In [1]:

```
#load data
import pickle

with open("Integrated_prediction_hype1_test57.pkl", "rb") as fp:
    results= pickle.load(fp)

with open("y_test_57.pkl", "rb") as fp:
    y_test= pickle.load(fp)
```

In [2]:

```
results
```

Out[2]:

```
42308    [DÜNYA, DÜNYA, DÜNYA, DÜNYA, DÜNYA, DÜNYA, DÜN...
19743    [DÜNYA, DÜNYA, DÜNYA, DÜNYA, DÜNYA, DÜNYA, Tek...
26134    [SPOR, SPOR, SPOR, SPOR, SPOR, SPOR, SANAT, SA...
36274    [SPOR, SPOR, SPOR, SPOR, SPOR, SPOR, DÜNYA, DÜ...
39013    [Teknoloji, Teknoloji, Teknoloji, Teknoloji, T...
                               ...
19154    [DÜNYA, DÜNYA, DÜNYA, DÜNYA, DÜNYA, DÜNYA, DÜN...
19512    [SPOR, SPOR, Teknoloji, Teknoloji, Teknoloji, ...
23092    [SANAT, SANAT, SANAT, SANAT, SANAT, SANAT, SPO...
17335    [SANAT, SANAT, SANAT, SANAT, SANAT, SANAT, SAN...
22839    [DÜNYA, DÜNYA, DÜNYA, DÜNYA, DÜNYA, DÜNYA, Tek...
Name: Title, Length: 2325, dtype: object
```

In [3]:

```
def result_to_predict(result,j):
    p=[]
    for i in result:
        p.append(i[j])

    predict=pd.Series(p,index=result.index)
    return predict
```

In [4]:

```
import pandas as pd
result_to_predict(results,2)
```

Out[4]:

```
42308        DÜNYA
19743        DÜNYA
26134         SPOR
36274         SPOR
39013    Teknoloji
           ...
19154        DÜNYA
19512    Teknoloji
23092        SANAT
17335        SANAT
22839        DÜNYA
Length: 2325, dtype: object
```

In [5]:

```
from sklearn.metrics import confusion_matrix,  accuracy_score

def acc_sc_hype(test,result):
    n=len(result[result.index[0]])
    output=[]
    for i in range(0,n):
        output.append(round(accuracy_score(test,result_to_predict(result,i)),4))


    return output
```

In [6]:

```
acc_score_all=acc_sc_hype(y_test,results)
```

In [8]:

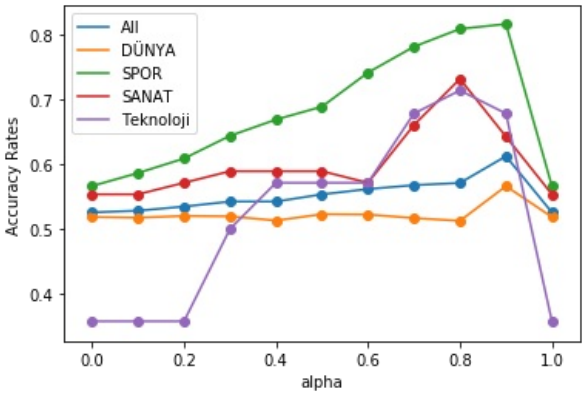
```
main_categories=["DÜNYA","SPOR","SANAT","Teknoloji"]

acc_all_with_cat=[acc_score_all]

for category in main_categories:
    cat_ind=y_test[y_test==category].index
    cat_test_y=y_test[cat_ind]
    cat_results=results[cat_ind]
    acc_all_with_cat.append(acc_sc_hype(cat_test_y,cat_results))



acc_all_with_cat
```

Out[8]:

```
[[0.5256,
  0.5282,
  0.5346,
  0.5424,
  0.5424,
  0.5535,
  0.5617,
  0.5677,
  0.5712,
  0.6125,
  0.5256],
 [0.5185,
  0.5174,
  0.5201,
  0.5195,
  0.513,
  0.5228,
  0.5223,
  0.5168,
  0.5125,
  0.5662,
  0.5185],
 [0.5664,
  0.5865,
  0.609,
  0.6441,
  0.6692,
  0.6892,
  0.7419,
  0.782,
  0.8095,
  0.817,
  0.5664],
 [0.5536,
  0.5536,
  0.5714,
  0.5893,
  0.5893,
  0.5893,
  0.5714,
  0.6607,
  0.7321,
  0.6429,
  0.5536],
 [0.3571,
  0.3571,
  0.3571,
  0.5,
  0.5714,
  0.5714,
  0.5714,
  0.6786,
  0.7143,
  0.6786,
  0.3571]]
```

