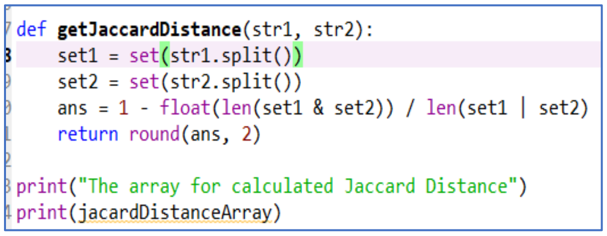
Q1) **Jaccard Distance – The Jaccard Distance tells us the dissimilarity between the two data sets.**

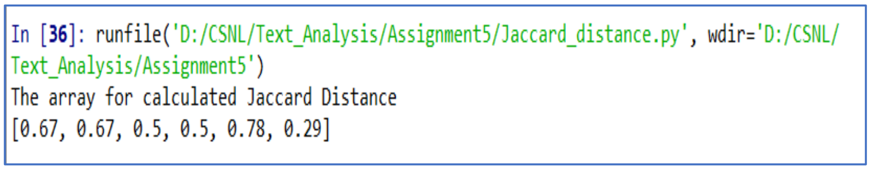
The Jaccard distance is calculated as (|AUB**|-| A∩B|)/|**AUB**|**

The higher the dissimilarity the greater is the distance between the two.
Example : If we look at the base and target 6 they are very Similar .example The calculated Jaccard Array in the figure 2 shows the distance between them as 0.29 which is less.
base = "Mummy teaches numbers and baby dances"
target = ["Mummy sings Rockabye and baby awakens", "Mummy works and baby eats apple",
"Mummy sings songs and baby dances", "Mummy teaches alphabets or baby dances",
"Mummy sings lullaby and Mummy sleeps", "Mummy teaches numbers or baby dances"]



```
def getJaccardDistance(str1, str2):
    set1 = set(str1.split())
    set2 = set(str2.split())
    ans = 1 - float(len(set1 & set2)) / len(set1 | set2)
    return round(ans, 2)

print("The array for calculated Jaccard Distance")
print(jacardDistanceArray)
```



```
In [36]: runfile('D:/CSNL/Text_Analysis/Assignment5/Jaccard_distance.py', wdir='D:/CSNL/
Text_Analysis/Assignment5')
The array for calculated Jaccard Distance
[0.67, 0.67, 0.5, 0.5, 0.78, 0.29]
```

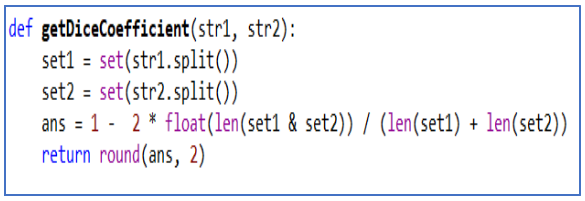
Figure 2.

Figure 1.

Triangle Inequality : The triangle inequality property states that **a+b>=c** i.e. The sum of the two sides is always greater than or equal to the third side.

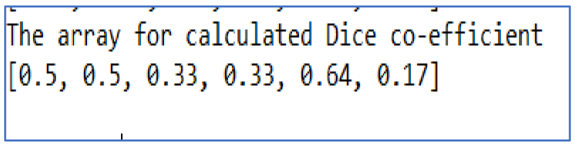
Now after summing up the 4rd and the 6th value (0.5+0.29) = 0.79, we can conclude that the Triangle Inequality Property holds true for Jaccard Distance because none of the values in the array is greater than the summed up value.

