|  |  |
| --- | --- |
| Algorithm: K means clustering | |
| USN : 1MS17CS143 | NAME : Sathvik K P |
| USN : 1MS17CS148 | NAME : Sathvik B |

**Description of the Algorithm:**

Clustering is one of the most common exploratory data analysis technique used to get an intuition about the structure of the data. It can be defined as the task of identifying subgroups in the data such that data points in the same subgroup (cluster) are very similar while data points in different clusters are very different. In other words, we try to find homogeneous subgroups within the data such that data points in each cluster are as similar as possible according to a similarity measure such as euclidean-based distance or correlation-based distance. The decision of which similarity measure to use is application-specific.

Clustering analysis can be done on the basis of features where we try to find subgroups of samples based on features or on the basis of samples where we try to find subgroups of features based on samples. We’ll cover here clustering based on features. Clustering is used in market segmentation; where we try to fined customers that are similar to each other whether in terms of behaviors or attributes, image segmentation/compression; where we try to group similar regions together, document clustering based on topics, etc.

Unlike supervised learning, clustering is considered an unsupervised learning method since we don’t have the ground truth to compare the output of the clustering algorithm to the true labels to evaluate its performance. We only want to try to investigate the structure of the data by grouping the data points into distinct subgroups.

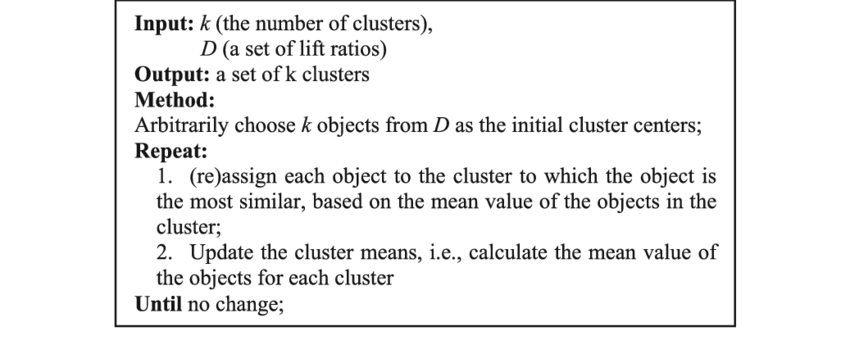
Kmeans algorithm is an iterative algorithm that tries to partition the dataset into Kpre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster’s centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

The way kmeans algorithm works is as follows:

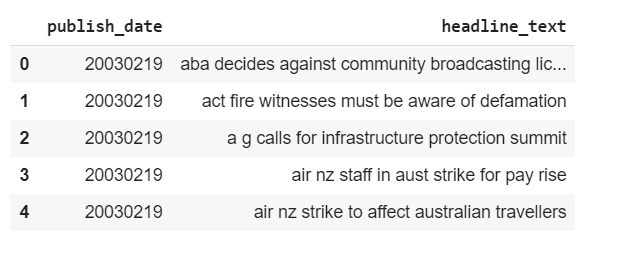
1. Specify number of clusters K.
2. Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement.
3. Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn’t changing.
4. Compute the sum of the squared distance between data points and all centroids.
5. Assign each data point to the closest cluster (centroid).
6. Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.

The approach kmeans follows to solve the problem is called Expectation-Maximization.

**Algorithm Pseudocode:**



**Data set Used: (Attach Screen shot of the few rows)**



**Challenges faced during the implementation of the program:**

1. Determining optimal number of clusters
2. Data sampling (otherwise high computation time)
3. Data pre-processing is necessary to remove stop words and for stemming
4. Dataset contains duplicates. So additional pre-processing steps had to be taken.

**Code:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.feature\_extraction import text

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.cluster import KMeans

from nltk.tokenize import RegexpTokenizer

from nltk.stem.snowball import SnowballStemmer

%matplotlib inline

data = pd.read\_csv("/content/drive/My Drive/Datasets/abcnews-date-text.csv",error\_bad\_lines=False)

data.head()

data['publish\_year'] = data['publish\_date'].apply(lambda x:int(x/10000))

data['publish\_month'] = data['publish\_date'].apply(lambda x:int(((x)%10000)/100))

data['publish\_day'] = data['publish\_date'].apply(lambda x:((x)%10000)%100)

import matplotlib.pyplot as plt

plt.hist(data['publish\_year'], facecolor='blue', alpha=0.8, rwidth = 0.5)

plt.xlabel('Year')

plt.ylabel('#News Headlines')

plt.title('#News Headlines in each year')

plt.show()

plt.hist(data['publish\_month'],12, facecolor='blue', alpha=0.8, rwidth = 0.5)

plt.xlabel('Month')

plt.ylabel('#News Headlines')

plt.title('#News Headlines in each month')

plt.show()

plt.hist(data['publish\_day'],31, facecolor='blue', alpha=0.8, rwidth = 0.5)

plt.xlabel('Day')

plt.ylabel('#News Headlines')

plt.title('#News Headlines on each day of month')

plt.show()

data = data[:10000]

data = data[["headline\_text"]]

data[data['headline\_text'].duplicated(keep=False)].sort\_values('headline\_text').head(8)

data = data.drop\_duplicates('headline\_text')

punc = ['.', ',', '"', "'", '?', '!', ':', ';', '(', ')', '[', ']', '{', '}',"%"]

stop\_words = text.ENGLISH\_STOP\_WORDS.union(punc)

desc = data['headline\_text'].values

vectorizer = TfidfVectorizer(stop\_words = stop\_words)

X = vectorizer.fit\_transform(desc)

word\_features = vectorizer.get\_feature\_names()

print(len(word\_features))

stemmer = SnowballStemmer('english')

tokenizer = RegexpTokenizer(r'[a-zA-Z\']+')

def tokenize(text):

    return [stemmer.stem(word) for word in tokenizer.tokenize(text.lower())]

vectorizer2 = TfidfVectorizer(stop\_words = stop\_words, tokenizer = tokenize)

X2 = vectorizer2.fit\_transform(desc)

word\_features2 = vectorizer2.get\_feature\_names()

print(len(word\_features2))

vectorizer3 = TfidfVectorizer(stop\_words = stop\_words, tokenizer = tokenize, max\_features = 100)

X3 = vectorizer3.fit\_transform(desc)

words = vectorizer3.get\_feature\_names()

from sklearn.cluster import KMeans

wcss = []

for i in range(1,11):

    kmeans = KMeans(n\_clusters=i,init='k-means++',max\_iter=300,n\_init=10,random\_state=0)

    kmeans.fit(X3)

    wcss.append(kmeans.inertia\_)

plt.plot(range(1,11),wcss)

plt.title('The Elbow Method')

plt.xlabel('Number of clusters')

plt.ylabel('WCSS')

plt.savefig('elbow.png')

plt.show()

kmeans = KMeans(n\_clusters = 3, n\_init = 20, n\_jobs = 1) # n\_init(number of iterations for clsutering) n\_jobs(number of cpu cores to use)

kmeans.fit(X3)

# We look at 3 the clusters generated by k-means.

common\_words = kmeans.cluster\_centers\_.argsort()[:,-1:-26:-1]

for num, centroid in enumerate(common\_words):

    print(str(num) + ' : ' + ', '.join(words[word] for word in centroid))

from sklearn.metrics import silhouette\_score

print(silhouette\_score(X3, labels=kmeans.predict(X3)))

kmeans = KMeans(n\_clusters = 5, n\_init = 20, n\_jobs = 1) # n\_init(number of iterations for clsutering) n\_jobs(number of cpu cores to use)

kmeans.fit(X3)

# We look at 3 the clusters generated by k-means.

common\_words = kmeans.cluster\_centers\_.argsort()[:,-1:-26:-1]

for num, centroid in enumerate(common\_words):

    print(str(num) + ' : ' + ', '.join(words[word] for word in centroid))

from sklearn.metrics import silhouette\_score

print(silhouette\_score(X3, labels=kmeans.predict(X3)))

kmeans = KMeans(n\_clusters = 6, n\_init = 20, n\_jobs = 1) # n\_init(number of iterations for clsutering) n\_jobs(number of cpu cores to use)

kmeans.fit(X3)

# We look at 3 the clusters generated by k-means.

common\_words = kmeans.cluster\_centers\_.argsort()[:,-1:-26:-1]

for num, centroid in enumerate(common\_words):

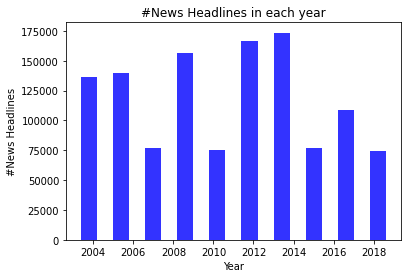
    print(str(num) + ' : ' + ', '.join(words[word] for word in centroid))

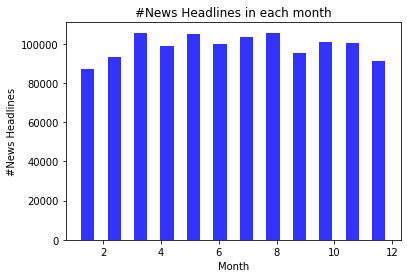
from sklearn.metrics import silhouette\_score

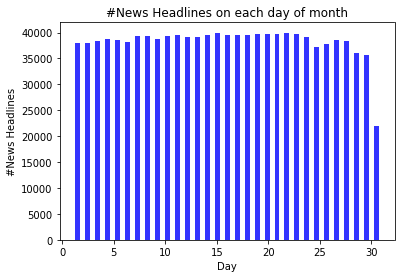
print(silhouette\_score(X3, labels=kmeans.predict(X3)))

**Output: (Screen shots)**

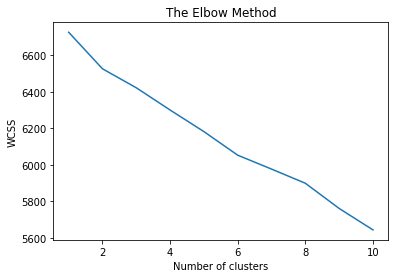
Exploratory Data Analysis



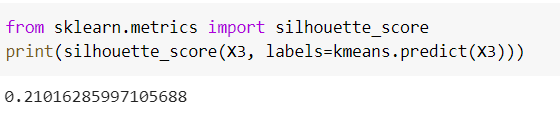
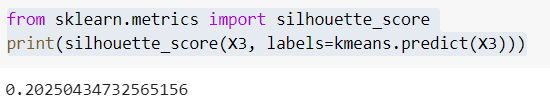
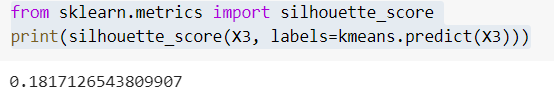




Determining number of clusters using Elbow Method



Results of silhouette scores for number of clusters = 3,5 and 6



**References:**

1. <https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a>
2. <https://en.wikipedia.org/wiki/K-means_clustering>
3. <https://towardsdatascience.com/k-means-clustering-introduction-to-machine-learning-algorithms-c96bf0d5d57a>
4. <https://www.researchgate.net/figure/The-pseudo-code-for-K-means-clustering-algorithm_fig2_273063437>