In [9]:

```
import numpy as np
import matplotlib.pyplot as plt
x=[ round(i*0.1,1) for i in range(0,11) ]

plt.plot(x, acc_all_with_cat[0],label = "All")
plt.scatter(x, acc_all_with_cat[0])
plt.plot(x, acc_all_with_cat[1],label = "DÜNYA")
plt.scatter(x, acc_all_with_cat[1])

plt.plot(x, acc_all_with_cat[2],label = "SPOR")
plt.scatter(x, acc_all_with_cat[2])
plt.plot(x, acc_all_with_cat[3],label = "SANAT")
plt.scatter(x, acc_all_with_cat[3])

plt.plot(x, acc_all_with_cat[4],label ="Teknoloji")
plt.scatter(x, acc_all_with_cat[4])

plt.xlabel("alpha")
plt.ylabel("Accuracy Rates")

plt.legend()
plt.show()
```



In [10]:

```
alpha=x
beta=[round(1-i,1) for i in alpha]

All_list=acc_all_with_cat.copy()
All_list.insert(0,beta)
All_list.insert(0,alpha)


import plotly.graph_objects as go

fig = go.Figure(data=[go.Table(header=dict(values=['alpha', 'beta',"All","DÜNYA","SPOR","SANAT","Teknoloji"]),
                 cells=dict(values=All_list))
                     ])
fig.show()
#Accuracy Rates of Categories and all under alpha-beta value
```

| alpha | beta | All | DÜNYA | SPOR | SANAT | Teknoloji |
|---|---|---|---|---|---|---|
| 0 | 1 | 0.5256 | 0.5185 | 0.5664 | 0.5536 | 0.3571 |
| 0.1 | 0.9 | 0.5282 | 0.5174 | 0.5865 | 0.5536 | 0.3571 |
| 0.2 | 0.8 | 0.5346 | 0.5201 | 0.609 | 0.5714 | 0.3571 |
| 0.3 | 0.7 | 0.5424 | 0.5195 | 0.6441 | 0.5893 | 0.5 |
| 0.4 | 0.6 | 0.5424 | 0.513 | 0.6692 | 0.5893 | 0.5714 |
| 0.5 | 0.5 | 0.5535 | 0.5228 | 0.6892 | 0.5893 | 0.5714 |
| 0.6 | 0.4 | 0.5617 | 0.5223 | 0.7419 | 0.5714 | 0.5714 |
| 0.7 | 0.3 | 0.5677 | 0.5168 | 0.782 | 0.6607 | 0.6786 |
| 0.8 | 0.2 | 0.5712 | 0.5125 | 0.8095 | 0.7321 | 0.7143 |
| 0.9 | 0.1 | 0.6125 | 0.5662 | 0.817 | 0.6429 | 0.6786 |
| 1 | 0 | 0.5256 | 0.5185 | 0.5664 | 0.5536 | 0.3571 |

In [73]:

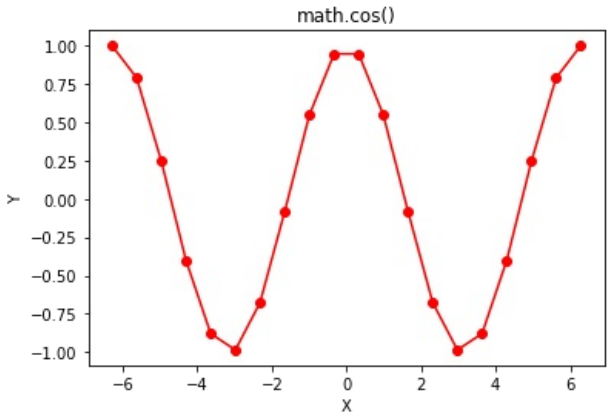
```
[i*pi/2 for i in alpha]
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-73-5c0f789685e2> in <module>
----> 1 [i*pi/2 for i in alpha]

<ipython-input-73-5c0f789685e2> in <listcomp>(.0)
----> 1 [i*pi/2 for i in alpha]

NameError: name 'pi' is not defined
```

In [72]:

```
import math
math.cos((3.14)/2)


%matplotlib inline
# Python program showing
# Graphical representation of
# cos() function
import math
import numpy as np
import matplotlib.pyplot as plt

in_array = np.linspace(-(2 * np.pi), 2 * np.pi, 20)

out_array = []

for i in range(len(in_array)):
    out_array.append(math.cos(in_array[i]))
    i += 1


print("in_array : ", in_array)
print("\nout_array : ", out_array)

# red for numpy.sin()
plt.plot(in_array, out_array, color = 'red', marker = "o")
plt.title("math.cos()")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```

