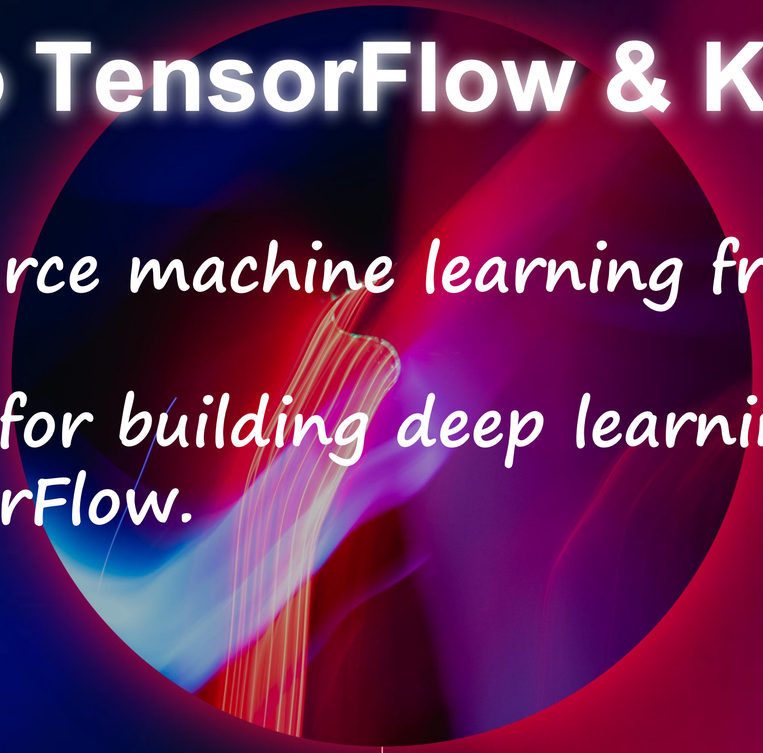
Deep learning basics with TensorFlow & Keras

By: Eng. Shereen Ebrahim

INCEPTUM
EGYPT

ENG. SHEREEN EBRAHIM

Introduction to TensorFlow & Keras

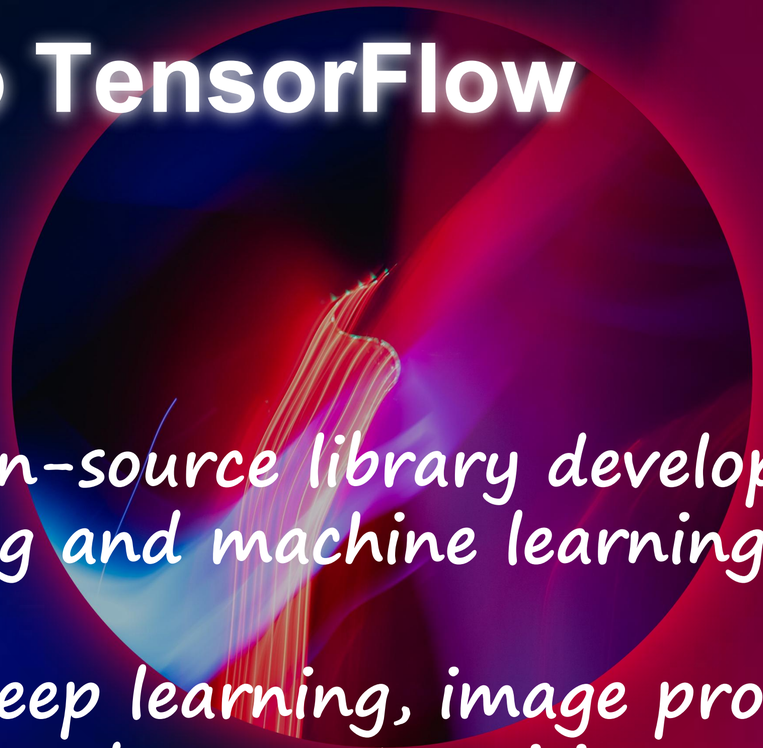


TensorFlow: Open-source machine learning framework developed by Google.
Keras: High-level API for building deep learning models, integrated with TensorFlow.

Why use them?

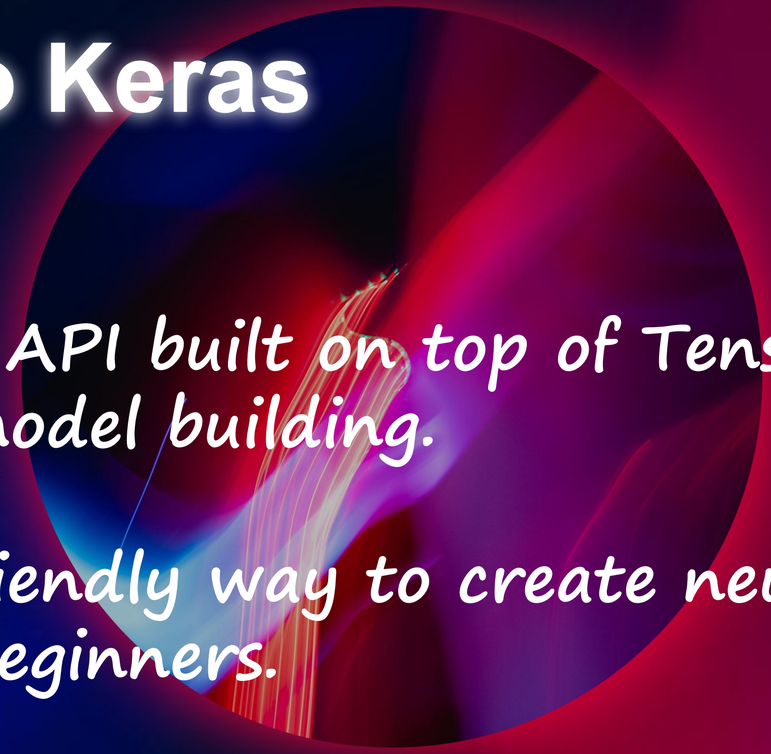
- Easy to use and scalable.
- Supports GPU acceleration.
- Large community support.

Introduction to TensorFlow



- TensorFlow is an open-source library developed by Google for building deep learning and machine learning models.
- It is widely used in deep learning, image processing, natural language processing, and computer vision.

Introduction to Keras



- Keras is a high-level API built on top of TensorFlow, designed for easy and quick model building.
- It provides a user-friendly way to create neural networks, making it ideal for beginners.
- Supports convolutional neural networks (CNNs), recurrent neural networks (RNNs), and other types of architectures.

