented by structures or patterns in the information.

**Association**

An association rule learning problem is where we want to discover rules that describe large portions of the data, such as people that buy X also tend to buy Y.

The main distinction between the two approaches is the use of labeled datasets. To put it simply, supervised learning uses labeled input and output data, while an unsupervised learning algorithm does not.

|  |  |  |
| --- | --- | --- |
| Parameters | Supervised machine learning technique | Unsupervised machine learning technique |
| Process | In a supervised learning model, input and output variables will be given. | In unsupervised learning model, only input data will be given |
| Input Data | Algorithms are trained using labeled data. | Algorithms are used against data which is not labeled |
| Algorithms Used | Support vector machine, Neural network, Linear and logistics regression, random forest, and Classification trees. | Unsupervised algorithms can be divided into different categories: like Cluster algorithms, K-means, Hierarchical clustering, etc. |
| Use of Data | Supervised learning model uses training data to learn a link between the input and the outputs. | Unsupervised learning does not use output data. |
| Number of Classes | Number of classes is known. | Number of classes is not known. |

Example:

**Supervised learning:** One of the examples of supervised learning is Fraud detection in financial applications. [Fraud is a massive problem for financial institutions](https://algorithmxlab.com/blog/2017/10/30/danske-bank-joins-others-using-ai-detect-fraud/).

[Fraud](https://algorithmxlab.com/blog/2018/06/27/insurers-using-artificial-intelligence-to-fight-fraud-2/) losses incurred by banks and merchants on all credit, debit, and prepaid general purpose and private label payment cards issued globally amounted to $20 billion in 2015, according to a [Bloomberg report](https://www.forbes.com/sites/rogeraitken/2016/10/26/us-card-fraud-losses-could-exceed-12bn-by-2020/?sh=54c2559ad243).

The challenge in fraud detection is that the majority of transactions that occur are genuine transactions and only a very small portion account for fraudulent behavior. So, the fraud detection algorithm has to be very careful about the False positive and False negative rate to avoid customer dissatisfaction and maintain their loyalty. It also has to be kept in mind that in case of fraud, the customer or card holder is not liable for the charge, either the card issuer or the merchant has to bear the financial loss, so a very high accuracy of the model is mandatory.

The solution to the biased data in credit card transactional data towards genuine transactions is to balance data so that random dataset without the bias of any kind like geographical location, kind of product, type of customer, amount of transaction etc. are chosen.

As a next step, the complete dataset is segregated into training and test dataset randomly. The training dataset can be considered as “supervised” or “teacher” dataset to generate the model. Once the model is available, we will evaluate the model in terms of reliability, accuracy using several means like AUC curve, or confusion matrix, or precision/recall score, accuracy\_score or mean squared error.The model will be refined continuously until a satisfactory accuracy score is available. At this point the model has intelligence of the impact of different features like geographical location of transaction from the cardholders address, the number of transactions in a specific time frame, the kind of merchant where the transaction is happening, amount of the transaction, currency of transaction (if different from the currency the card does the transaction normally), channel of transaction like primarily online etc. on determining the fraud or genuine transaction with a high accuracy.

The next step is to use the model for the test dataset for prediction of fraud in terms of “Yes” or “No” and refine the model in case of issues like overfitting or dependence of specific parameters to define fraud.

**Unsupervised learning:** One of the interesting unsupervised learning can be choice of the department in an university from the prospective students. Some very interesting analysis has been done in this field. Based on the technique and outcome of the data, the universities can prepare and realign and plan for courses, number of professors, number of lab instruments, realignment of classrooms etc. well in advance to the enrollment process begins.

The dataset consists of the high school students’ information like family income, parents’ education status, siblings course status, kind of courses taken by the students during their high school, hobbies and interest, availability of internet at home, dwelling place, home ownership, age of student, parents’ occupation etc. Based on the analysis of the data using clustering technique, the model predicts the likelihood of the students to opt for computer science, other engineering, biological science, other science, modern art, traditional art, language, social science or no college admission clusters.

