ASSIGNMENT-03

Q1.

Time taken to make a dictionary is :

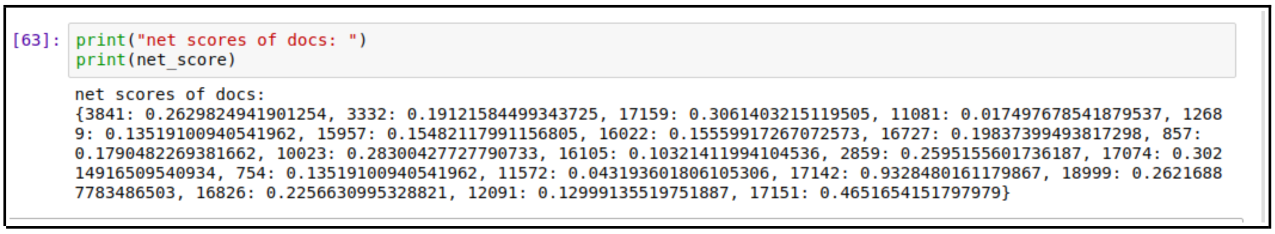
<u>Heuristics used for choosing 'r':</u>

- Setting threshold via mean tf for every term and choosing r on the basis of basis of mean tf value and further dividing it into high and low.

  Drawbacks: Since rarest term document list for and high and low will be small whereas frequent term list will be large for high low. So, if a query consists of one rare term and one frequent term. Then while fetching top k documents chances of rarest term document to appeared in that list is very less.
  So, this approach is not used.

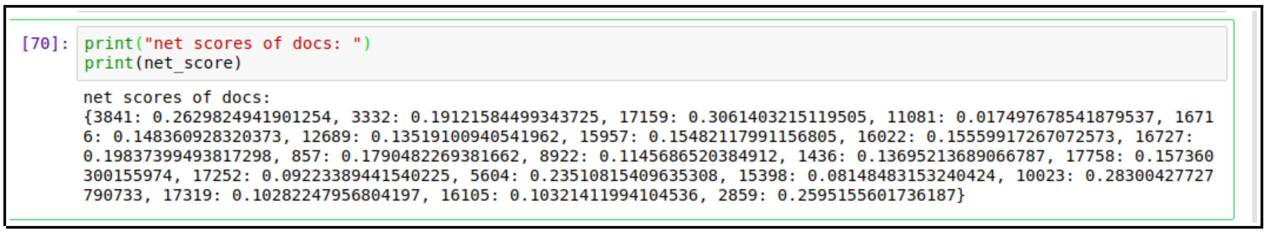
- *Randomly assign r.*

  Query: 'illustrious justice department'



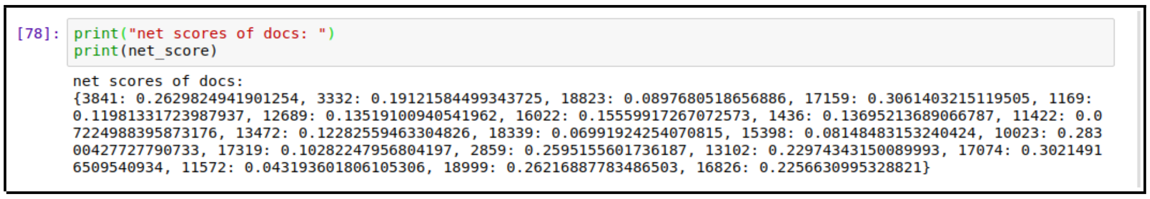
```
[63]: print("net scores of docs: ")
      print(net_score)

      net scores of docs:
      {3841: 0.2629824941901254, 3332: 0.19121584499343725, 17159: 0.3061403215119505, 11081: 0.017497678541879537, 1268
      9: 0.13519100940541962, 15957: 0.15482117991156805, 16022: 0.15559917267072573, 16727: 0.19837399493817298, 857:
      0.1790482269381662, 10023: 0.28300427727790733, 16105: 0.10321411994104536, 2859: 0.2595155601736187, 17074: 0.302
      14916509540934, 754: 0.13519100940541962, 11572: 0.043193601806105306, 17142: 0.9328480161179867, 18999: 0.2621688
      7783486503, 16826: 0.2256630995328821, 12091: 0.12999135519751887, 17151: 0.4651654151797979}
```

- *Choosing r using idf value.*



```
[70]: print("net scores of docs: ")
      print(net_score)

      net scores of docs:
      {3841: 0.2629824941901254, 3332: 0.19121584499343725, 17159: 0.3061403215119505, 11081: 0.017497678541879537, 1671
      6: 0.148360928320373, 12689: 0.13519100940541962, 15957: 0.15482117991156805, 16022: 0.15559917267072573, 16727:
      0.19837399493817298, 857: 0.1790482269381662, 8922: 0.1145686520384912, 1436: 0.13695213689066787, 17758: 0.157360
      300155974, 17252: 0.09223389441540225, 5604: 0.23510815409635308, 15398: 0.08148483153240424, 10023: 0.28300427727
      790733, 17319: 0.10282247956804197, 16105: 0.10321411994104536, 2859: 0.2595155601736187}
```

- *'r' = 30+(int)(((maxDF)-len(postings))/(maxDF))*10*



```
[78]: print("net scores of docs: ")
      print(net_score)

      net scores of docs:
      {3841: 0.2629824941901254, 3332: 0.19121584499343725, 18823: 0.0897680518656886, 17159: 0.3061403215119505, 1169:
      0.11981331723987937, 12689: 0.13519100940541962, 16022: 0.15559917267072573, 1436: 0.13695213689066787, 11422: 0.0
      7224988395873176, 13472: 0.12282559463304826, 18339: 0.06991924254070815, 15398: 0.08148483153240424, 10023: 0.283
      00427727790733, 17319: 0.10282247956804197, 2859: 0.2595155601736187, 13102: 0.22974343150089993, 17074: 0.3021491
      6509540934, 11572: 0.043193601806105306, 18999: 0.26216887783486503, 16826: 0.2256630995328821}
```

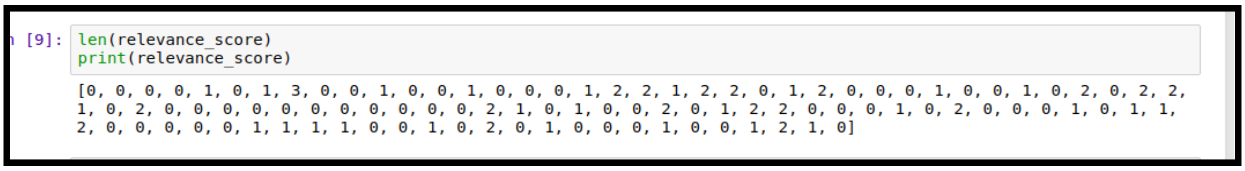
Q2.

1. Make a file having URLs in order of max DCG. State how many such files could be made.

   <u>For qid:4</u> :

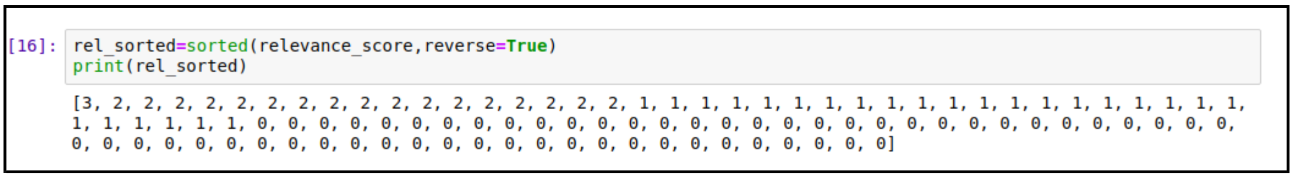
   Total number of documents with qid:4 are 103.

- Relevance score list



```
[9]: len(relevance_score)
     print(relevance_score)

     [0, 0, 0, 0, 1, 0, 1, 3, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 2, 2, 1, 2, 2, 0, 1, 2, 0, 0, 0, 1, 0, 0, 1, 0, 2, 0, 2, 2,
     1, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 0, 1, 0, 0, 2, 0, 1, 2, 2, 0, 0, 0, 1, 0, 2, 0, 0, 0, 1, 0, 1, 1,
     2, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 2, 0, 1, 0, 0, 0, 1, 0, 0, 1, 2, 1, 0]
```

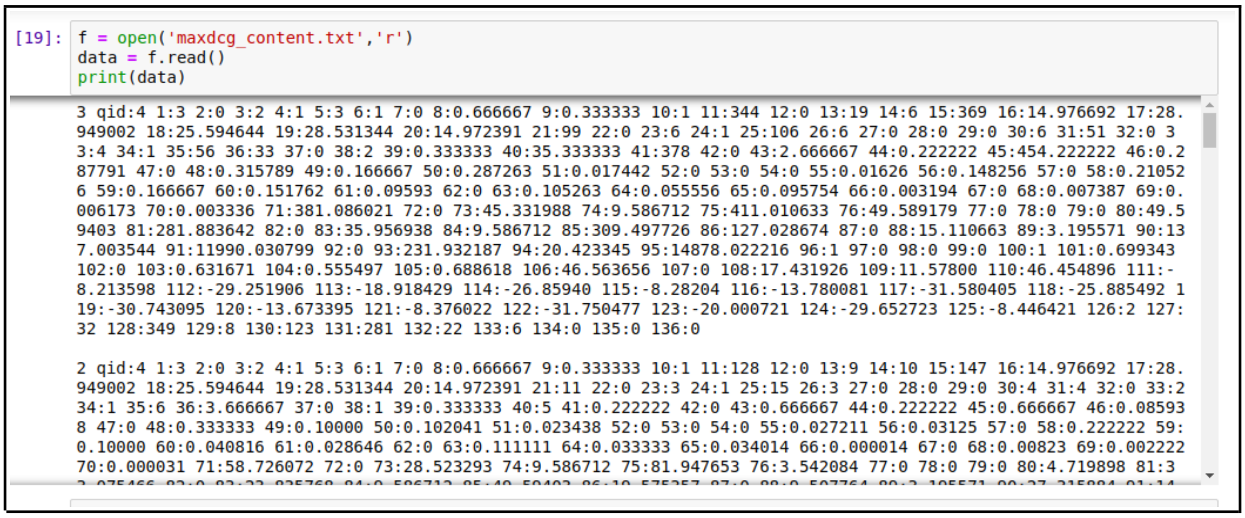
- Sorted Relevance score



```
[16]: rel_sorted=sorted(relevance_score,reverse=True)
      print(rel_sorted)

      [3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
      1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

- File containing URL in order of max DCG .
  [maxdcg_content.txt]

- Data in file.

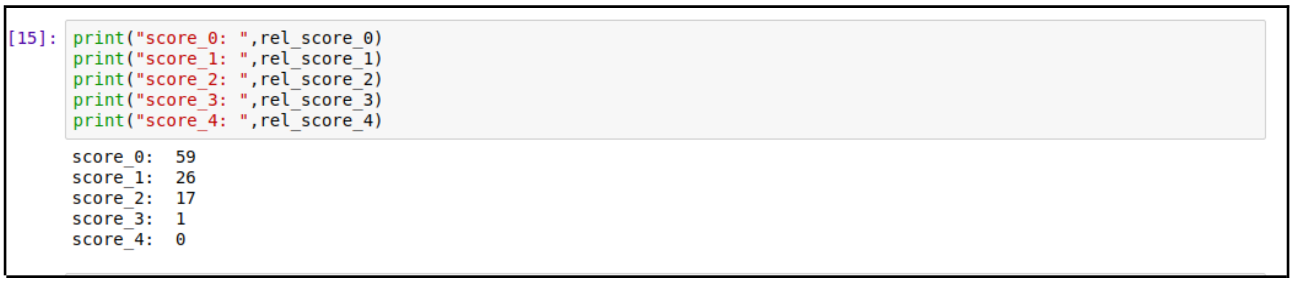


```
[19]: f = open('maxdcg_content.txt','r')
      data = f.read()
      print(data)

      3 qid:4 1:3 2:0 3:2 4:1 5:3 6:1 7:0 8:0.666667 9:0.333333 10:1 11:344 12:0 13:19 14:6 15:369 16:14.976692 17:28.
      949002 18:25.594644 19:28.531344 20:14.972391 21:99 22:0 23:6 24:1 25:106 26:6 27:0 28:0 29:0 30:6 31:51 32:0 3
      3:4 34:1 35:56 36:33 37:0 38:2 39:0.333333 40:35.333333 41:378 42:0 43:2.666667 44:0.222222 45:454.222222 46:0.2
      87791 47:0 48:0.315789 49:0.166667 50:0.287263 51:0.017442 52:0 53:0 54:0 55:0.01626 56:0.148256 57:0 58:0.21052
      6 59:0.166667 60:0.151762 61:0.09593 62:0 63:0.105263 64:0.055556 65:0.095754 66:0.003194 67:0 68:0.007387 69:0.
      006173 70:0.003336 71:381.086021 72:0 73:45.331988 74:9.586712 75:411.010633 76:49.589179 77:0 78:0 79:0 80:49.5
      9403 81:281.883642 82:0 83:35.956938 84:9.586712 85:309.497726 86:127.028674 87:0 88:15.110663 89:3.195571 90:13
      7.003544 91:11990.030799 92:0 93:231.932187 94:20.423345 95:14878.022216 96:1 97:0 98:0 99:0 100:1 101:0.699343
      102:0 103:0.631671 104:0.555497 105:0.688618 106:46.563656 107:0 108:17.431926 109:11.57800 110:46.454896 111:-
      8.213598 112:-29.251906 113:-18.918429 114:-26.85940 115:-8.28204 116:-13.780081 117:-31.580405 118:-25.885492 1
      19:-30.743095 120:-13.673395 121:-8.376022 122:-31.750477 123:-20.000721 124:-29.652723 125:-8.446421 126:2 127:
      32 128:349 129:8 130:123 131:281 132:22 133:6 134:0 135:0 136:0

      2 qid:4 1:3 2:0 3:2 4:1 5:3 6:1 7:0 8:0.666667 9:0.333333 10:1 11:128 12:0 13:9 14:10 15:147 16:14.976692 17:28.
      949002 18:25.594644 19:28.531344 20:14.972391 21:11 22:0 23:3 24:1 25:15 26:3 27:0 28:0 29:0 30:4 31:4 32:0 33:2
      34:1 35:6 36:3.666667 37:0 38:1 39:0.333333 40:5 41:0.222222 42:0 43:0.666667 44:0.222222 45:0.666667 46:0.08593
      8 47:0 48:0.333333 49:0.10000 50:0.102041 51:0.023438 52:0 53:0 54:0 55:0.027211 56:0.03125 57:0 58:0.222222 59:
      0.10000 60:0.040816 61:0.028646 62:0 63:0.111111 64:0.033333 65:0.034014 66:0.000014 67:0 68:0.00823 69:0.002222
      70:0.000031 71:58.726072 72:0 73:28.523293 74:9.586712 75:81.947653 76:3.542084 77:0 78:0 79:0 80:4.719898 81:3
```

- Count of each relevance score

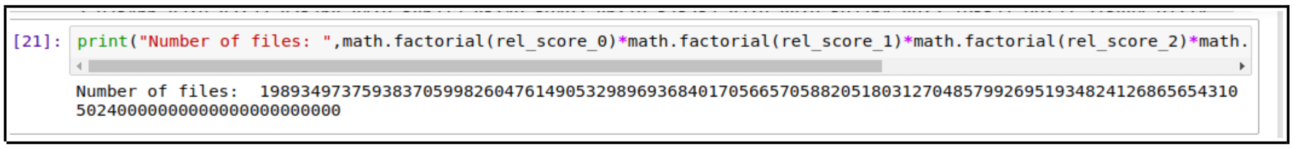
  Number documnets having relevance score as ‘0’: 59
  Number documnets having relevance score as ‘1’: 26

Number documnets having relevance score as ‘2’: 17
Number documnets having relevance score as ‘3’: 1
Number documnets having relevance score as ‘4’: 0

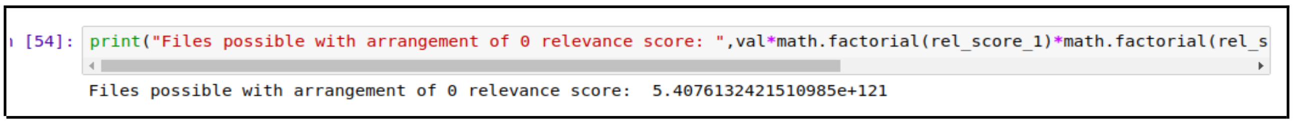


```
[15]: print("score_0: ",rel_score_0)
      print("score_1: ",rel_score_1)
      print("score_2: ",rel_score_2)
      print("score_3: ",rel_score_3)
      print("score_4: ",rel_score_4)

      score_0:  59
      score_1:  26
      score_2:  17
      score_3:  1
      score_4:  0
```

- Number of files that can be made with max DCG.



```
[21]: print("Number of files: ",math.factorial(rel_score_0)*math.factorial(rel_score_1)*math.factorial(rel_score_2)*math.

