B.TECH. (2020-24)
Artificial Intelligence

Open Ended
LAB File
on
FUNDAMENTALS OF MACHINE LEARNING
[CSE313]

Submitted To
Dr Monika Arora

Submitted By

HITESH	**A023119820027**	**5AI 1**
GAURI TYAGI	**A023119820028**	**5AI 1**
KUSHAGRA DUBEY	**A023119820029**	**5AI 1**
BHOOMIKA SHARMA	**A023119820030**	**5AI 1**
SUNIDHI SINGH	**A023119820032**	**5AI 1**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY
AMITY UNIVERSITY UTTAR PRADESH
NOIDA (U.P)

OPEN ENDED EXPERIMENT

Aim

To implement k means clustering algorithm over a dataset

Software Used

Google Colab

Program Code and Output

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # for data visualization
import seaborn as sns # for statistical data visualization
%matplotlib inline
```

```
data = 'Live.csv'

df = pd.read_csv(data)
```

```
df.shape
```

```
(7050, 16)
```

```
df.head()
```

status_type	status_published	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	num_hahas	num_sads	num_angrys	Column1
video	4/22/2018 6:00	529	512	262	432	92	3	1	1	0	NaN
photo	4/21/2018 22:45	150	0	0	150	0	0	0	0	0	NaN
video	4/21/2018 6:17	227	236	57	204	21	1	1	0	0	NaN
photo	4/21/2018 2:29	111	0	0	111	0	0	0	0	0	NaN
photo	4/18/2018 3:22	213	0	0	204	9	0	0	0	0	NaN

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7050 entries, 0 to 7049
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   status_id         7050 non-null   object
 1   status_type       7050 non-null   object
 2   status_published  7050 non-null   object
 3   num_reactions     7050 non-null   int64
 4   num_comments      7050 non-null   int64
 5   num_shares        7050 non-null   int64
 6   num_likes         7050 non-null   int64
 7   num_loves         7050 non-null   int64
 8   num_wows          7050 non-null   int64
 9   num_hahas         7050 non-null   int64
 10  num_sads          7050 non-null   int64
 11  num_angrys        7050 non-null   int64
 12  Column1           0 non-null      float64
 13  Column2           0 non-null      float64
 14  Column3           0 non-null      float64
 15  Column4           0 non-null      float64
dtypes: float64(4), int64(9), object(3)
memory usage: 881.4+ KB
```

```
df.isnull().sum()
```

```
status_id               0
status_type             0
status_published        0
num_reactions           0
num_comments            0
num_shares              0
num_likes               0
num_loves               0
num_wows                0
num_hahas               0
num_sads                0
num_angrys              0
Column1              7050
Column2              7050
Column3              7050
Column4              7050
dtype: int64
```

```
df.drop(['Column1', 'Column2', 'Column3', 'Column4'], axis=1, inplace=True)
```

```
df.describe()
```

	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	num_hahas	num_sads	num_angrys
count	7050.000000	7050.000000	7050.000000	7050.000000	7050.000000	7050.000000	7050.000000	7050.000000	7050.000000
mean	230.117163	224.356028	40.022553	215.043121	12.728652	1.289362	0.696454	0.243688	0.113191
std	462.625309	889.636820	131.599965	449.472357	39.972930	8.719650	3.957183	1.597156	0.726812
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	17.000000	0.000000	0.000000	17.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	59.500000	4.000000	0.000000	58.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	219.000000	23.000000	4.000000	184.750000	3.000000	0.000000	0.000000	0.000000	0.000000
max	4710.000000	20990.000000	3424.000000	4710.000000	657.000000	278.000000	157.000000	51.000000	31.000000

```
df.corr()
```