The model also reduces the dimensionality from different data points to specific data points like different activities enrollment to a column of hobbies and interests etc. and prepares a primary component analysis for the model.

Using the model, universities can predict the likely number of applicants in different courses and plan ahead for hiring new professor or instruments well in advance to the actual beginning of the semester.

## 2. There are 2 boxes containing 2 sets of colored balls. The first box contains 5 red balls and 3 blue balls, while the second box contains 6 blue balls and 4 red balls. If a ball is drawn at random and found to be red, what is the probability that it was drawn from the second box?

|  |  |  |  |
| --- | --- | --- | --- |
|  | Set 1 | Set 2 | Total |
| Red | 5 | 4 | 9 |
| Blue | 3 | 6 | 9 |
| Total | 8 | 10 | 18 |

Bayes theorem:

P (A|B) = P(B|A) \* P(A) / P(B)

hence

P(Box2 | Red) = P( Red | Box2 ) \* P(Box2) / P (Red)

P(Red | Box2) = 4/10

P(Red) = 9/18 = 1/2

P(Box2) = 10/18 = 5/9

P(Box2 | Red) = P( Red | Box2 ) \* P(Box2) / P (Red)

= 4/10 \* 5/9 / (1/2)

= (2/9) / (1/2) = 4/9

3. A production company has 3 sites A, B, and C, with a production capacity of 50%, 30%, and 20% respectively. 6% of total production from site A is found to be defective, 4% of total production from site B is defective, and 1.5% of total production from site C is defective. An item selected at random is found to be defective. Find the probability that the item was produced at site B

|  |  |  |  |
| --- | --- | --- | --- |
|  | Prod Capacity | Defect Rate | SUM |
| A | 0.5 | 0.06 | 0.03 |
| B | 0.3 | 0.04 | 0.012 |
| C | 0.2 | 0.015 | 0.003 |
|  |  |  | 0.045 |

P(D/A) = 0.06

P(D/B) = 0.04

P(D/C) = 0.015

P(A) = 0.5

P(B) = 0.3

P(C) = 0.2

P(B/D)=

P(B)×P(D/B)

---------------------------------------------------------

[P(B)×P(D/B)]+[P(A)×P(D/A)]+[P(C)×P(D/C)]

=

0.3 \* 0.04

--------------------------------------------------

0.3 \* 0.04 + 0.5 \* 0.06 + 0.2 \* 0.015

= 0.012/ 0.045 = 0.267, i.e. 26.7%

4. Implement basic k-NN classification and the condensed 1-NN algorithm for the Letter Recognition Data Set. The first 15,000 examples are for training and the remaining 5,000 for testing.

**About the dataset:**

The dataset contains 17 columns with the first column being the expected label (letter)

**K-NN classification:**

In this algorithm, the driving factor is that similar items tend to be closer in groups. There is no learning needed as the model stores the complete dataset and classification is based on the points that are like it. The implemented algorithm has four steps as explained below:

Step 1: Find the Euclidean distance between each test data against all training data points.

Step 2: Sort the distances in ascending order (to get closest to training points) and select the first K elements.

Step 3: Find the most occurring class in the K elements. This will be new class for the data point to be classified.

Step 4: Repeat for remaining test data points

**Validation and accuracy:**

The KNN algorithm was applied on multiple values of K ranging from 1 to 9. The results are shown below. The best K value for this dataset is 1 with an accuracy of 95.44

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| K-value | 1 | 3 | 5 | 7 | 9 |
| Accuracy | 95.44% | 95.28% | 95.32% | 95.12% | 94.8% |

**Condensed 1-NN**:

In this method, the training dataset is reduced to the minimum that is required to get best accuracy. Initially, the first training data point is taken, and the remaining training points are used as the test dataset. Points are randomly selected and classified using KNN algorithm with k=1, and all incorrectly classified points are added to the condensed dataset. This process is repeated till all the remaining training data are classified. This condensed dataset is then used as the training dataset for the actual test data to classify letters using 1NN algorithm as explained earlier.