ENG. SHEREEN EBRAHIM

Inceptum Egypt

Applications of Keras and TensorFlow

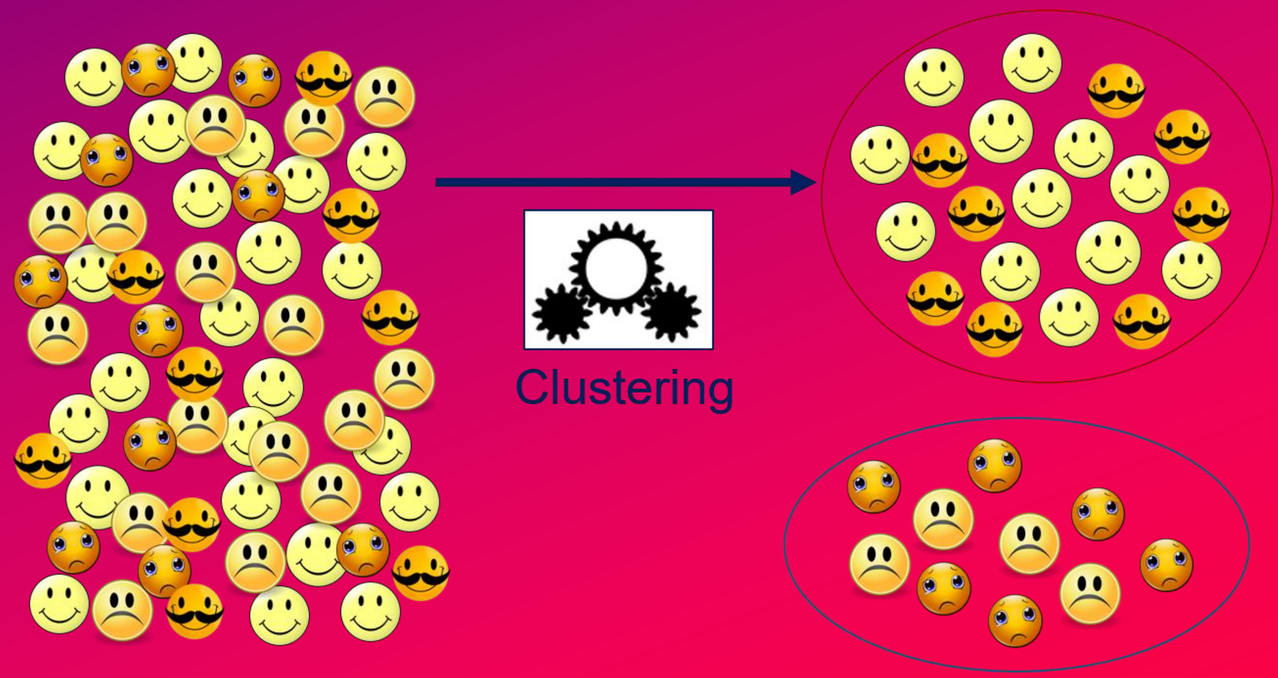
- Image Classification
- Speech Recognition
- Text Analysis & NLP
- Robotics and Computer Vision

ENG. SHEREEN EBRAHIM

K-Means Clustering

INTRODUCTION- What is clustering?

- Clustering is somewhat different from the classification, numeric prediction, and pattern detection tasks we examined.
- cluster refers to a group of similar things or people that are positioned or occur closely together.



INTRODUCTION-
What is clustering?

Distance measure will determine how the *similarity* of two elements is calculated and it will influence the shape of the clusters.

They include:

1. The Manhattan distance (also called 2-norm distance) is given by $d(x,y)=\sum_{i=1}^{p}|x_i-y_i|$

2. The Euclidean distance (also called taxicab norm or 1-norm) is given by: $d(x,y)=\sqrt[2]{\sum_{i=1}^{p}|x_i-y_i|^2}$

INTRODUCTION- What is clustering?

An algorithm for partitioning (or clustering) N data points into K disjoint subsets S_j containing data points so as to minimize the sum-of-squares criterion

$$J = \sum_{j=1}^{K} \sum_{n \in S_j} |x_n - \mu_j|^2,$$

where x_n is a vector representing the the n^{th} data point and u_j is the geometric centroid of the data points in S_j.

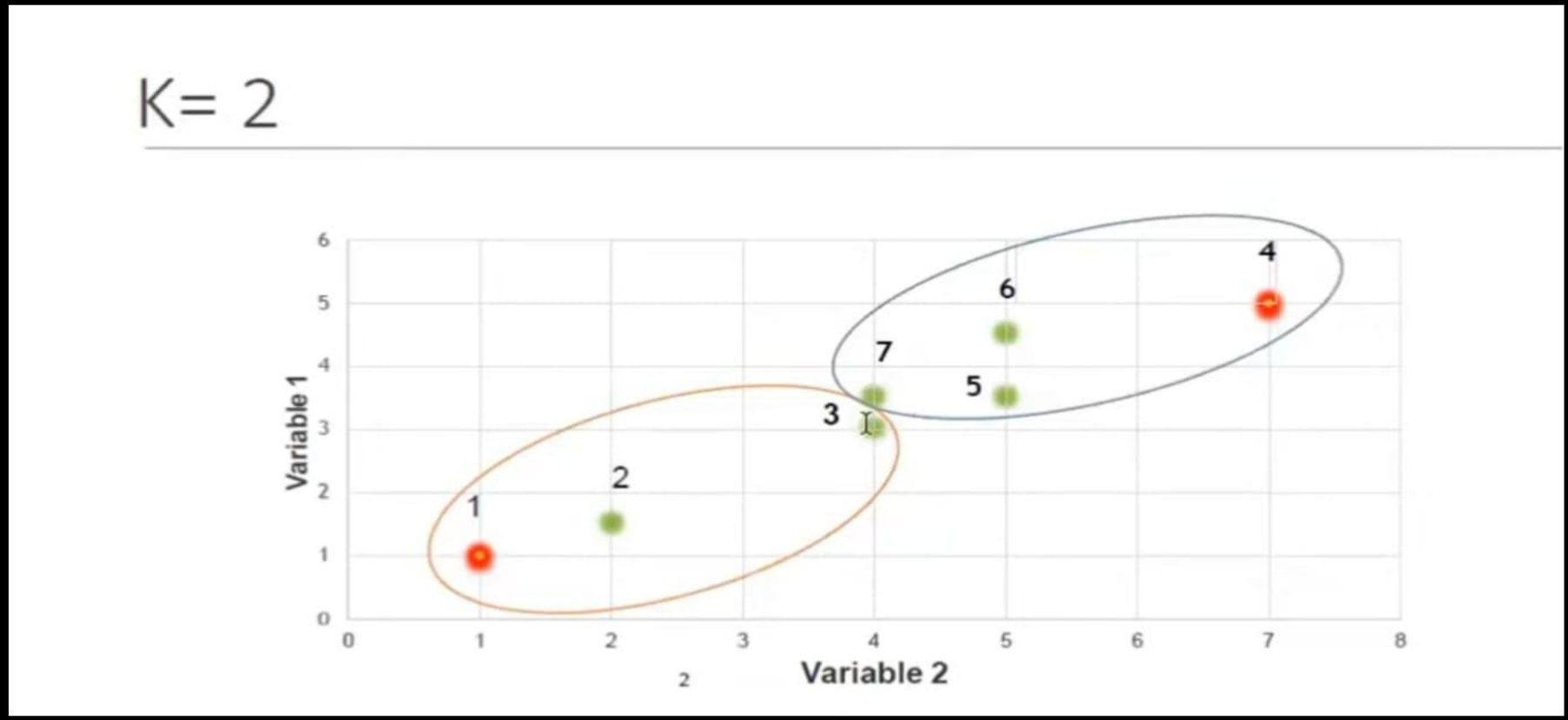
INTRODUCTION- What is clustering?

- Simply speaking k-means clustering is an algorithm to classify or to group the objects based on attributes/features into K number of group.
- K is positive integer number.
- The grouping is done by minimizing the sum of squares of distances between data and the corresponding cluster centroid.

A Simple example showing the implementation of k-means algorithm (using K=2)

Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

A Simple example showing the implementation of k-means algorithm (using K=2)



Step 1:
Initialization: Randomly we choose following two centroids (k=2) for two clusters.
In this case the 2 centroid are: m1=(1.0,1.0) and m2=(5.0,7.0).

Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

	Individual	Mean Vector
Group 1	1	(1.0, 1.0)
Group 2	4	(5.0, 7.0)

Step 2:

	Centroid 1	Centroid 2
1	$\sqrt{(1-1)^2 + (1-1)^2} = 0$	$\sqrt{(5-1)^2 + (7-1)^2} = 7.21$
2	$\sqrt{(1-1.5)^2 + (1-2)^2} = 1.12$	$\sqrt{(5-1.5)^2 + (7-2)^2} = 6.10$
3	$\sqrt{(1-3)^2 + (1-4)^2} = 3.61$	$\sqrt{(5-3)^2 + (7-4)^2} = 3.61$
4	$\sqrt{(1-5)^2 + (1-7)^2} = 7.21$	$\sqrt{(5-5)^2 + (7-7)^2} = 0$
5	$\sqrt{(1-3.5)^2 + (1-5)^2} = 4.72$	$\sqrt{(5-3.5)^2 + (7-5)^2} = 2.5$
6	$\sqrt{(1-4.5)^2 + (1-5)^2} = 5.31$	$\sqrt{(5-4.5)^2 + (7-5)^2} = 2.06$
7	$\sqrt{(1-3.5)^2 + (1-4.5)^2} = 4.30$	$\sqrt{(5-3.5)^2 + (7-4.5)^2} = 2.92$

Step 2:

- Thus, we obtain two clusters containing:
  $\{1,2,3\}$ and $\{4,5,6,7\}$.
- Their new centroids are:

$$m_1 = \left(\frac{1}{3}(1.0+1.5+3.0), \frac{1}{3}(1.0+2.0+4.0)\right) = (1.83, 2.33)$$

$$m_2 = \left(\frac{1}{4}(5.0+3.5+4.5+3.5), \frac{1}{4}(7.0+5.0+5.0+4.5)\right)$$

$$= (4.12, 5.38)$$

$$d(m_1, 2) = \sqrt{|1.0-1.5|^2 + |1.0-2.0|^2} = 1.12$$

$$d(m_2, 2) = \sqrt{|5.0-1.5|^2 + |7.0-2.0|^2} = 6.10$$

Individual	Centroid 1	Centroid 2
1	0	7.21
2 (1.5, 2.0)	1.12	6.10
3	3.61	3.61
4	7.21	0
5	4.72	2.5
6	5.31	2.06
7	4.30	2.92

Step 3:

	Centroid 1	Centroid 2
1	$\sqrt{(1.83-1)^2+(2.33-1)^2}=1.57$	$\sqrt{(4.12-1)^2+(5.38-1)^2}=5.38$
2	$\sqrt{(1.83-1.5)^2+(2.33-2)^2}=0.47$	$\sqrt{(4.12-1.5)^2+(5.38-2)^2}=4.29$
3	$\sqrt{(1.83-3)^2+(2.33-4)^2}=2.04$	$\sqrt{(4.12-3)^2+(5.38-4)^2}=1.78$
4	$\sqrt{(1.83-5)^2+(2.33-7)^2}=5.64$	$\sqrt{(4.12-5)^2+(5.38-7)^2}=1.84$
5	$\sqrt{(1.83-3.5)^2+(2.33-5)^2}=3.15$	$\sqrt{(4.12-3.5)^2+(5.38-5)^2}=0.73$
6	$\sqrt{(1.83-4.5)^2+(2.33-5)^2}=3.78$	$\sqrt{(4.12-4.5)^2+(5.38-5)^2}=0.54$
7	$\sqrt{(1.83-3.5)^2+(2.33-4.5)^2}=2.74$	$\sqrt{(4.12-3.5)^2+(5.38-4.5)^2}=1.08$

Step 3:

Now using these centroids we compute the Euclidean distance of each object, as shown in table.

Therefore, the new clusters are:

{1,2} and {3,4,5,6,7}

Next centroids are:

m1=(1.25,1.5) and m2 = (3.9,5.1)

Individual	Centroid 1	Centroid 2
1	1.57	5.38
2	0.47	4.28
3	2.04	1.78
4	5.64	1.84
5	3.15	0.73
6	3.78	0.54
7	2.74	1.08

Step 4:

	Centroid 1	Centroid 2
1	$\sqrt{(1.25 - 1)^2 + (1.5 - 1)^2} = 0.58$	$\sqrt{(3.9 - 1)^2 + (5.1 - 1)^2} = 5.02$
2	$\sqrt{(1.25 - 1.5)^2 + (1.5 - 2)^2} = 0.56$	$\sqrt{(3.9 - 1.5)^2 + (5.1 - 2)^2} = 3.92$
3	$\sqrt{(1.25 - 3)^2 + (1.5 - 4)^2} = 3.05$	$\sqrt{(3.9 - 3)^2 + (5.1 - 4)^2} = 1.42$
4	$\sqrt{(1.25 - 5)^2 + (1.5 - 7)^2} = 6.66$	$\sqrt{(3.9 - 5)^2 + (5.1 - 7)^2} = 2.20$
5	$\sqrt{(1.25 - 3.5)^2 + (1.5 - 5)^2} = 4.16$	$\sqrt{(3.9 - 3.5)^2 + (5.1 - 5)^2} = 0.41$
6	$\sqrt{(1.25 - 4.5)^2 + (1.5 - 5)^2} = 4.78$	$\sqrt{(3.9 - 4.5)^2 + (5.1 - 5)^2} = 0.61$
7	$\sqrt{(1.25 - 3.5)^2 + (1.5 - 4.5)^2} = 3.75$	$\sqrt{(3.9 - 3.5)^2 + (5.1 - 4.5)^2} = 0.72$

Step 4 :

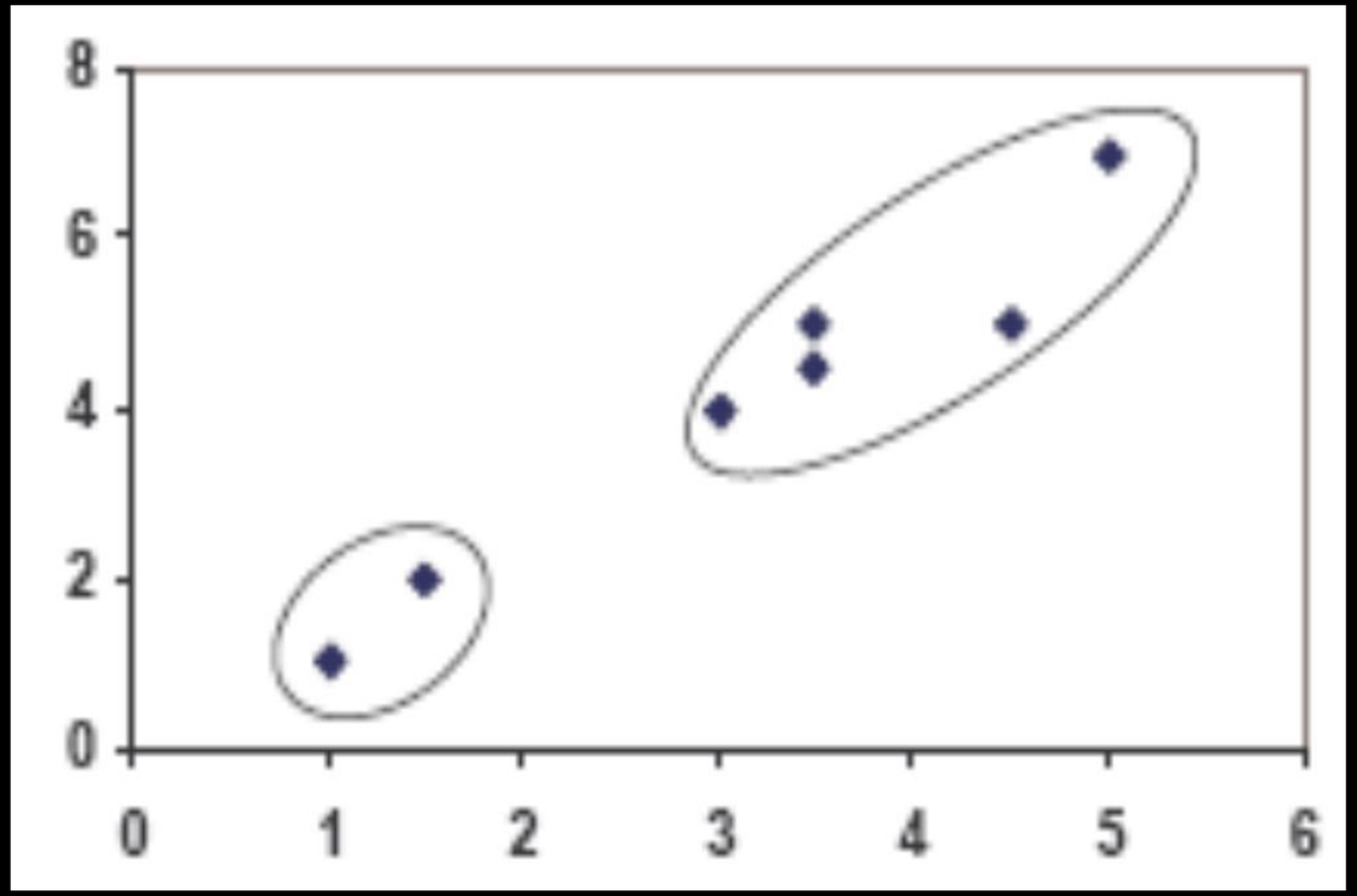
The clusters obtained are: $\{1,2\}$ and $\{3,4,5,6,7\}$

Therefore, there is no change in the cluster.