```
in_array :  [-6.28318531 -5.62179738 -4.96040945 -4.29902153 -3.6376336  -2.97624567
 -2.31485774 -1.65346982 -0.99208189 -0.33069396  0.33069396  0.99208189
  1.65346982  2.31485774  2.97624567  3.6376336   4.29902153  4.96040945
  5.62179738  6.28318531]

out_array :  [1.0, 0.7891405093963934, 0.2454854871407988, -0.40169542465296987, -0.8794737512064891
, -0.9863613034027223, -0.6772815716257412, -0.08257934547233249, 0.5469481581224268, 0.945817241700
6346, 0.9458172417006346, 0.5469481581224268, -0.0825793454723316, -0.6772815716257405, -0.986361303
4027223, -0.8794737512064893, -0.40169542465296987, 0.2454854871407988, 0.7891405093963934, 1.0]
```




In [71]:

```
plt.show()
```

In [94]:

```
#get number of non-unique predictions
k=0
index_list=[]
for i in range(0,len(results1)):
    if len(list(set(results1[results1.index[i]])))!=1:
        index_list.append(results1.index[i])
        k=k+1
print(k)
print(len(results1))
```

```
728
2325
```

In [95]:

```
index_list
```

Out[95]:

Out[95]:

```
[19743,
 38035,
 37518,
 27047,
 38368,
 20350,
 34518,
 33757,
 15401,
 12886,
 30312,
 24771,
 19777,
 36548,
 32502,
 14035,
 13995,
 41778,
 32247,
 40712,
 35206,
 41199,
 22563,
 38094,
 15831,
 28468,
 33375,
 36369,
 40342,
 40092,
 31510,
 37884,
 23121,
 22542,
 27659,
 33080,
 29801,
 22945,
 33020,
 17161,
 37544,
 32001,
 27413,
 22018,
 35167,
 38166,
 25006,
 18177,
 33097,
 23210,
 40171,
 28306,
 37951,
 20442,
 32081,
 24850,
 38167,
 32108,
 15057,
 18055,
 29654,
 25257,
 37436,
 38724,
 21352,
 37743,
 35792,
 37714,
 31987,
 24665,
 33501,
 37978,
 35944,
 35506,
 38631,
 43477,
 18164,
 23049,
 14643,
 16308,
 19092,
 20780,
 32984,
```

34422,
21732,
14943,
31832,
31672,
33441,
20556,
27314,
15343,
29312,
43387,
29643,
23021,
40690,
31022,
42682,
21475,
23813,
26792,
35789,
42500,
43456,
25437,
20883,
34479,
13391,
25516,
41088,
20877,
31127,
22039,
26106,
24189,
38128,
30630,
40519,
20277,
36251,
29863,
32986,
40964,
42194,
28110,
37803,
37140,
33485,
37777,
15801,
29612,
33021,
16835,
28338,
15559,
37347,
16388,
22290,
37915,
25562,
33361,
18364,
31315,
18032,
23433,
18400,
31216,
34463,
23706,
35505,
26902,
14108,
24186,
42542,
24258,
22202,
40529,
19738,
25561,
28470,
16976,
23909,
23201,
39766,
41083,

22859,

14635,
38546,
43323,
15243,
36506,
39336,
42891,
40132,
30953,
40117,
21630,
13080,
14627,
20163,
22153,
32521,
27556,
23936,
22232,
36013,
42815,
30600,
25570,
34670,
37036,
17745,
26470,
23080,
13653,
17381,
21625,
15743,
23126,
32720,
20810,
32140,
35849,
13958,
13413,
36072,
29219,
31496,
43340,
34619,
21004,
33873,
32616,
21021,
16320,
26534,
28925,
18238,
29964,
25625,
32594,
23815,
23763,
39266,
23606,
37359,
18245,
32302,
42841,
22196,
43311,
27147,
12939,
38796,
18257,
25433,
16605,
22487,
28547,
15044,
42625,
16491,
43277,
15233,
38691,
35511,
13838,
35628,

38539,

29322,
31820,
13414,
25670,
17981,
22147,
15548,
38285,
34049,
39572,
27932,
30257,
37738,
32125,
37570,
41143,
31676,
37273,
34617,
33836,
19170,
36527,
26671,
35309,
29692,
40010,
29817,
40600,
36722,
33545,
25209,
13218,
40209,
31652,
19652,
24520,
24495,
30224,
38293,
24626,
23949,
15172,
13741,
43582,
28204,
24708,
18016,
33399,
29890,
42656,
23321,
26137,
19244,
39294,
34435,
23854,
20543,
14027,
35821,
21867,
25517,
37137,