Q1 b) Dice Coefficient : Its calculated **as 2(| A∩B|)/|**A|+|B**|**
The Dice Co-efficient gives us the similarity between two datasets . But its not regarded as a proper Distance Metric because it doesn't hold true the triangle inequality property.
`



```
def getDiceCoefficient(str1, str2):
    set1 = set(str1.split())
    set2 = set(str2.split())
    ans = 1 -  2 * float(len(set1 & set2)) / (len(set1) + len(set2))
    return round(ans, 2)
```

For Example when I sum up the values at position 4th and 6th in the array (0.33 + 0.17) = 0.50 which is less than the 5th value i.e. 0.64 which clearly indicates of **Dice co-efficient not satisfying the triangle in equality property.**

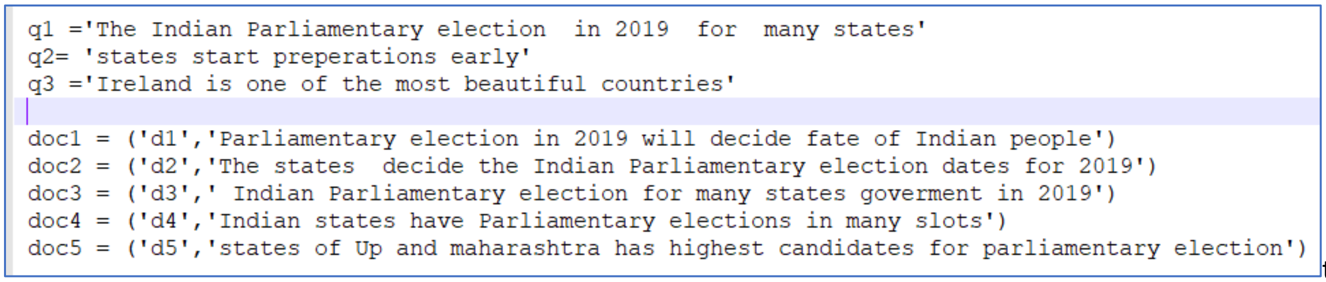


```
The array for calculated Dice co-efficient
[0.5, 0.5, 0.33, 0.33, 0.64, 0.17]
```

But at times it may satisfy the Triangle inequality property. For example if we add 1st and 2nd value = 1.0 which is greater than any other value in our array.

Figure : Calculated values for Dice Co-efficient.

Q2 a) Data set :

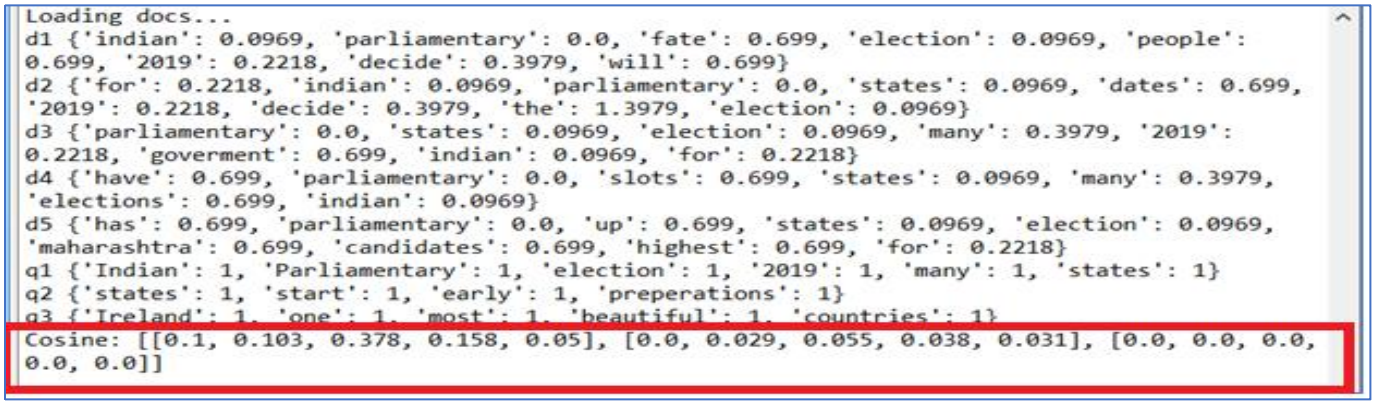


```
q1 ='The Indian Parliamentary election  in 2019  for  many states'
q2= 'states start preperations early'
q3 ='Ireland is one of the most beautiful countries'

doc1 = ('d1','Parliamentary election in 2019 will decide fate of Indian people')
doc2 = ('d2','The states  decide the Indian Parliamentary election dates for 2019')
doc3 = ('d3',' Indian Parliamentary election for many states goverment in 2019')
doc4 = ('d4','Indian states have Parliamentary elections in many slots')
doc5 = ('d5','states of Up and maharashtra has highest candidates for parliamentary election')
```

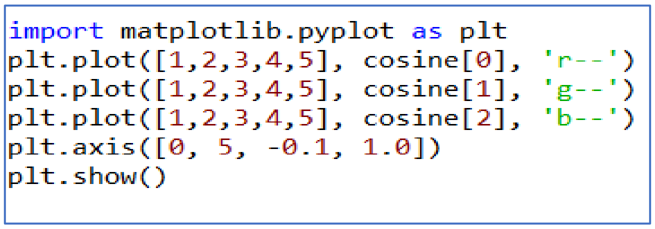
t

The docs from doc1 to doc5 are the variants of base q1.
The cosine similarity for all docs is calculated with respect to bases q1 , q2 and q3 and are stored in the cosine array highlighted below . Index 0 – q1, 1-q2, 2 -q3.
The Cosine Similarity states that if the cosine of angle is near one then quantities(here our two documents) are in the same direction i.e. similar.
So, It can be inferred that the cosine similarity score is highest for q1 and lowest for q3 because all the docs are variants of q1 and not at all related to q3.

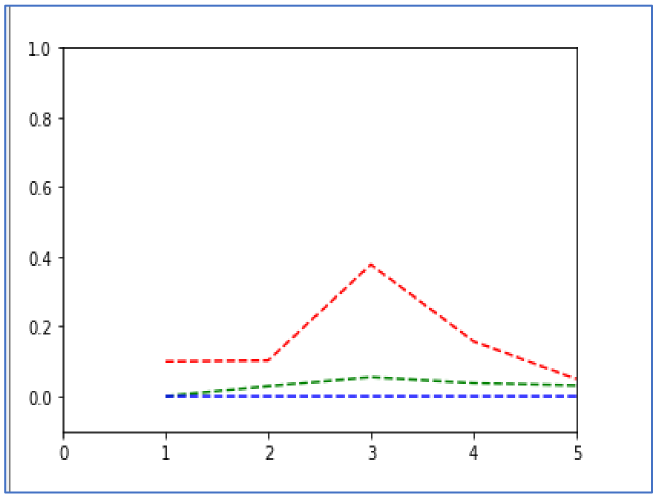


```
Loading docs...
d1 {'indian': 0.0969, 'parliamentary': 0.0, 'fate': 0.699, 'election': 0.0969, 'people':
0.699, '2019': 0.2218, 'decide': 0.3979, 'will': 0.699}
d2 {'for': 0.2218, 'indian': 0.0969, 'parliamentary': 0.0, 'states': 0.0969, 'dates': 0.699,
'2019': 0.2218, 'decide': 0.3979, 'the': 1.3979, 'election': 0.0969}
d3 {'parliamentary': 0.0, 'states': 0.0969, 'election': 0.0969, 'many': 0.3979, '2019':
0.2218, 'goverment': 0.699, 'indian': 0.0969, 'for': 0.2218}
d4 {'have': 0.699, 'parliamentary': 0.0, 'slots': 0.699, 'states': 0.0969, 'many': 0.3979,
'elections': 0.699, 'indian': 0.0969}
d5 {'has': 0.699, 'parliamentary': 0.0, 'up': 0.699, 'states': 0.0969, 'election': 0.0969,
'maharashtra': 0.699, 'candidates': 0.699, 'highest': 0.699, 'for': 0.2218}
q1 {'Indian': 1, 'Parliamentary': 1, 'election': 1, '2019': 1, 'many': 1, 'states': 1}
q2 {'states': 1, 'start': 1, 'early': 1, 'preperations': 1}
q3 {'Ireland': 1, 'one': 1, 'most': 1, 'beautiful': 1, 'countries': 1}
Cosine: [[0.1, 0.103, 0.378, 0.158, 0.05], [0.0, 0.029, 0.055, 0.038, 0.031], [0.0, 0.0, 0.0,
0.0, 0.0]]
```

Q2 b) Cosine -plot



```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4,5], cosine[0], 'r--')
plt.plot([1,2,3,4,5], cosine[1], 'g--')
plt.plot([1,2,3,4,5], cosine[2], 'b--')
plt.axis([0, 5, -0.1, 1.0])
plt.show()
```

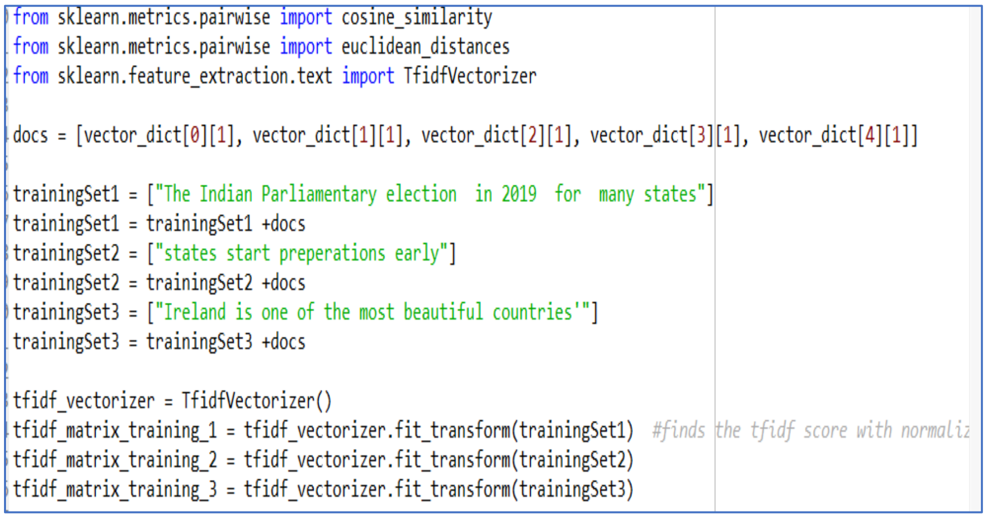


1)Now if we look at the graph below the Red dotted line indicates the plot of q1 vs all the five docs, It shows the highest score amongst all three lines since all the docs are related to q1.

2)The dotted green line shows a plot of q2 vs all five docs ,there are a few spikes for this line due to the inclusion of some similar words, but is very less as compared to q1.

3) The blue dotted line represents a plot of q3 vs all other docs , q3 is totally unrelated to all other docs so the plot shows a straight line which doesn't move up the zero scale along the y-axis

Q2 c) **Euclidian Distance** - The Euclidian distance is a metric used to find the distance between any two vectors which can be even of varying length.
Higher the distance more dis similarity between the vectors.
Package used :The sci-kit learn package is used to perform calculations for both cosine and Euclidian distance.

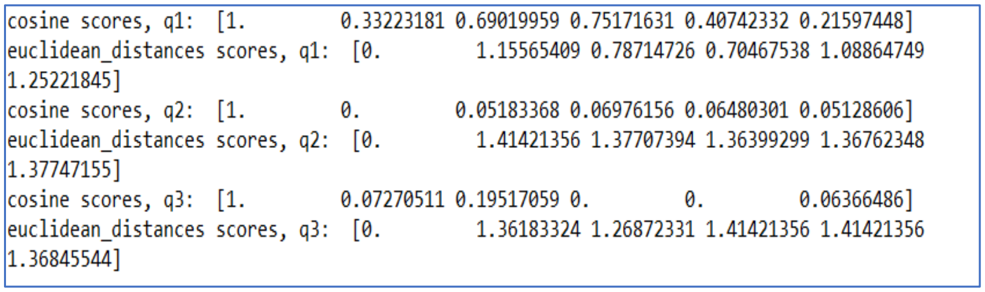


```
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics.pairwise import euclidean_distances
from sklearn.feature_extraction.text import TfidfVectorizer

docs = [vector_dict[0][1], vector_dict[1][1], vector_dict[2][1], vector_dict[3][1], vector_dict[4][1]]

trainingSet1 = ["The Indian Parliamentary election  in 2019  for  many states"]
trainingSet1 = trainingSet1 +docs
trainingSet2 = ["states start preperations early"]
trainingSet2 = trainingSet2 +docs
trainingSet3 = ["Ireland is one of the most beautiful countries'"]
trainingSet3 = trainingSet3 +docs

tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix_training_1 = tfidf_vectorizer.fit_transform(trainingSet1)  #finds the tfidf score with normaliz
tfidf_matrix_training_2 = tfidf_vectorizer.fit_transform(trainingSet2)
tfidf_matrix_training_3 = tfidf_vectorizer.fit_transform(trainingSet3)
```

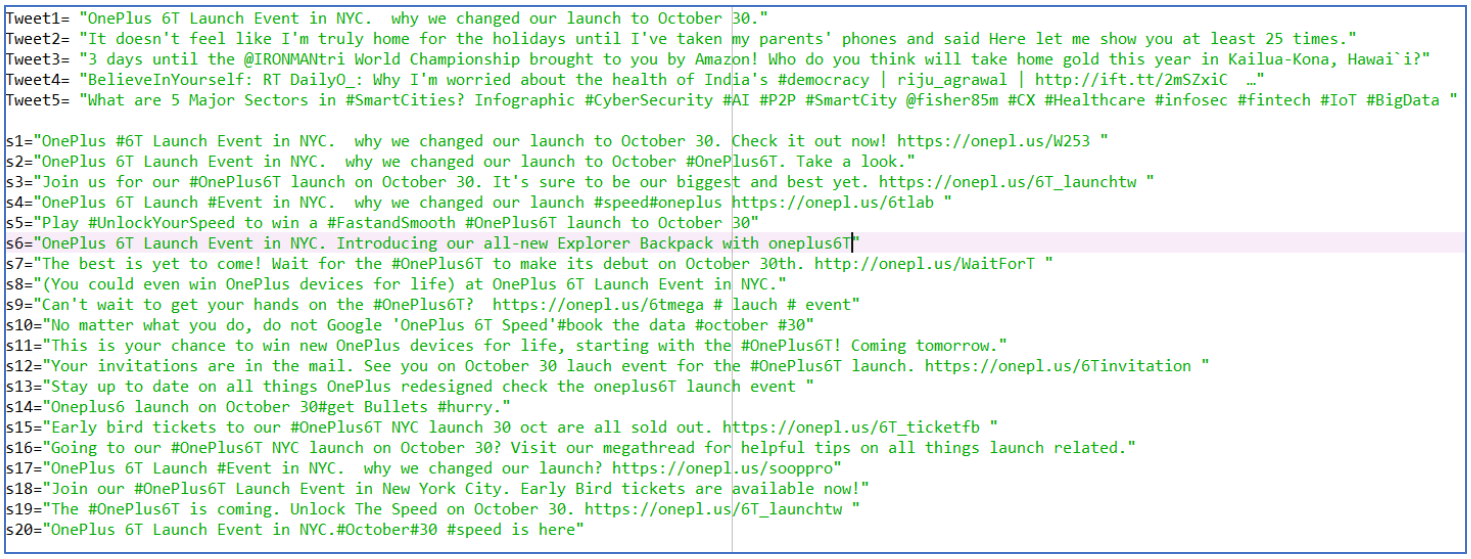


```
cosine scores, q1:  [1.         0.33223181 0.69019959 0.75171631 0.40742332 0.21597448]
euclidean_distances scores, q1:  [0.         1.15565409 0.78714726 0.70467538 1.08864749
1.25221845]
cosine scores, q2:  [1.         0.         0.05183368 0.06976156 0.06480301 0.05128606]
euclidean_distances scores, q2:  [0.         1.41421356 1.37707394 1.36399299 1.36762348
1.37747155]
cosine scores, q3:  [1.         0.07270511 0.19517059 0.         0.         0.06366486]
euclidean_distances scores, q3:  [0.         1.36183324 1.26872331 1.41421356 1.41421356
1.36845544]
```

The cosine similarity calculated in both the cases leads to the same conclusion that q1 has the highest similarity score followed by q2 and q3.
Whereas the Euclidian distance is less for more similar vectors i.e. q1 has small Euclidian distance due to similarity with all 5 docs , than q2 and then longest distance between q3 and all other 5 docs which are totally distinct.
So we can conclude that the Euclidian distance is inversely proportional to the cosine similarity score.
i.e. as the cosine similarity increases the Euclidian distance will tend to decrease.

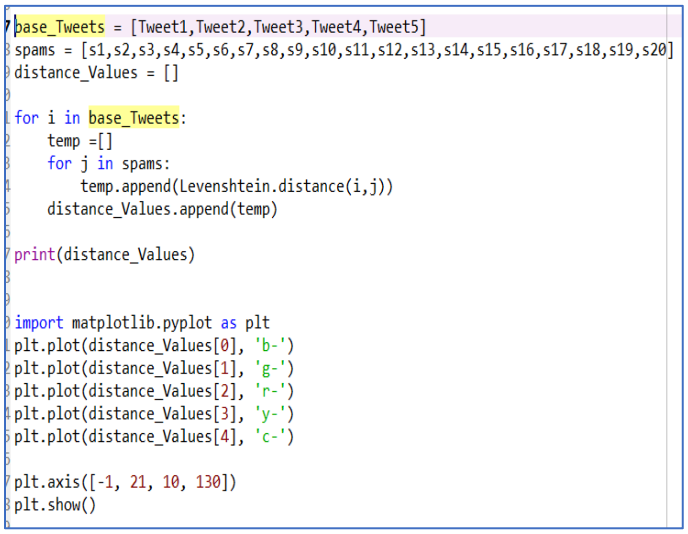
References : stack-overflow :eucledian-distance,
https://jakevdp.github.io/PythonDataScienceHandbook/04.01-simple-line-plots.html

Q3) Data set for 5 normal tweets Tweet 1 to Tweet 5 and Spam tweets from s1 to s10



```
Tweet1= "OnePlus 6T Launch Event in NYC.  why we changed our launch to October 30."
Tweet2= "It doesn't feel like I'm truly home for the holidays until I've taken my parents' phones and said Here let me show you at least 25 times."
Tweet3= "3 days until the @IRONMANtri World Championship brought to you by Amazon! Who do you think will take home gold this year in Kailua-Kona, Hawai`i?"
Tweet4= "BelieveInYourself: RT DailyO_: Why I'm worried about the health of India's #democracy | riju_agrawal | http://ift.tt/2mSZxiC  …"
Tweet5= "What are 5 Major Sectors in #SmartCities? Infographic #CyberSecurity #AI #P2P #SmartCity @fisher85m #CX #Healthcare #infosec #fintech #IoT #BigData "

s1="OnePlus #6T Launch Event in NYC.  why we changed our launch to October 30. Check it out now! https://onepl.us/W253 "
s2="OnePlus 6T Launch Event in NYC.  why we changed our launch to October #OnePlus6T. Take a look."
s3="Join us for our #OnePlus6T launch on October 30. It's sure to be our biggest and best yet. https://onepl.us/6T_launchtw "
s4="OnePlus 6T Launch #Event in NYC.  why we changed our launch #speed#oneplus https://onepl.us/6tlab "
s5="Play #UnlockYourSpeed to win a #FastandSmooth #OnePlus6T launch to October 30"
s6="OnePlus 6T Launch Event in NYC. Introducing our all-new Explorer Backpack with oneplus6T"
s7="The best is yet to come! Wait for the #OnePlus6T to make its debut on October 30th. http://onepl.us/WaitForT "
s8="(You could even win OnePlus devices for life) at OnePlus 6T Launch Event in NYC."
s9="Can't wait to get your hands on the #OnePlus6T?  https://onepl.us/6tmega # lauch # event"
s10="No matter what you do, do not Google 'OnePlus 6T Speed'#book the data #october #30"
s11="This is your chance to win new OnePlus devices for life, starting with the #OnePlus6T! Coming tomorrow."
s12="Your invitations are in the mail. See you on October 30 lauch event for the #OnePlus6T launch. https://onepl.us/6Tinvitation "
s13="Stay up to date on all things OnePlus redesigned check the oneplus6T launch event "
s14="Oneplus6 launch on October 30#get Bullets #hurry."
s15="Early bird tickets to our #OnePlus6T NYC launch 30 oct are all sold out. https://onepl.us/6T_ticketfb "
s16="Going to our #OnePlus6T NYC launch on October 30? Visit our megathread for helpful tips on all things launch related."
s17="OnePlus 6T Launch #Event in NYC.  why we changed our launch? https://onepl.us/sooppro"
s18="Join our #OnePlus6T Launch Event in New York City. Early Bird tickets are available now!"
s19="The #OnePlus6T is coming. Unlock The Speed on October 30. https://onepl.us/6T_launchtw "
s20="OnePlus 6T Launch Event in NYC.#October#30 #speed is here"
```



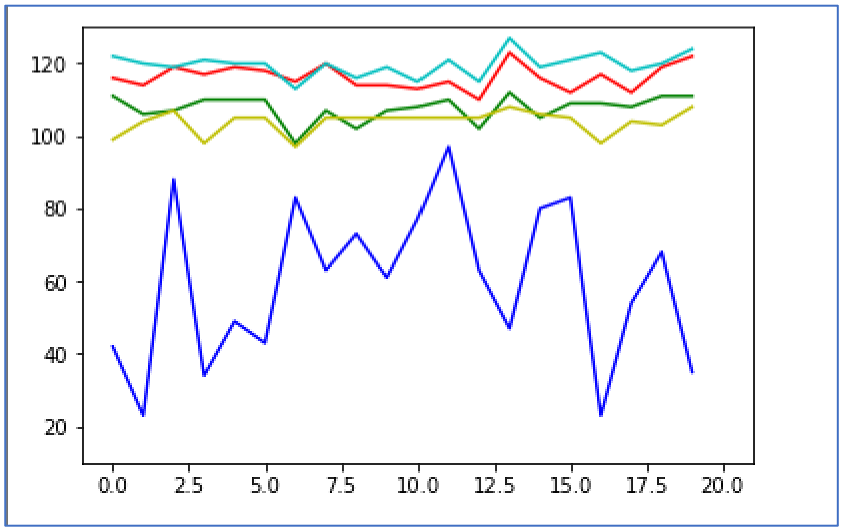
```
base_Tweets = [Tweet1,Tweet2,Tweet3,Tweet4,Tweet5]
spams = [s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15,s16,s17,s18,s19,s20]
distance_Values = []

for i in base_Tweets:
    temp =[]
    for j in spams:
        temp.append(Levenshtein.distance(i,j))
    distance_Values.append(temp)

print(distance_Values)


import matplotlib.pyplot as plt
plt.plot(distance_Values[0], 'b-')
plt.plot(distance_Values[1], 'g-')
plt.plot(distance_Values[2], 'r-')
plt.plot(distance_Values[3], 'y-')
plt.plot(distance_Values[4], 'c-')

plt.axis([-1, 21, 10, 130])
plt.show()
```



Code Snippet **Graph for spam vs Normal Tweets**

- The spam tweets were generated using urls, hashtags at various locations and adding some extra textual data.
- We can observe that the blue line shows the comparison between base Tweet1 and all other spam tweets.
- The other colored lines each represent the comparison against the spam tweets.
- I have used the levenshtein package in python to calculate the distance between the two data sets (documents/Tweets)

Inference :
The blue line shows **less levenshtein distance because all the 20 tweets are derived from the Tweet1** and all revolve around the topic oneplus6T.

.

The rest all lines show a much higher distance as they are not associated with the spam tweets in all aspects and have a distance from 100 to 120.

The blue line also shows spikes at times, which indicates that the spam tweets are not obvious, as Levenshtein distance is high for some spam tweets this may be because :

- While calculating levenshtein distance the length of the tweets is also taken into consideration, and in my documents also the tweets are of varying length which can be one of the reasons for the spikes observed in blue line and increase in the distance.
- Also there are spam tweets which have very less mention of the topic oneplus6T and lot involve urls which have no association with the topic which may be the other reason for increase in the levenshtein distance.