Thus, the algorithm comes to a halt here and final result consist of 2 clusters $\{1,2\}$ and $\{3,4,5,6,7\}$.

Individual	Centroid 1	Centroid 2
1	0.56	5.02
2	0. 56	3.92
3	3.05	1.42
4	6.66	2.20
5	4.16	0.41
6	4.78	0.61
7	3.75	0.72

PLOT



(with K=3)

Individual	$m_1 = 1$	$m_2 = 2$	$m_3 = 3$	cluster
1	0	1.11	3.61	1
2	1.12	0	2.5	2
3	3.61	2.5	0	3
4	7.21	6.10	3.61	3
5	4.72	3.61	1.12	3
6	5.31	4.24	1.80	3
7	4.30	3.20	0.71	3

C_3

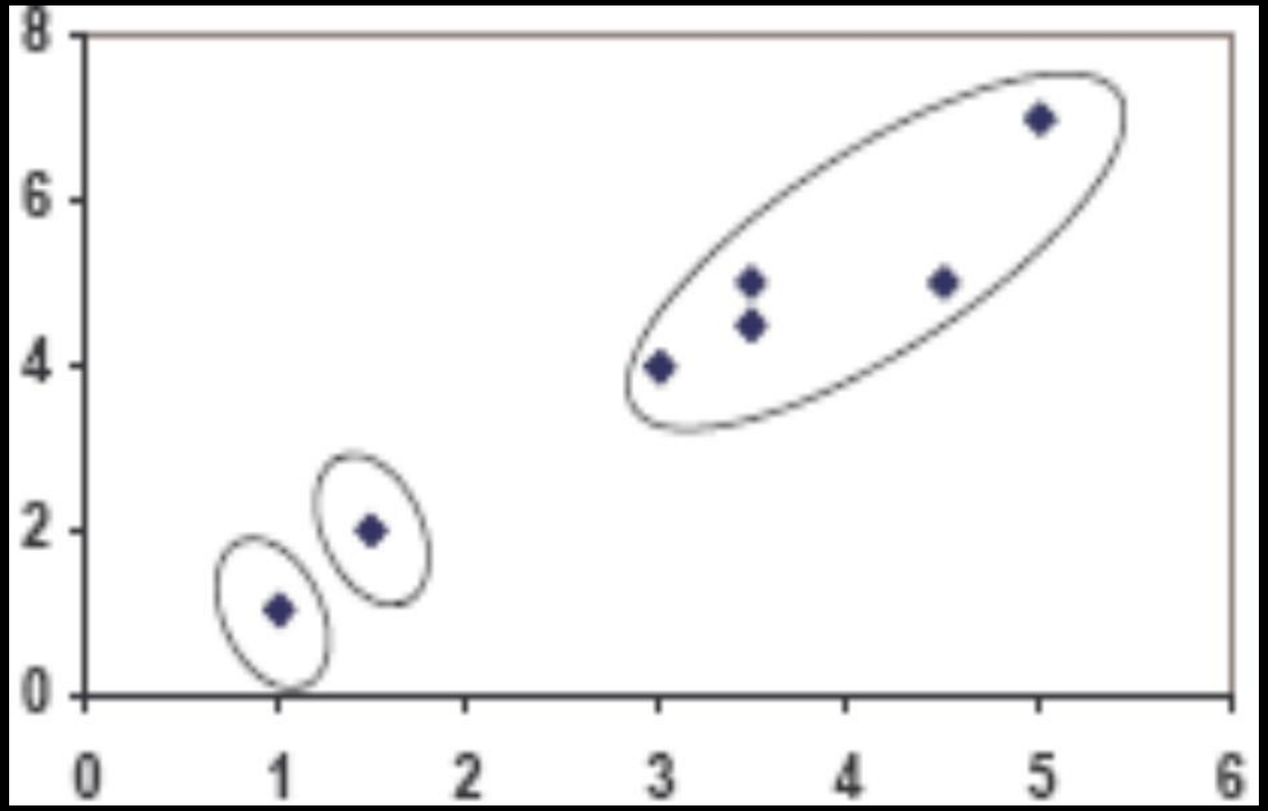
clustering with initial centroids (1, 2, 3)

Step 1

Individual	m_1 (1.0, 1.0)	m_2 (1.5, 2.0)	m_3 (3.9,5.1)	cluster
1	0	1.11	5.02	1
2	1.12	0	3.92	2
3	3.61	2.5	1.42	3
4	7.21	6.10	2.20	3
5	4.72	3.61	0.41	3
6	5.31	4.24	0.61	3
7	4.30	3.20	0.72	3

Step 2

PLOT



K-Means Clustering

```
# K-Means Clustering

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Mall_Customers.csv')
X = dataset.iloc[:, [3, 4]].values
from sklearn.preprocessing import StandardScaler # 0:1 MinMaxScaler //// ---> (z = (x - u) / s)
sc = StandardScaler()
X = sc.fit_transform(X)

# Using the elbow method to find the optimal number of clusters
#It measures the total squared distance of each sample to its closest cluster center.
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, random_state = 0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

# # Training the K-Means model on the dataset
kmeans = KMeans(n_clusters = 5, random_state = 0)
y_kmeans = kmeans.fit_predict(X)

# Visualising the clusters
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 10, c = 'red', label = 'Cluster 1')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Cluster 3')
plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 100, c = 'yellow', label = 'Centroids')
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```

Dataset Link

https://www.kaggle.com/datasets/shwetabh123/mall-customers

ENG. SHEREEN EBRAHIM

K-Nearest Neighbor(KNN)

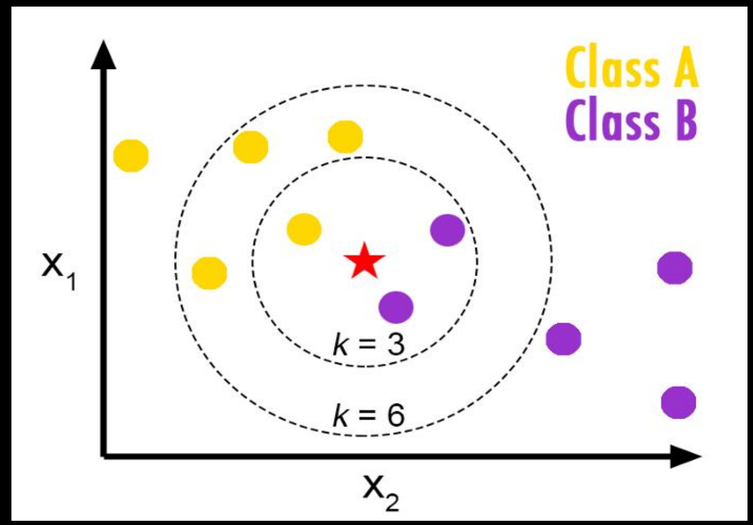
K-Nearest Neighbor(KNN)

- The K-NN algorithm works by finding the K nearest neighbors to a given data point based on a distance metric, such as Euclidean distance. The class or value of the data point is then determined by the majority vote or average of the K neighbors.

Euclidian distance: $d_E(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{i=1}^{D} (a_i - b_i)^2}$

Manhattan distance: $d_M(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{D} |a_i - b_i|$

$$D(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_{l=1}^{d} |x_{il} - x_{jl}|^{1/p} \right)^p .$$



Examples

Name	Cigarettes	Weight	Heart Attack
A	7	70	Bad
B	7	70	Bad
C	3	40	Good
D	1	40	Good

Test Data -> Name=E , Cigarettes=3, Weight=70, Heart Attack =??
Given K=1

Calculate using distance measure like Euclidean

$$d(x,y) = \sqrt[2]{(x1 - x2)^2 + (y1 - y2)^2}$$

So

- At Name A → distance =d(A,E)= $\sqrt[2]{(7-3)^2 + (70-70)^2} = 4$
- At Name B → distance =d(B,E)= $\sqrt[2]{(7-3)^2 + (40-70)^2} = 30.27$
- At Name C→ distance =d(C,E)= $\sqrt[2]{(3-3)^2 + (40-70)^2} = 30$
- At Name D → distance =d(D,E)= $\sqrt[2]{(1-3)^2 + (40-70)^2} = 30.07$

Examples

Name	Cigarettes	Weight	Heart Attack	Distance
A	7	70	Bad	4
B	7	70	Bad	30.27
C	3	40	Good	30
D	1	40	Good	30.07
E	3	70	?	

Then we Must Sort Distance

Examples

Name	Cigarettes	Weight	Heart Attack	Distance	Rank
A	7	70	Bad	4	1
C	3	40	Good	30	2
D	1	40	Good	30.07	3
B	7	70	Bad	30.27	4
E	3	70	?		

When we look first at smallest distance “4” we Heart Attack =Bad we can Predict that Heart Attack =Bad Based on smallest distance

KNN Project :

https://colab.research.google.com/drive/18QcZpOUdhlNeG_sylqaisTY6gyQu1MIq?usp=sharing

DataSet link:

https://www.kaggle.com/datasets/godfatherfigure/healthcare-dataset-stroke-data

Standardization vs. Normalization – Why & When to Use?

- **Standardization** (Z-score Normalization)
- Standardization transforms data such that it has a mean of 0 and a standard
- deviation of 1. This ensures that the values are centered around 0, which helps
- models interpret data effectively.

When to Use?

- When the data follows a Gaussian (normal) distribution
- o For linear regression, logistic regression, SVM, PCA, and neural networks
- o When dealing with outliers, as standardization is less affected by them
- compared to normalization

How to Apply Standardization in Scikit-Learn?

```python
from sklearn.preprocessing import StandardScaler
import numpy as np

# Sample data
data = np.array([[100, 2.5], [200, 3.0], [300, 3.5], [400, 4.0]])

# Apply standardization
scaler = StandardScaler()
standardized_data = scaler.fit_transform(data)

print(standardized_data)
```

Normalization (Min-Max Scaling)

- Normalization rescales data to a fixed range, typically [0,1] or [-1,1]. This ensures all
- values are on the same scale, which is crucial for certain machine learning models.

When to Use?

- When the data does not follow a normal distribution
- o For KNN, neural networks, and clustering algorithms like K-Means
- o When working with features that have different ranges (e.g., income in
- thousands vs. age in years)

How to Apply Normalization in Scikit-Learn?

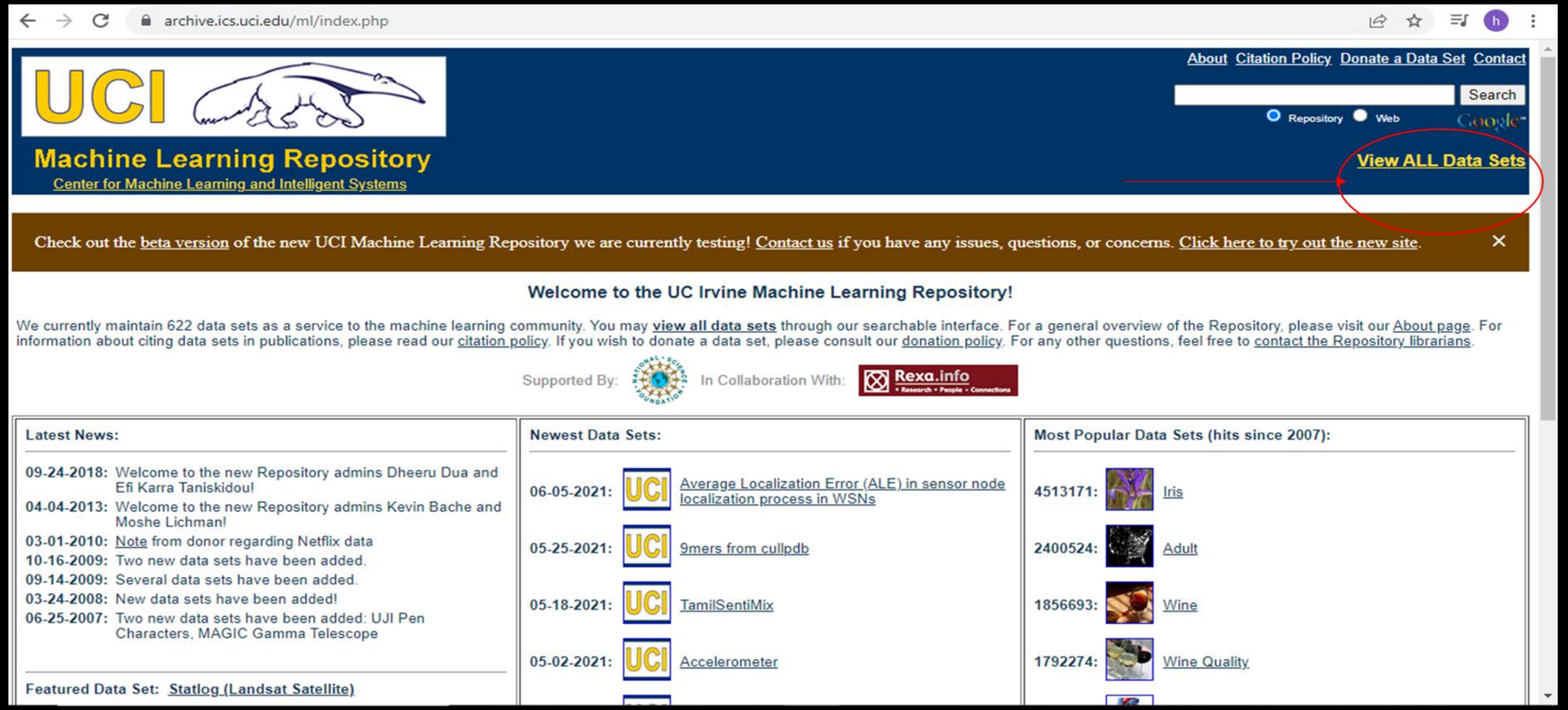
```
from sklearn.preprocessing import MinMaxScaler