**Execution statistics:**

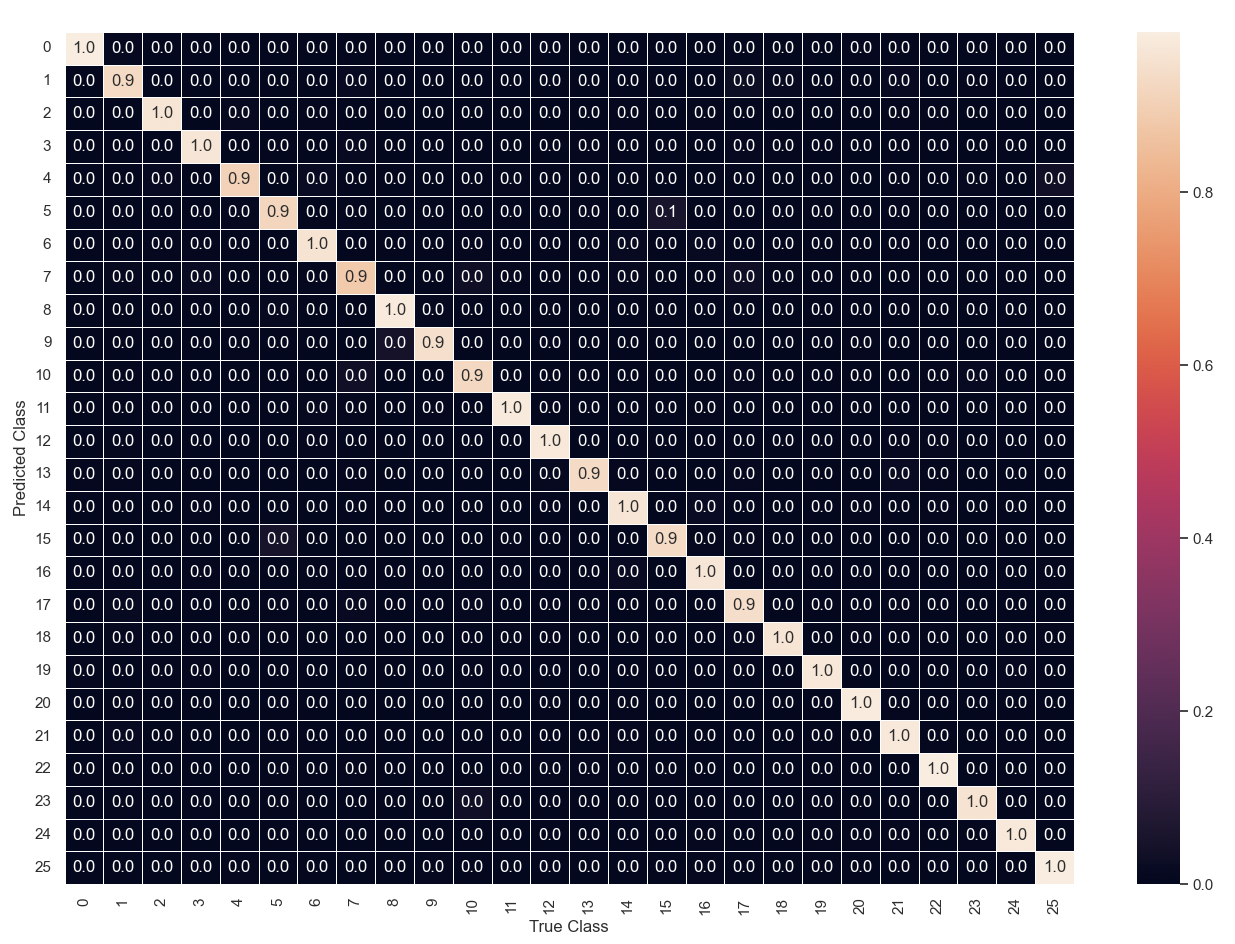
Size of full training dataset 🡪 15000

Size of condensed dataset 🡪 2240

Accuracy with k=1 🡪 91.62%

**Confusion Matrix:**

The following confusion matrix was generated against the full data set with a KNN classifier and k = 1