	status_type	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	num_hahas	num_sads	num_angrys
status_type	1.000000	0.102860	0.320975	0.390910	0.067423	0.388612	0.093844	0.177903	0.081233	0.130989
num_reactions	0.102860	1.000000	0.150843	0.250723	0.994923	0.305003	0.267752	0.176028	0.075138	0.124326
num_comments	0.320975	0.150843	1.000000	0.640637	0.101687	0.521223	0.162394	0.325048	0.236453	0.225184
num_shares	0.390910	0.250723	0.640637	1.000000	0.172492	0.820000	0.407628	0.399826	0.199970	0.312513
num_likes	0.067423	0.994923	0.101687	0.172492	1.000000	0.209308	0.207800	0.120784	0.052169	0.087431
num_loves	0.388612	0.305003	0.521223	0.820000	0.209308	1.000000	0.508798	0.507830	0.207600	0.371001
num_wows	0.093844	0.267752	0.162394	0.407628	0.207800	0.508798	1.000000	0.287756	0.086503	0.183087
num_hahas	0.177903	0.176028	0.325048	0.399826	0.120784	0.507830	0.287756	1.000000	0.141421	0.211910
num_sads	0.081233	0.075138	0.236453	0.199970	0.052169	0.207600	0.086503	0.141421	1.000000	0.142072
num_angrys	0.130989	0.124326	0.225184	0.312513	0.087431	0.371001	0.183087	0.211910	0.142072	1.000000

```
df['status_id'].unique()
```

```
array(['246675545449582_1649696485147474',
       '246675545449582_1649426988507757',
       '246675545449582_1648730588577397', ...,
       '1050855161656896_1060126464063099',
       '1050855161656896_1058663487542730',
       '1050855161656896_1050858841656528'], dtype=object)
```

```
len(df['status_id'].unique())
```

```
6997
```

```
df['status_published'].unique()
```

```
array(['4/22/2018 6:00', '4/21/2018 22:45', '4/21/2018 6:17', ...,
       '9/21/2016 23:03', '9/20/2016 0:43', '9/10/2016 10:30'],
      dtype=object)
```

```
len(df['status_published'].unique())
```

```
6913
```

```
df.drop(['status_id', 'status_published'], axis=1, inplace=True)
```

```
df.head()
```

	status_type	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	num_hahas	num_sads	num_angrys
0	video	529	512	262	432	92	3	1	1	0
1	photo	150	0	0	150	0	0	0	0	0
2	video	227	236	57	204	21	1	1	0	0
3	photo	111	0	0	111	0	0	0	0	0
4	photo	213	0	0	204	9	0	0	0	0

```
X = df

y = df['status_type']
```

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

X['status_type'] = le.fit_transform(X['status_type'])

y = le.transform(y)
```

```
cols = X.columns
```

```
from sklearn.preprocessing import MinMaxScaler

ms = MinMaxScaler()

X = ms.fit_transform(X)
```

```
X = pd.DataFrame(X, columns=[cols])
```

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=2, random_state=0)

kmeans.fit(X)
```

```
KMeans(n_clusters=2, random_state=0)
```

```
kmeans.cluster_centers_
```

```
array([[3.28506857e-01, 3.90710874e-02, 7.54854864e-04, 7.53667113e-04,
        3.85438884e-02, 2.17448568e-03, 2.43721364e-03, 1.20039760e-03,
        2.75348016e-03, 1.45313276e-03],
       [9.54921576e-01, 6.46330441e-02, 2.67028654e-02, 2.93171709e-02,
        5.71231462e-02, 4.71007076e-02, 8.18581889e-03, 9.65207685e-03,
        8.04219428e-03, 7.19501847e-03]])
```

```
kmeans.inertia_
```

```
237.7572640441955
```

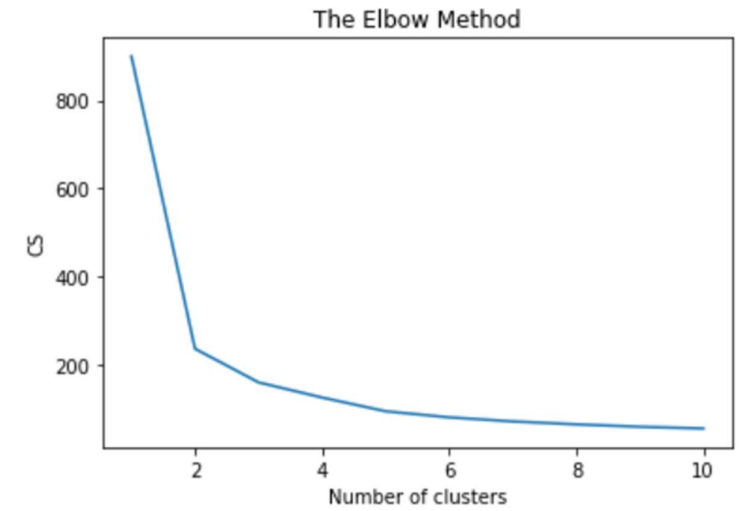
```
labels = kmeans.labels_

# check how many of the samples were correctly labeled
correct_labels = sum(y == labels)

print("Result: %d out of %d samples were correctly labeled." % (correct_labels, y.size))
```

```
Result: 63 out of 7050 samples were correctly labeled.
```

```
print('Accuracy score: {0:0.2f}'. format(correct_labels/float(y.size)))
```

```
Accuracy score: 0.01
```

```
from sklearn.cluster import KMeans
cs = []
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(X)
    cs.append(kmeans.inertia_)
plt.plot(range(1, 11), cs)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('CS')
plt.show()
```



```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=2,random_state=0)

kmeans.fit(X)

labels = kmeans.labels_

# check how many of the samples were correctly labeled

correct_labels = sum(y == labels)

print("Result: %d out of %d samples were correctly labeled." % (correct_labels, y.size))

print('Accuracy score: {0:0.2f}'. format(correct_labels/float(y.size)))
```

```
Result: 63 out of 7050 samples were correctly labeled.
Accuracy score: 0.01
```

```
kmeans = KMeans(n_clusters=3, random_state=0)

kmeans.fit(X)

# check how many of the samples were correctly labeled
labels = kmeans.labels_

correct_labels = sum(y == labels)
print("Result: %d out of %d samples were correctly labeled." % (correct_labels, y.size))
print('Accuracy score: {0:0.2f}'. format(correct_labels/float(y.size)))
```

```
Result: 138 out of 7050 samples were correctly labeled.
Accuracy score: 0.02
```

```
kmeans = KMeans(n_clusters=4, random_state=0)

kmeans.fit(X)

# check how many of the samples were correctly labeled
labels = kmeans.labels_

correct_labels = sum(y == labels)
print("Result: %d out of %d samples were correctly labeled." % (correct_labels, y.size))
print('Accuracy score: {0:0.2f}'. format(correct_labels/float(y.size)))
```

```
Result: 4340 out of 7050 samples were correctly labeled.
Accuracy score: 0.62
```

```
kmeans = KMeans(n_clusters=5, random_state=0)

kmeans.fit(X)

# check how many of the samples were correctly labeled
labels = kmeans.labels_

correct_labels = sum(y == labels)
print("Result: %d out of %d samples were correctly labeled." % (correct_labels, y.size))
print('Accuracy score: {0:0.2f}'. format(correct_labels/float(y.size)))
```

```
Result: 82 out of 7050 samples were correctly labeled.
Accuracy score: 0.01
```

Discussion and Conclusion

The lesser the model inertia, the better the model fit. So, we use the Use elbow method to find optimal number of clusters. There is a kink at k=2.Hence k=2 can be considered a good number of the cluster to cluster this data. So, we have changed the value of k and found relatively higher classification accuracy of 62% with k=4. Hence, we can conclude that k=4 being the optimal number of clusters.

CRITERIA	TOTAL MARKS	MARKS OBTAINED	COMMENTS
Concept (A)	2		
Implementation (B)	2		
Performance (C)	2		
Total	6		