# Apply normalization
scaler = MinMaxScaler()
normalized_data = scaler.fit_transform(data)

print(normalized_data)
```

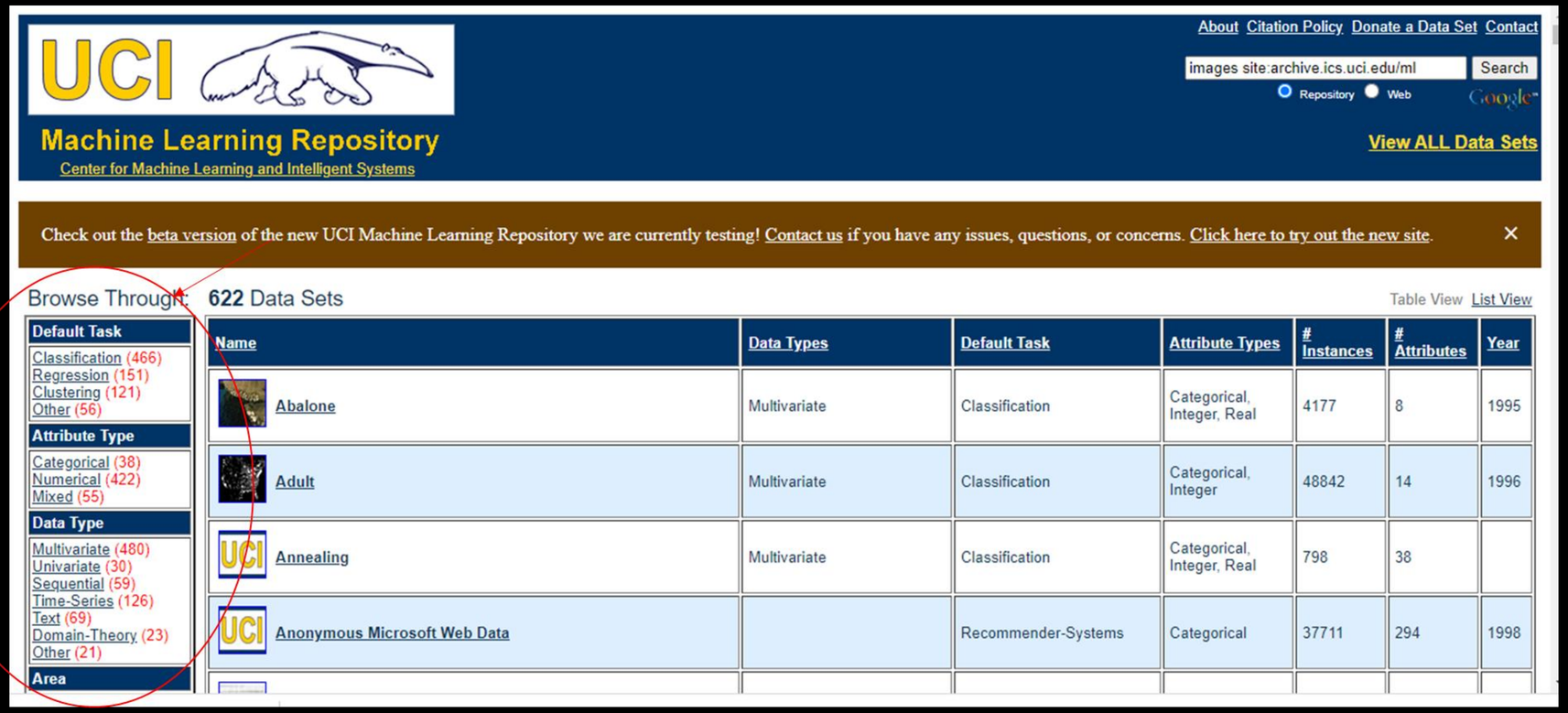
Getting dataset

We can download free data from

- UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets.php
- Kaggle datasets: https://www.kaggle.com/datasets
- Google dataset search: https://datasetsearch.research.google.com/

UCI Machine Learning Repository




UCI
Machine Learning Repository
Center for Machine Learning and Intelligent Systems
About Citation Policy Donate a Data Set Contact
images site:archive.ics.uci.edu/ml
Search
Repository Web
View ALL Data Sets
Check out the beta version of the new UCI Machine Learning Repository we are currently testing! Contact us if you have any issues, questions, or concerns. Click here to try out the new site.
Browse Through: 622 Data Sets
Table View List View
Default Task
Classification (466)
Regression (151)
Clustering (121)
Other (56)
Attribute Type
Categorical (38)
Numerical (422)
Mixed (55)
Data Type
Multivariate (480)
Univariate (30)
Sequential (59)
Time-Series (126)
Text (69)
Domain-Theory (23)
Other (21)
Area
Name
Data Types
Default Task
Attribute Types
Instances
Attributes
Year
Abalone
Multivariate
Classification
Categorical, Integer, Real
4177
8
1995
Adult
Multivariate
Classification
Categorical, Integer
48842
14
1996
Annealing
Multivariate
Classification
Categorical, Integer, Real
798
38
Anonymous Microsoft Web Data
Recommender-Systems
Categorical
37711
294
1998

Machine Learning Repository

Center for Machine Learning and Intelligent Systems

View ALL Data Sets

Check out the beta version of the new UCI Machine Learning Repository we are currently testing! Contact us if you have any issues, questions, or concerns. Click here to try out the new site. X

Abalone Data Set

Download: Data Folder, Data Set Description

Abstract: Predict the age of abalone from physical measurements

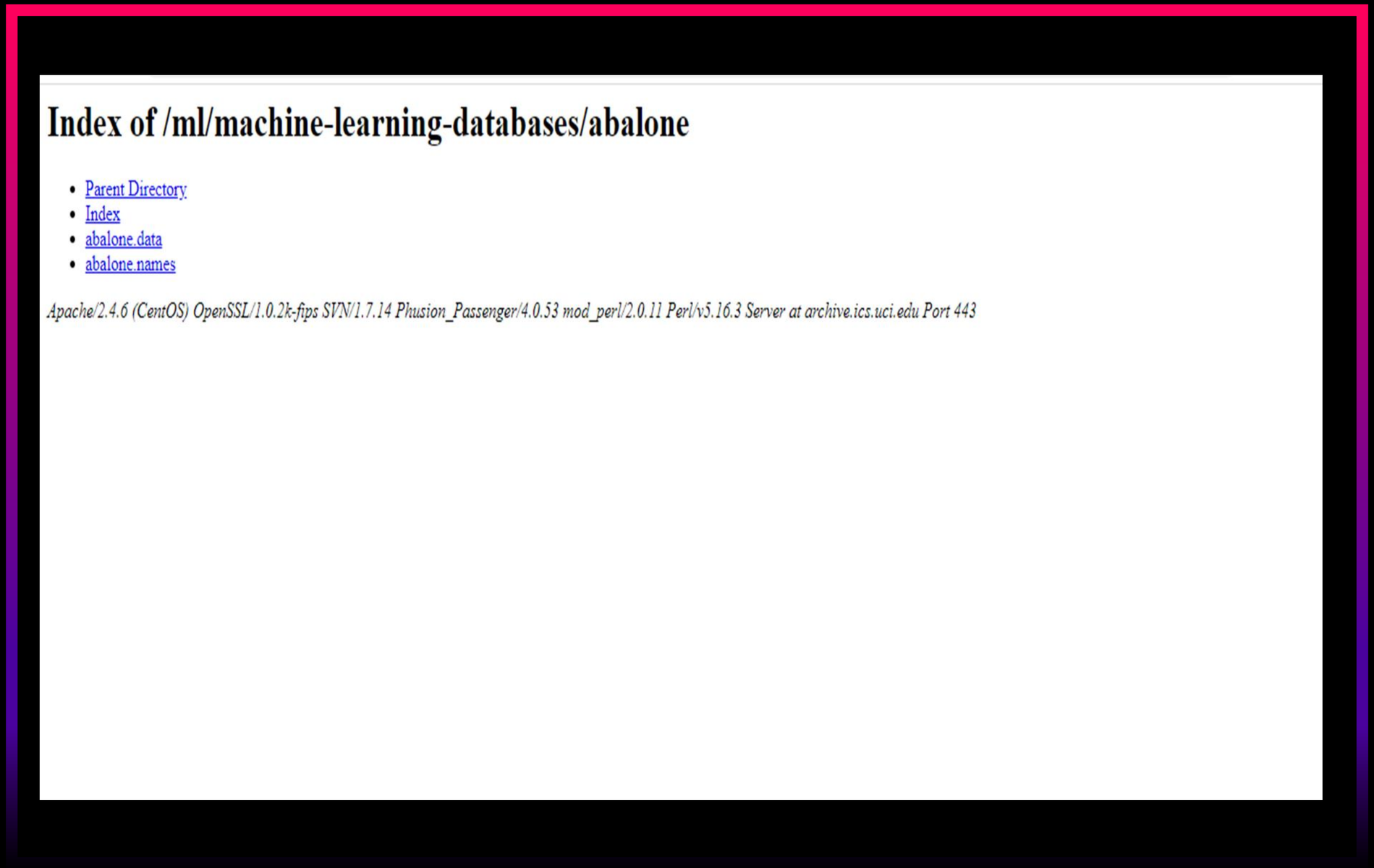
Data Set Characteristics:	Multivariate	**Number of Instances:**	4177	**Area:**	Life
Attribute Characteristics:	Categorical, Integer, Real	**Number of Attributes:**	8	**Date Donated**	1995-12-01
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	1223310

Source:

Data comes from an original (non-machine-learning) study:
Warwick J Nash, Tracy L Sellers, Simon R Talbot, Andrew J Cawthorn and Wes B Ford (1994)
"The Population Biology of Abalone (_Haliotis_ species) in Tasmania. I. Blacklip Abalone (_H. rubra_) from the North Coast and Islands of Bass Strait",
Sea Fisheries Division, Technical Report No. 48 (ISSN 1034-3288)

segmentation.names

Show all

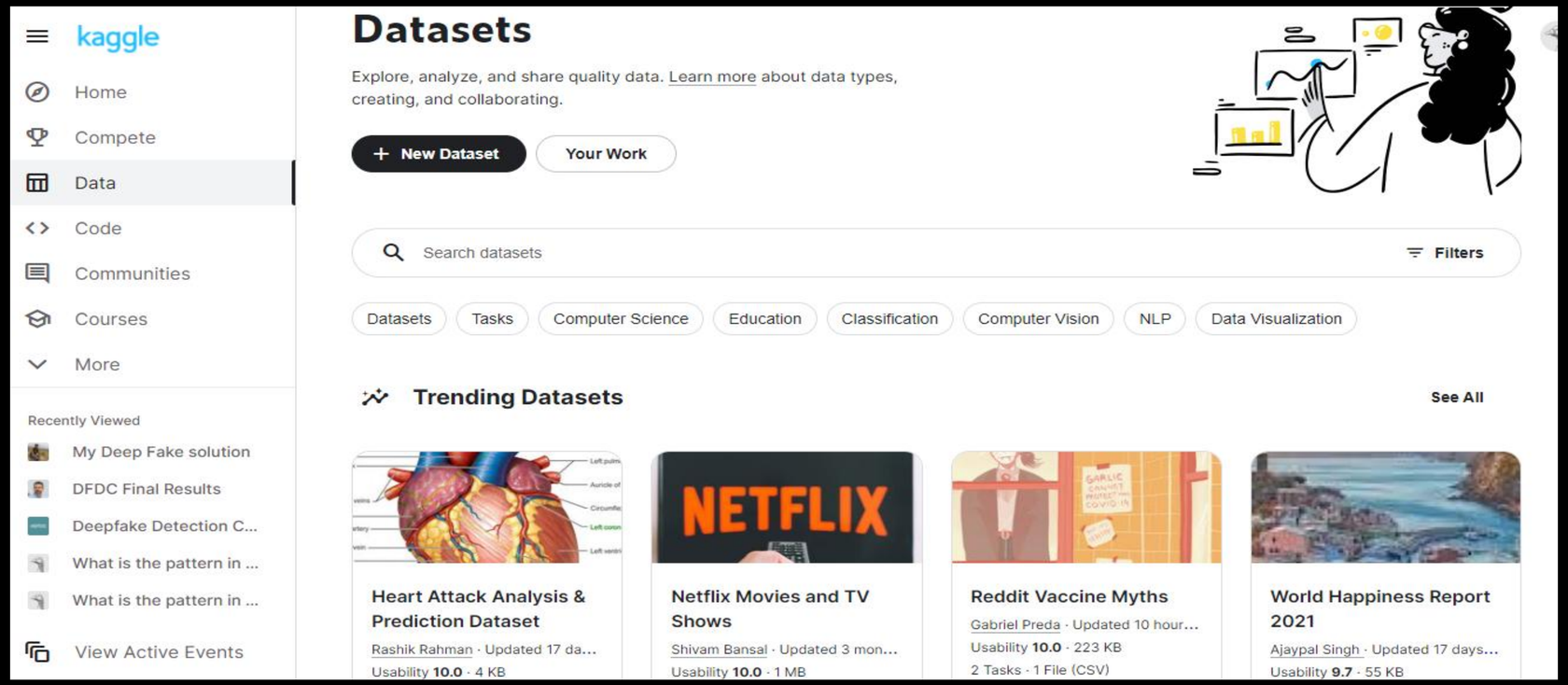


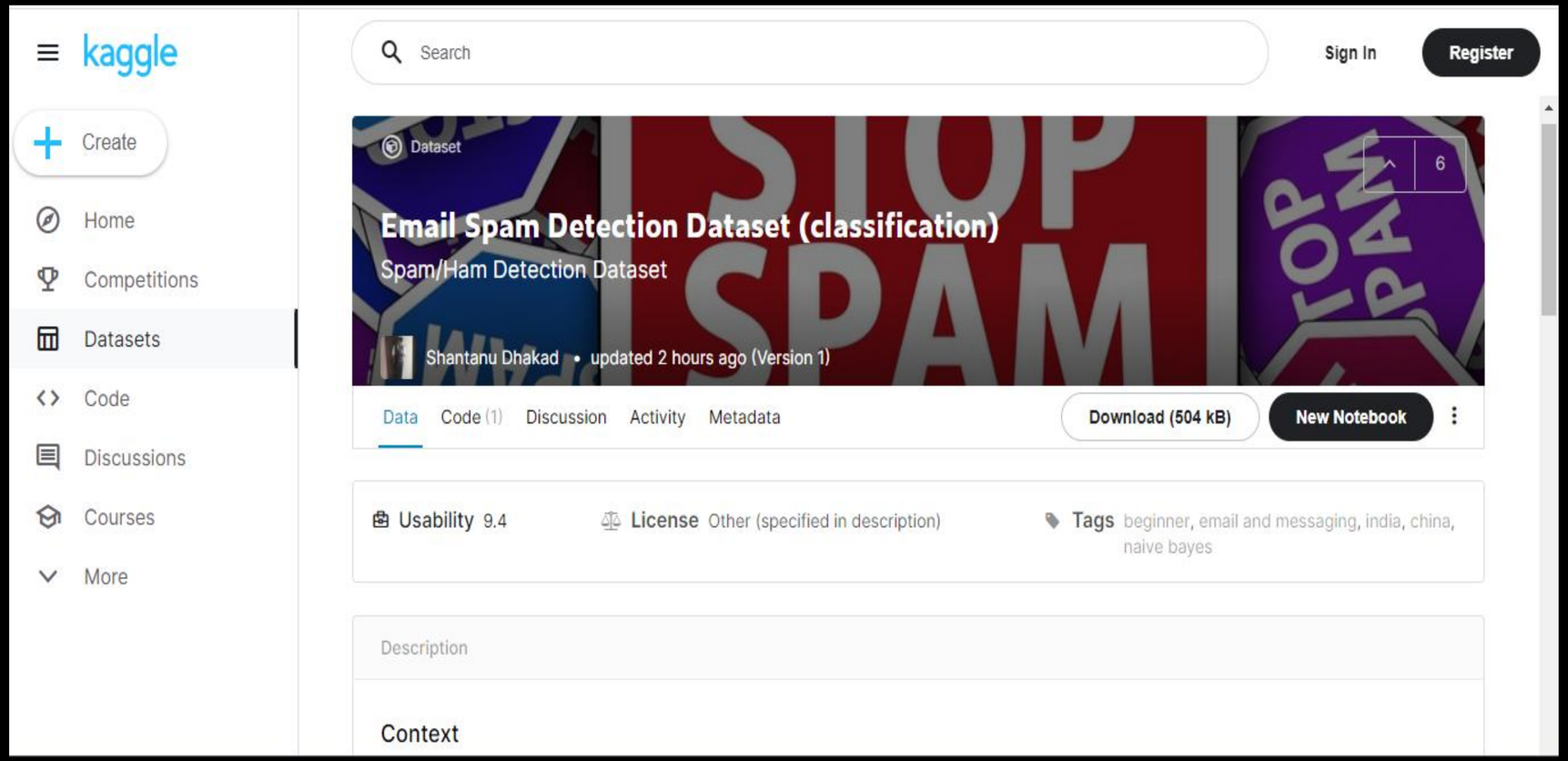
Index of /ml/machine-learning-databases/abalone

- Parent Directory
- Index
- abalone.data
- abalone.names

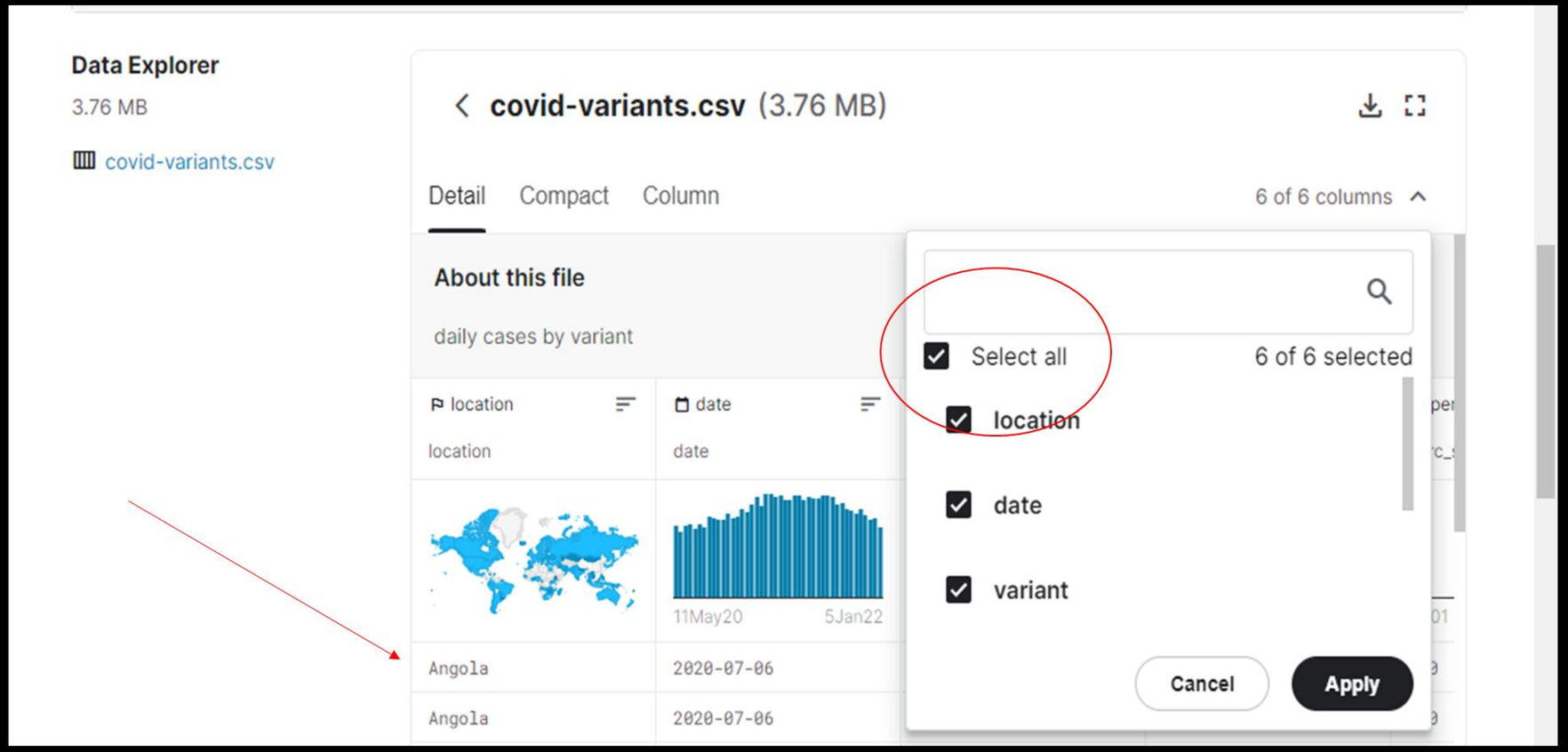
Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips SVN/1.7.14 Phusion_Passenger/4.0.53 mod_perl/2.0.11 Perl/v5.16.3 Server at archive.ics.uci.edu Port 443

Kaggle datasets

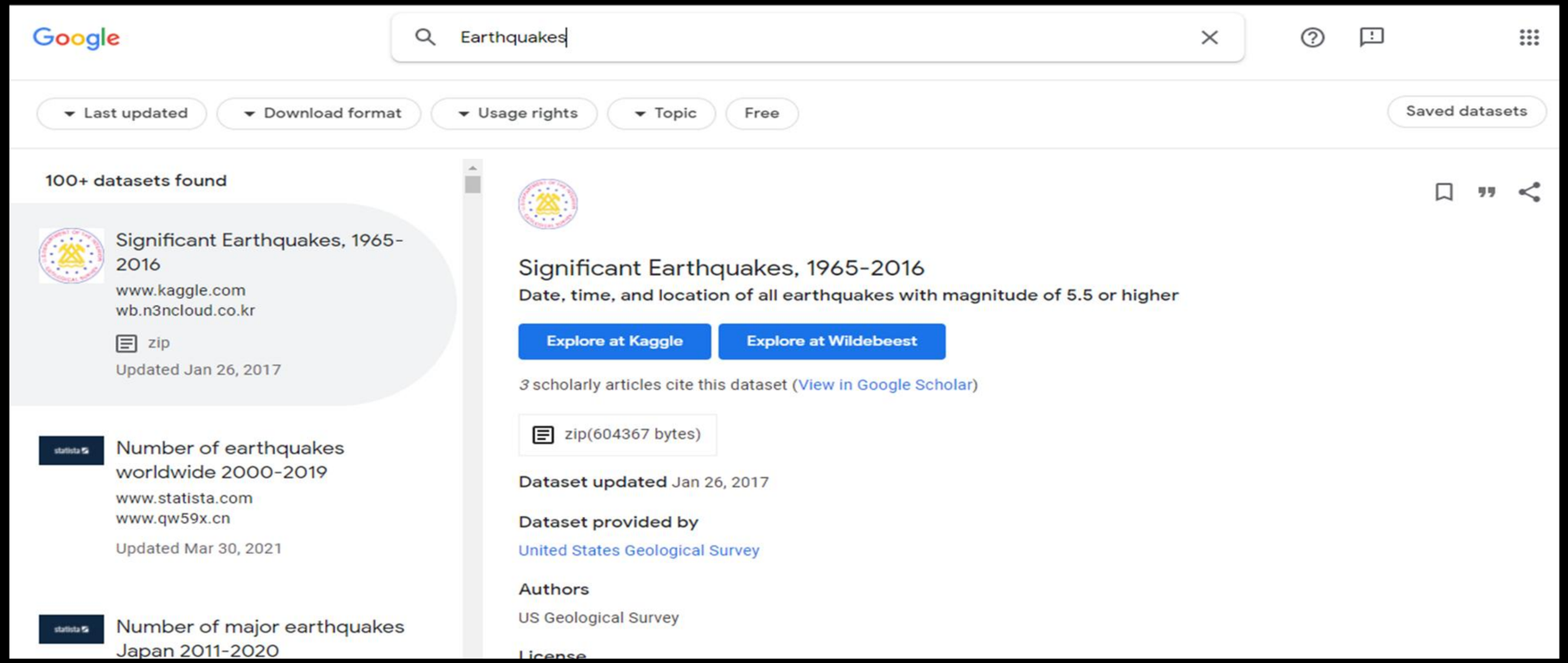



kaggle
Search
Sign In
Register
Create
Home
Competitions
Datasets
Code
Discussions
Courses
More
Dataset
6
Email Spam Detection Dataset (classification)
Spam/Ham Detection Dataset
Shantanu Dhakad • updated 2 hours ago (Version 1)
Data
Code (1)
Discussion
Activity
Metadata
Download (504 kB)
New Notebook
Usability 9.4
License Other (specified in description)
Tags beginner, email and messaging, india, china, naive bayes
Description
Context

Inceptum Egypt



Google dataset search



See you in the next session

INCEPTUM
EGYPT