**Code:**

import datetime

import random as rd

from statistics import mode

import numpy as np

import pandas as pd

# KNN algorithm

def knn\_predict(train\_x, train\_y, test\_x, k):

predicted = []

# iterate through the test data set to be classified

for item in test\_x:

distances = []

# find distance between test data and each individual training Data

for j in range(len(train\_x)):

distances.append(calculate\_distance(np.array(train\_x[j]), item))

distances = np.array(distances)

# Sort the distances and get the first k record's index

dist = np.argsort(distances)[:k]

# get labels of the k data points

labels = train\_y[dist]

# majority label occurrence

predicted.append(mode(labels)[0])

return predicted

# compute distance between two vectors

def calculate\_distance(v1, v2):

# using euclidean distance

return np.linalg.norm(v1 - v2)

# iterate over mulitple ks to determine the one with best accuracy

def knn\_best\_k\_accuracy(train\_x, test\_x, train\_y, test\_y, start\_k, max\_k):

for k in range(start\_k, max\_k, 2):

actual\_y = knn\_predict(train\_x, train\_y, test\_x, k)

accuracy = (test\_y == actual\_y).sum() / float(test\_y.size) \* 100

print('k is {}, score is {}'.format(k, accuracy))

# method to condense data

def condense\_data(train\_x, train\_y, k):

count = len(train\_x)

# start condensing with the first element

condensed\_x = [train\_x[0]]

condensed\_y = [train\_y[0]]

condensed\_idx = []

processed = []

# iterate through the training data set

while len(processed) != len(train\_x):

# identify a random index that is in the range of training data set and is not yet processed

i = rd.choice([x for x in range(count) if x not in processed])

processed.append(i)

# perform knn on the condensed dataset as training set and classify remaining training samples

test\_y = knn\_predict(np.array(condensed\_x), np.array(condensed\_y), np.array([train\_x[i]]), k)

# if classification don't match, add to the condensed training set

if test\_y[0] != train\_y[i]:

condensed\_x.append(train\_x[i])

condensed\_y.append(train\_y[i])

condensed\_idx.append(i)

return np.asarray(condensed\_x), np.asarray(condensed\_y)

def read\_data(split\_records):

df = pd.read\_csv('letter-recognition.data', header=None)

# df = df.iloc[0:10]

data = df[df.columns[1:]].to\_numpy()

label = df[df.columns[0]].to\_numpy()

train\_x, test\_x = data[0:split\_records, :], data[split\_records:]

train\_y, test\_y = label[0:split\_records], label[split\_records:]

return train\_x, test\_x, train\_y, test\_y

def main():

# tuning params

split\_records = 15000

start\_k = 1

max\_k = 11

# read data

train\_x, test\_x, train\_y, test\_y = read\_data(split\_records)

# ####### KNN ####################

start\_time = datetime.datetime.now()

print('knn started at:', start\_time)

knn\_best\_k\_accuracy(train\_x, test\_x, train\_y, test\_y, start\_k, max\_k)

end\_time = datetime.datetime.now()

print('knn ended at {}. It took {} '.format(end\_time, end\_time - start\_time))

# ####### Condensed 1NN ####################

k = 1

start\_time = datetime.datetime.now()

print('Condensed 1-NN started at:', start\_time)

condensed\_x, condensed\_y = condense\_data(train\_x, train\_y, k)

print('Condensed training size {}, original size {}'.format(len(condensed\_x), len(train\_x)))

actual\_y = knn\_predict(condensed\_x, condensed\_y, test\_x, k)

accuracy = (test\_y == actual\_y).sum() / float(test\_y.size) \* 100

print('Condensed 1-NN score is {}'.format(accuracy))

end\_time = datetime.datetime.now()

print('Condensed 1-NN ended at {}. It took {} '.format(end\_time, end\_time - start\_time))

if \_\_name\_\_ == '\_\_main\_\_':

main()