33636,
42943,
38944,
25795,
32011,
42352,
23495,
19850,
17464,
23165,
29865,
31824,
23428,
14429,
25294,
43707,
16055,
41677,
15085,
14032,

15213,

40446,
18763,
37025,
37534,
28353,
15477,
37561,
27263,
30992,
26562,
36685,
16422,
29284,
28094,
33804,
19928,
35381,
27568,
32993,
41886,
21032,
20742,
38400,
21576,
39477,
31306,
43202,
22963,
43115,
33133,
34314,
21738,
33085,
38990,
18642,
20027,
29767,
23450,
38434,
32949,
25027,
42050,
32952,
23215,
13960,
19711,
42517,
30069,
17611,
16425,
27741,
25077,
42686,
40547,
35394,
15741,
19117,
36997,
35643,
17225,
32288,
41408,
24317,
39940,
25736,
32971,
36864,
20527,
13017,
31424,
26132,
33181,
34436,
42757,
34370,
38926,
17902,
40392,
24343,
19136,
23084,
15698,

18604,

32063,
43220,
16595,
42680,
32465,
36001,
34149,
35314,
22116,
33691,
41309,
18015,
38067,
35637,
43151,
30472,
26146,
43386,
35867,
32652,
13225,
30158,
21524,
27522,
21792,
25048,
29587,
20497,
31157,
36189,
37523,
43006,
23247,
33422,
31728,
36806,
23943,
25741,
15140,
21999,
28489,
28385,
14315,
34267,
28230,
29490,
18985,
35712,
38874,
35341,
28467,
27496,
13583,
20845,
16758,
40818,
26036,
43651,
32714,
14369,
33676,
33005,
20664,
22286,
21652,
26107,
34184,
33946,
27974,
38321,
24864,
17612,
25508,
15464,
41623,
38556,
19343,
16145,
25121,
39373,
32689,
37332,

25655,

19991,
32027,
25859,
33508,
14912,
34616,
28936,
19581,
13364,
34067,
37857,
23549,
15976,
20998,
37189,
35047,
42017,
24028,
20651,
28786,
19857,
32137,
18521,
12877,
36691,
36913,
24512,
18877,
20077,
15444,
33429,
21250,
29524,
37823,
24213,
20511,
43075,
38080,
14598,
40483,
14274,
37204,
31156,
28311,
33472,
27731,
19960,
36937,
13037,
15594,
37933,
36357,
37904,
36879,
20594,
18886,
18455,
18383,
43231,
14297,
41072,
23237,
23876,
13873,
22390,
35172,
13421,
18152,
36500,
37727,
34896,
13504,
28697,
41955,
35870,
26492,
14267,
33248,
21978,
30668,
31687,
42967,

15372,

19677,
27880,
13562,
26700,
28383,
33492,
24759,
39855,
15513,
29891,
29247,
42292,
25553,
13006,
26239,
25890,
23555,
23332,
34322,
33274,
32693,
13473,
40393,
25042,
36730,
40590,
14158,
14080,
32153,
38317,
22751,
31990,
21650,
43046,
31485,
36135,
31149,
33326,
13183,
35852,
33772,
32052,
15898,
23671,
37538,
24391,
34546,
18860,
20695,
39353,
18879,
22265,
17665,
23457,
40507,
24141,
13866,
28755,
20101,
23314,
40765,
18446,
35039,
20570,
22980,
18192,
20761,
33073,
29402,
38019,
20151,
37531,
21334,
21341,
35747,
23037,
19256,
27331,
20632,
26794,
23252,
31325,

20841,

19798,
25635,
19760,
36320,
13784,
39258,
27826,
18370,
21817,
43540,
30295,
36469,
33624,
25381,
32786,
26554,
36589,
42747,
14844,
42817,
14443,
34531,
19807,
27806,
35073,
18832,
26897,
13902,
37508,
38002,
25205,
22248,
23148,
36029,
28831,
43718,
21413,
13336,
15792,
17320,
36166,
32507,
26589,
29481,
38126,
39351,
32980,
18934,
14273,
18403,
24254,
16049,
13761,
13479,
20951,
29721,
15112,
21022,
24263,
39970,
19512,
23092,
17335]

In [97]:

```
n=18177

print(results1[n])

for j in range(0,len(predict_list1)):
    print(predict_list1[j][n])
```

```
['DÜNYA', 'DÜNYA', 'DÜNYA', 'DÜNYA', 'Teknoloji', 'Teknoloji', 'Teknoloji', 'Teknoloji', 'DÜNYA', 'D
ÜNYA', 'DÜNYA', 'DÜNYA']
DÜNYA
DÜNYA
DÜNYA
DÜNYA
DÜNYA
DÜNYA
DÜNYA
DÜNYA
DÜNYA
DÜNYA
DÜNYA
DÜNYA
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: