ANOMALY DETECTION FROM TWEETS BY K-MEANS CLUSTERING

Project By

18BIT0126

18BIT0148

SUBMITTED

TO

PARIMALA.M

SUMMARY

Anomaly/Outlier detection is one of very popular topic in ML world. It comes under 'unsupervised learning' process. Here we don't have any prior knowledge of the data patterns unlike 'supervised learning'. 'Anomaly or Outlier' is that data point which is not that much similar with other data points in our entire data set.

Now a days, we are spending the most of the time online in the social media. The example for such social media is Twitter, Facebook and Instagram. Here in our project we are gone help the users to identify the how far a tweet is true or else it was an anomaly.

I will use Python, Sci-kit learn, 'pyod' library from 'pypi.org', Gensim, NLTK for solution of this problem.

- Data is from Kaggle Donald_Tweets we use https://www.kaggle.com/austinreese/trump-tweets
- After getting the data we now have to select the data on which we are going to work
- After that we convert the text to vector by doc2vec function
- After that we use PCA to identify principal components
- After that we cluster the data
- After that we find the *Silhoutte Score*
- After that we sort the scores
- Find the least scores that are nearer to zero
- Then we go for anomaly detection by basing on the *Silhoutte Score* if *Silhoutte Score* = 0
- Finally we will be able to view anomaly tweets.

In [1]:
```
# IMPORTING THE LIBRARIES THAT ARE ALL NEEDED
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from scipy.sparse import csr_matrix
from mpl_toolkits.axes_grid1 import make_axes_locatable

from sklearn.cluster import KMeans
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error
from sklearn.base import BaseEstimator
from sklearn import utils
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score

from gensim.models.doc2vec import TaggedDocument, Doc2Vec
from gensim.parsing.preprocessing import preprocess_string


import itertools
#here the silhouette_score helps us to effectively define the clusters
from sklearn.metrics import silhouette_samples, silhouette_score
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
#reading the csv from given location
all_tweets=pd.read_csv(r'C:\Users\narur\Desktop\Donald.csv')
#getting usable things from csv
atweets=all_tweets['Tweet_Text'];
#printing the top tweets
atweets.head()
```

Out[2]:
```
0    Today we express our deepest gratitude to all ...
1    Busy day planned in New York. Will soon be mak...
2    Love the fact that the small groups of protest...
3    Just had a very open and successful presidenti...
4    A fantastic day in D.C. Met with President Oba...
Name: Tweet_Text, dtype: object
```

In [3]:
```
#prnting all the twets for reference
all_tweets.head()
```

Out[3]:

	Index	Time	Tweet_Text	Type	Media_Type	Hashtags	Tweet_Id	
0	0	15:26:37	Today we express our deepest gratitude to all ...	text	photo	ThankAVet	7.970000e+17	https://twitter.com/realI
1	1	13:33:35	Busy day planned in New York. Will soon be mak...	text	NaN	NaN	7.970000e+17	https://twitter.com/realI
2	2	11:14:20	Love the fact that the small groups of protest...	text	NaN	NaN	7.970000e+17	https://twitter.com/realI
3	3	2:19:44	Just had a very open and successful presidenti...	text	NaN	NaN	7.970000e+17	https://twitter.com/realI
4	4	2:10:46	A fantastic day in D.C. Met with President Oba...	text	NaN	NaN	7.970000e+17	https://twitter.com/realI

In [4]:
```
import multiprocessing
from multiprocessing import *
from tqdm import tqdm
#doc2vec to convert the tweets to vectors
#training the doc2vectransform
# estimators specify all the parameters that can be set at the class level in
 their __init__
#as explicit keyword arguments
#preprocessing would be converting the each text tweet into array tokens & rem
oval of unwanted characters, stop words etc.
#conversion of each array of tokens corresponding to each text tweet into nume
rical vectors .
#Vector Space Model generation
#It is recommended to keep ‘Doc2Vec’ vector size from 100 to 300
class Doc2VecTransformer(BaseEstimator):

    def __init__(self, vector_size=100, learning_rate=0.02, epochs=20):
        self.learning_rate = learning_rate
        self.epochs = epochs
        self._model = None
        self.vector_size = vector_size
        self.workers =(multiprocessing.cpu_count())-1

    def fit(self, x, y=None):
        tagged_x = [TaggedDocument(preprocess_string(item), [index]) for index
, item in enumerate(x)]
        model = Doc2Vec(documents=tagged_x, vector_size=self.vector_size, work
ers=self.workers)
        for epoch in range(self.epochs):
            #training the doc2vec
            model.train(utils.shuffle([x for x in tqdm(tagged_x)]), total_exam
ples=len(tagged_x), epochs=1)
            model.alpha -= self.learning_rate
            model.min_alpha = model.alpha
            self._model = model
            return self

    def transform(self, x):
        arr = np.array([self._model.infer_vector(preprocess_string(item))
                        for index, item in enumerate(x)])
        return arr
```

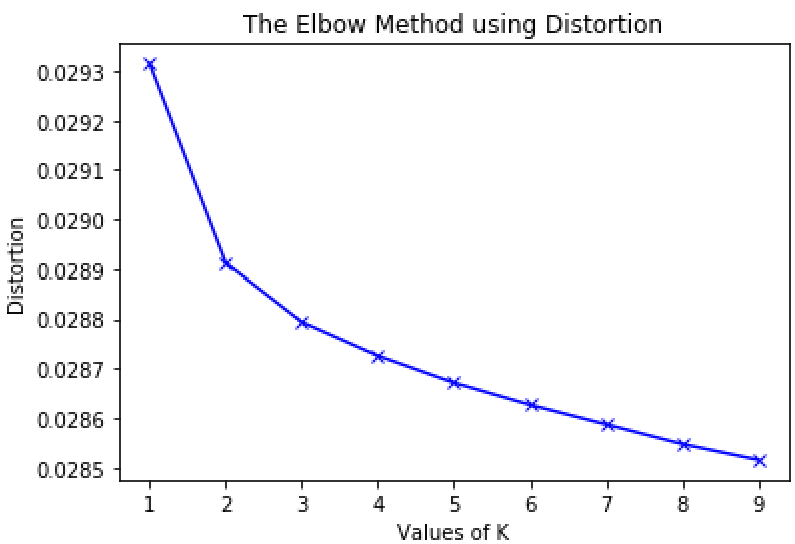
In [5]:
```
#Most of the  algorithm needs fare amount data preprocessing.
#All these preprocessing and the actual algorithm can be configured as separat
e reusable steps.
#Together all these steps connected in a single entity with an inlet and an ou
tlet is known as ‘Pipeline’.
#text classification using vector modelling is used
pl = Pipeline(steps=[('doc2vec', Doc2VecTransformer())])
#fitting the tweets into array as vectors
print("THE DATA IN VECTOR FORMAT")
vectors_df = pl.fit(atweets).transform(atweets)
vectors_df
```

```
THE DATA IN VECTOR FORMAT

100%|██████████| 7376/7376 [00:00<00:00, 634192.66it/s]
```

Out[5]:
```
array([[-0.0002226 , -0.00078273,  0.00349302, ...,  0.00509505,
        -0.00216749, -0.00329696],
       [ 0.00096381,  0.00316969, -0.00122428, ...,  0.00264962,
         0.00430936, -0.00124347],
       [ 0.00287613, -0.0046342 , -0.00358849, ...,  0.0040913 ,
         0.00217179, -0.00371479],
       ...,
       [-0.00054189,  0.00160454,  0.00258746, ...,  0.00365367,
        -0.00059056,  0.00434715],
       [-0.00312893,  0.00348994,  0.0046107 , ..., -0.00465256,
         0.00434811, -0.00391013],
       [ 0.00037742, -0.00093943, -0.00138079, ..., -0.00176827,
         0.00027079,  0.00200793]], dtype=float32)
```

In [5]:

```
#finding the optimal k for the data
#by the elbow curve method
#Pairwise distances between observations in n-dimensional space.
from scipy.spatial.distance import cdist
distortions = []
inertias = []
mapping1 = {}
mapping2 = {}
K = range(1,10)
X=vectors_df
for k in K:
    #Building and fitting the model
    kmeanModel = KMeans(n_clusters=k).fit(X)
    kmeanModel.fit(X)
    distortions.append(sum(np.min(cdist(X, kmeanModel.cluster_centers_,'euclid
ean'),axis=1)) / X.shape[0])
    inertias.append(kmeanModel.inertia_)
    mapping1[k] = sum(np.min(cdist(X, kmeanModel.cluster_centers_,'euclidean'
),axis=1)) / X.shape[0]
    mapping2[k] = kmeanModel.inertia_
plt.plot(K, distortions, 'bx-')
plt.xlabel('Values of K')
plt.ylabel('Distortion')
plt.title('The Elbow Method using Distortion')
plt.show()
```



In [6]:
```
from sklearn.decomposition import PCA
def analyze_tweets_pca(n_pca_components):
    doc2vectors = Pipeline(steps=[('doc2vec', Doc2VecTransformer())]).fit(atwe
ets).transform(atweets)
    #principal components identification
    #after identification we find the variance
    #by varieance we select the best
    #there by we get the n_components
    #here we select 10 components and find the varieance
    pca = PCA(n_components=n_pca_components)
    pca_vectors = pca.fit_transform(doc2vectors)
    print('All Principal Components ..')
    print(pca_vectors)
    for index, var in enumerate(pca.explained_variance_ratio_):
        print("Explained Variance ratio by Principal Component ",(index+1), "
 : ", var)
analyze_tweets_pca(15)
```

```
100%|██████████| 7376/7376 [00:00<00:00, 208310.18it/s]

All Principal Components ..
[[-2.9902367e-03  3.2389986e-03 -3.5948190e-03 ... -6.4300699e-04
  -7.4394550e-03 -1.9600685e-03]
 [ 3.3154848e-03 -3.3057990e-05  2.6365987e-04 ...  8.4918732e-04
  -8.0518946e-03  2.1157803e-03]
 [-4.8658410e-03  3.6684920e-03 -2.4421846e-03 ... -1.0604542e-03
   5.0981651e-04 -3.6131314e-03]
 ...
 [ 8.6531934e-04  2.6990778e-03 -2.8284886e-03 ... -2.1018307e-03
   4.9782405e-04 -1.8578261e-03]
 [ 3.7854447e-03 -3.4346136e-03  4.7816443e-03 ... -9.9620142e-04
   2.5772760e-03  2.9786017e-05]
 [-4.4071591e-03  4.9427464e-03  1.6082656e-04 ... -2.1221053e-03
   2.0324849e-03 -1.7234851e-03]]
Explained Variance ratio by Principal Component  1  :  0.045589283
Explained Variance ratio by Principal Component  2  :  0.01142543
Explained Variance ratio by Principal Component  3  :  0.011281333
Explained Variance ratio by Principal Component  4  :  0.011276222
Explained Variance ratio by Principal Component  5  :  0.011108515
Explained Variance ratio by Principal Component  6  :  0.011025072
Explained Variance ratio by Principal Component  7  :  0.010982002
Explained Variance ratio by Principal Component  8  :  0.01089648
Explained Variance ratio by Principal Component  9  :  0.010797306
Explained Variance ratio by Principal Component  10  :  0.010712784
Explained Variance ratio by Principal Component  11  :  0.010599911
Explained Variance ratio by Principal Component  12  :  0.010562724
Explained Variance ratio by Principal Component  13  :  0.010338022
Explained Variance ratio by Principal Component  14  :  0.010247799
Explained Variance ratio by Principal Component  15  :  0.010208069
```

In [7]:
```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score
#optimal kmeans clusterer
#here in the given range we have to find the optimal cluster
#from the above curve we can directly enter the optimal k
global optk;
optk=2;
class OptimalKMeansTextsClusterTransformer(BaseEstimator):

    def __init__(self, min_k, max_k):
        self.min_k = min_k
        self.max_k = max_k

    def fit(self, x, y=None):
        return self


    def transform(self, x):
        range_of_k = [x for x in range(self.min_k, self.max_k)]
        clusterer_pool = multiprocessing.Pool(processes=len(range_of_k))
        clusterer_process_responses = []
        for k in range_of_k:
            clusterer_process_responses.append(clusterer_pool.apply_async(self
._silhouette_score_with_k_, args=(x, k,)))

        optimal_k = optk
        clusterer_pool.close()
        print("Optimal k: ", optimal_k)
        optimal_clusterer = KMeansClusterer(num_means=optimal_k, distance=cosi
ne_distance, repeats=3)
        optimal_cluster_labels = optimal_clusterer.cluster(vectors=x, assign_c
lusters=True, trace=False)
        return x, optimal_cluster_labels

    def _silhouette_score_with_k_(self, vectors, k):
        clusterer = KMeansClusterer(num_means=k, distance=cosine_distance, rep
eats=3)
        cluster_labels = clusterer.cluster(vectors=vectors, assign_clusters=Tr
ue, trace=False)
        silhouette_score_k = silhouette_score(X=vectors, labels=cluster_labels
, metric='cosine')
        return k, silhouette_score_k
```

In [8]:
```
def plot_tweets_k_means_clusters_with_anomalies(pca_vectors, cluster_labels, p
ca_vectors_anomalies):
    pca_vectors_anomalies_x = []
    pca_vectors_anomalies_y = []

    for pca_vectors_elem in pca_vectors_anomalies:
        pca_vectors_anomalies_x.append(pca_vectors_elem[1])
        pca_vectors_anomalies_y.append(pca_vectors_elem[0])

    plt.title('Kmeans Cluster of Tweets')

    plt.scatter(x=pca_vectors[:, 1], y=pca_vectors[:, 0], c=cluster_labels)
    plt.scatter(x=pca_vectors_anomalies_x, y=pca_vectors_anomalies_y, marker=
'^')
    plt.show()

def plot_scatter_silhouette_scores(top_n_silhouette_scores, tweets_dict, silho
uette_score_per_tweet):
    plt.close('all')
    fig, (ax1, ax2) = plt.subplots(nrows=2, ncols=1, sharex=True)
    fig.suptitle('Silhouette Scores vs Tweets')
    sub_plot_scatter_silhouette_scores(ax=ax1, top_n_silhouette_scores=top_n_s
ilhouette_scores,
                                       tweets_dict=tweets_dict,
                                       silhouette_score_per_tweet=silhouette_s
core_per_tweet,
                                       with_annotation=False)

    sub_plot_scatter_silhouette_scores(ax=ax2, top_n_silhouette_scores=top_n_s
ilhouette_scores,
                                       tweets_dict=tweets_dict,
                                       silhouette_score_per_tweet=silhouette_s
core_per_tweet,
                                       with_annotation=True)
    plt.show()


def sub_plot_scatter_silhouette_scores(ax,top_n_silhouette_scores, tweets_dict
, silhouette_score_per_tweet, with_annotation):
    ax.set(xlabel='Tweet Index', ylabel='Silhouette Score')
    ax.scatter(*zip(*silhouette_score_per_tweet))
    ax.scatter(*zip(*top_n_silhouette_scores), edgecolors='red')

    if with_annotation:
        for (index, score) in top_n_silhouette_scores:
            ax.annotate(tweets_dict[index], xy=(index, score), xycoords='data'
)
```

In [9]:
```
#Value close to -1 :
#The data point is wrongly put in a cluster to which ideally it should not bel
ong to. Basically it is an ‘inlier’.
#Value close to 0:
#The data point should not belong to any cluster and should be separated out.
 It is an ‘outlier’ or ‘anomaly’.
#Value close to 1:
#The data point is perfectly placed in a right cluster.
from nltk.cluster.util import VectorSpaceClusterer
from nltk.cluster import KMeansClusterer,cosine_distance

def sort_key(t):
    return t[0]


def determine_anomaly_tweets_k_means(top_n):
    print("detecting anomaly tweets please wait.........")
    tweets_dict =atweets
    tweets = atweets
    pl = Pipeline(steps=[('doc2vec', Doc2VecTransformer()),('pca', PCA(n_compo
nents=5)),('kmeans', OptimalKMeansTextsClusterTransformer(min_k=2, max_k=5))])
    pl.fit(tweets)
    pca_vectors, cluster_labels = pl.transform(tweets)
    #sending the samples for processing
    silhouette_values = silhouette_samples(X=pca_vectors, labels=cluster_label
s, metric='cosine')
    tweet_index_silhouette_scores = []
    absolute_silhouette_scores_tweet_index = []

    for index, sh_score in enumerate(silhouette_values):
        #appending the scores
        absolute_silhouette_scores_tweet_index.append((abs(sh_score), index))
        tweet_index_silhouette_scores.append((index, sh_score))
#sorting of the scores
    sorted_scores = sorted(absolute_silhouette_scores_tweet_index, key=sort_ke
y)
#top anomaly scores into a array
    top_n_silhouette_scores = []
    pca_vectors_anomalies = []
    print("Top ", top_n, " anomalies")

    for i in range(top_n):
        abs_sh_score, index = sorted_scores[i]
        index_1, sh_score = tweet_index_silhouette_scores[index]
        top_n_silhouette_scores.append((index, sh_score))
        print(tweets_dict[index])
        print('PCA vector', pca_vectors[index])
        pca_vectors_anomalies.append(pca_vectors[index])
        print('Silhouette Score: ', sh_score)
        print("_____________________________________________________")
        print("        ")
    plot_tweets_k_means_clusters_with_anomalies(pca_vectors=pca_vectors, pca_v
ectors_anomalies=pca_vectors_anomalies,
                                                cluster_labels=cluster_labels)
    plot_scatter_silhouette_scores(top_n_silhouette_scores=top_n_silhouette_sc
ores,
```

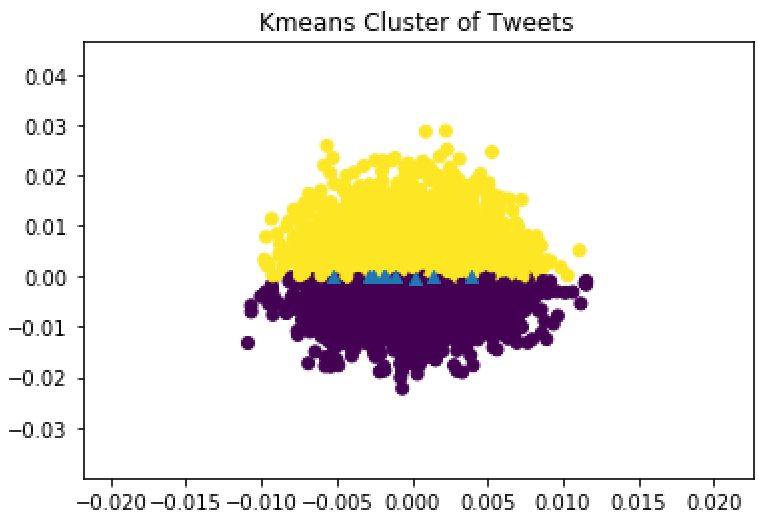
```
                                   tweets_dict=tweets_dict,
                                   silhouette_score_per_tweet=tweet_index_silh
ouette_scores)
determine_anomaly_tweets_k_means(10)
```

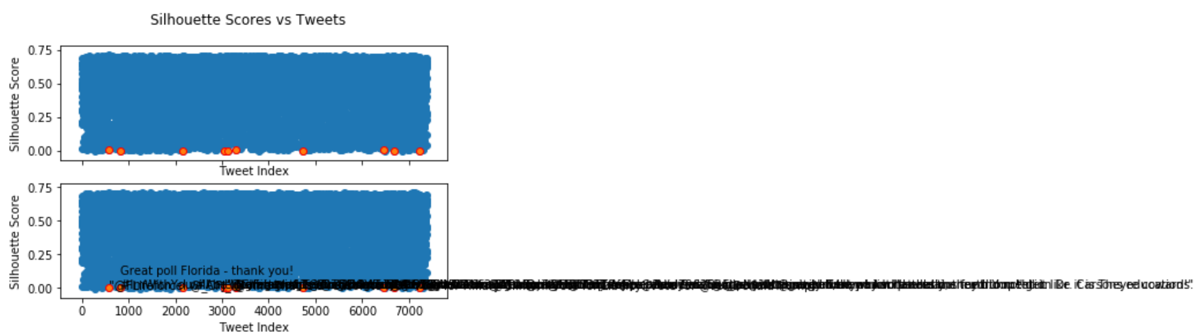
```
detecting anomaly tweets please wait.........

100%|██████████| 7376/7376 [00:00<00:00, 406197.06it/s]
```

```
Optimal k:  2
Top  10  anomalies
"@JWCarrr: @AnnCoulter @SenTenCes   Ben Carson has not criticized Trump! And
theres no need to mention Dr. Carsons education!"
PCA vector [-3.2053482e-05 -2.5690012e-03  5.8521293e-03 -4.3501430e-03
  5.3637696e-04]
Silhouette Score:  -0.000109924964
____________________________________________________

"@678b4612a62641f: @realDonaldTrump @Reid2962 @FoxNews @megynkelly my vote re
mains for trump!"
PCA vector [ 2.7334663e-05 -5.2391449e-03  3.0657016e-03  6.9453690e-04
 -1.3412528e-03]
Silhouette Score:  0.00013536551
____________________________________________________

"@The2ndguardsUS: @realDonaldTrump ROLL TIDE"
PCA vector [-9.4766474e-05  1.3967446e-04 -5.5229580e-03 -9.0200535e-04
  2.0221258e-03]
Silhouette Score:  0.0002770363
____________________________________________________

"@ChatteringTeef: @BornToBeGOP @realDonaldTrump @MittRomney hates to see Trum
ps success when he was so pathetic
PCA vector [ 1.9304702e-05  1.3890717e-03 -2.0462046e-03 -2.1152223e-04
 -2.4141071e-03]
Silhouette Score:  0.00043956868
____________________________________________________

Great poll Florida - thank you!
#ImWithYou #AmericaFirst https://t.co/6Odle7j1hd
PCA vector [ 6.6402572e-05 -2.5890544e-03  3.9905026e-03  1.0841555e-05
 -3.0809250e-03]
Silhouette Score:  0.00046262535
____________________________________________________

"@gregusp61: You really rocked them hard in S.C. Rubio and Cruz were pummled.
So glad Jeb is gone! Next no liar!"
PCA vector [-1.0180083e-06 -1.0827521e-03 -5.7604439e-03  1.7978915e-03
 -2.2253497e-03]
Silhouette Score:  0.00064649084
____________________________________________________

I will be interviewed on @TODAYshow and Good Morning America at 7:00 A.M.
PCA vector [-1.8472432e-05 -2.8075776e-03  2.5425404e-03 -2.4765243e-03
 -6.6297734e-04]
Silhouette Score:  -0.0008576783
____________________________________________________

"@FLifeforce: @_CFJ_ @vine That is a reason to NOT Vote for Hillary Clinton.
Vote for Liberty! Vote for @realDonaldTrump"
PCA vector [-4.6531452e-05 -1.9931660e-03  1.5120284e-03  1.7600132e-03
  5.2592247e-03]
Silhouette Score:  0.0009626799
____________________________________________________

Nothing conservative about the Club for Growth coming into my office and dema
```

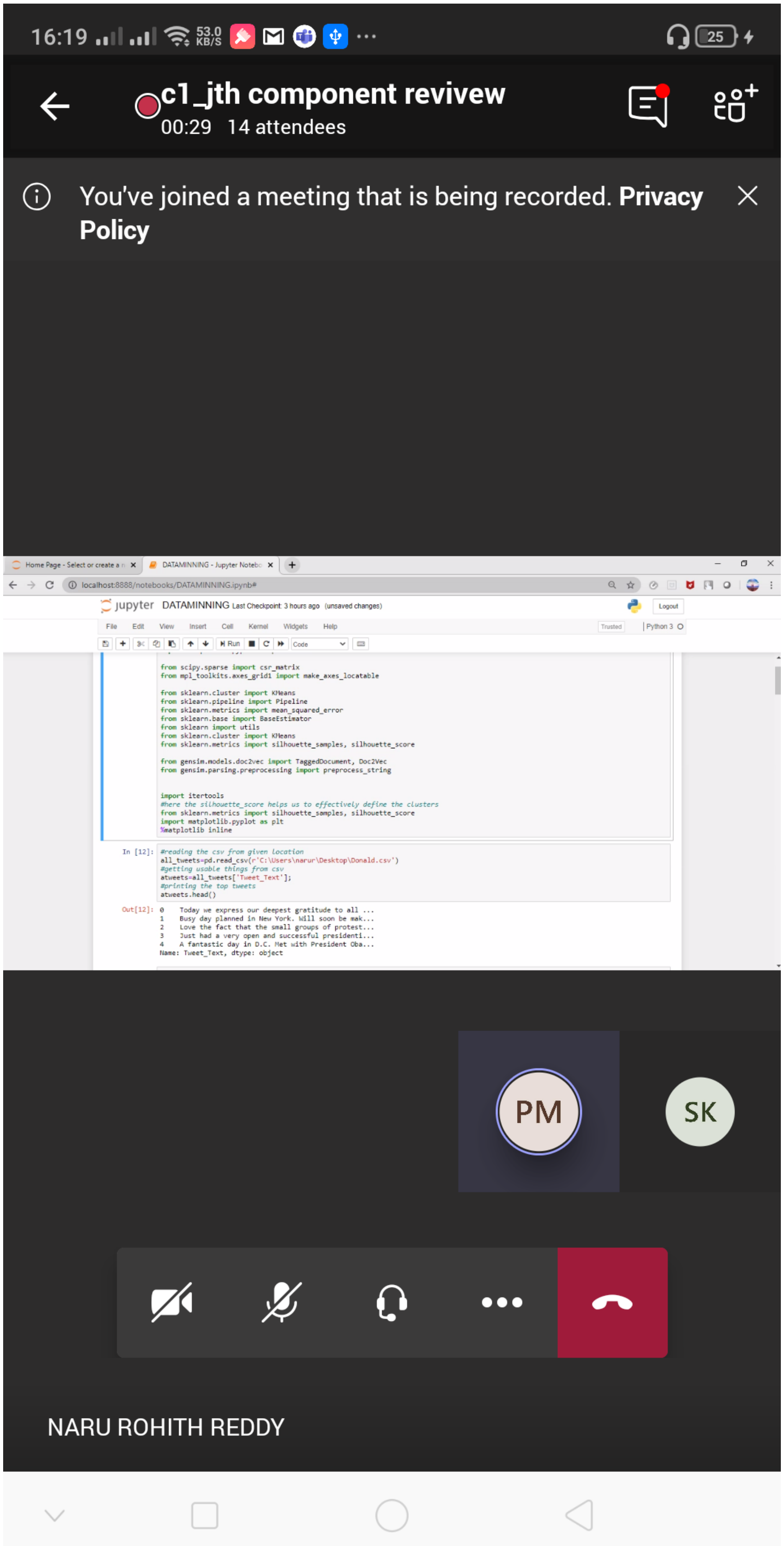
```
nding a $1M contribution, which naturally, they did not get.
PCA vector [ 2.7072740e-05  3.9043678e-03 -2.5129167e-03 -2.0975256e-03
 -3.9043322e-03]
Silhouette Score:  0.0010372016
```

```
"@AdriannaMarie: @realDonaldTrump People always take cheap shots at you,they
cant handle the truth.You tell it like it is.Theyre cowards.
PCA vector [ 9.0015528e-05 -1.8318299e-03  2.9857201e-03 -1.5183977e-03
 -6.5074274e-03]
Silhouette Score:  0.0012787855
```






```
In [ ]:
```



16:19
c1_jth component revivew
00:29 14 attendees
You've joined a meeting that is being recorded. Privacy Policy
DATAMINNING - Jupyter Notebo
localhost:8888/notebooks/DATAMINNING.ipynb#
jupyter DATAMINNING Last Checkpoint: 3 hours ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
from scipy.sparse import csr_matrix
from mpl_toolkits.axes_grid1 import make_axes_locatable
from sklearn.cluster import KMeans
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error
from sklearn.base import BaseEstimator
from sklearn import utils
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score
from gensim.models.doc2vec import TaggedDocument, Doc2Vec
from gensim.parsing.preprocessing import preprocess_string
import itertools
#here the silhouette_score helps us to effectively define the clusters
from sklearn.metrics import silhouette_samples, silhouette_score
import matplotlib.pyplot as plt
%matplotlib inline
In [12]: #reading the csv from given location
all_tweets=pd.read_csv(r'C:\Users\narur\Desktop\Donald.csv')
#getting usable things from csv
atweets=all_tweets['Tweet_Text'];
#printing the top tweets
atweets.head()
Out[12]: 0 Today we express our deepest gratitude to all ...
1 Busy day planned in New York. Will soon be mak...
2 Love the fact that the small groups of protest...
3 Just had a very open and successful presidenti...
4 A fantastic day in D.C. Met with President Oba...
Name: Tweet_Text, dtype: object
PM
SK
NARU ROHITH REDDY