      Number of files:  198934973759383705998260476149053298969368401705665705882051803127048579926951934824126865654310
      50240000000000000000000000
```



```
[54]: print("Files possible with arrangement of 0 relevance score: ",val*math.factorial(rel_score_1)*math.factorial(rel_s

      Files possible with arrangement of 0 relevance score:  5.4076132421510985e+121
```

2. Compute nDCG.

(1) At 50

- DCG : 10.379472755702539
- IDCG : 27.998739896756508
- nDCG: 0.37071213897397365

(2) Whole Dataset

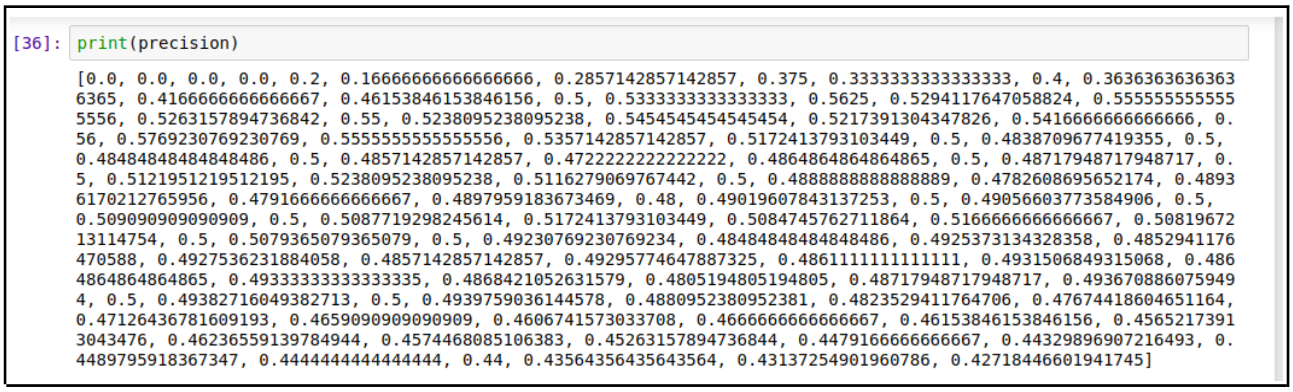
- DCG: 17.799227590651103
- IDCG: 27.998739896756508
- nDCG: 0.6357153091990773

3. Plot a Precision-Recall curve for query “qid:4”.

- <u>Normal curve of data:</u>

  Precision and Recall outputs on the data

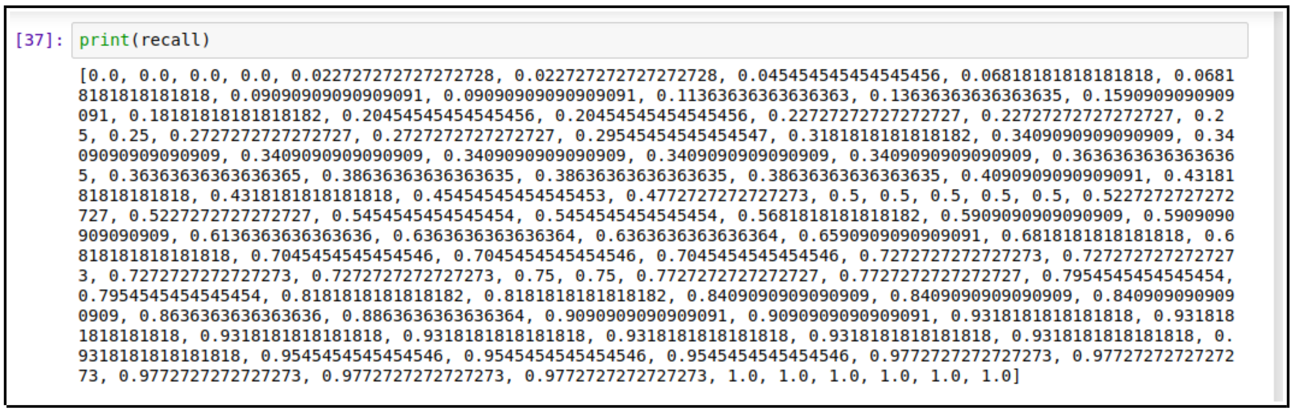
- Precision:



```
[36]: print(precision)
```

```
[0.0, 0.0, 0.0, 0.0, 0.2, 0.16666666666666666, 0.2857142857142857, 0.375, 0.3333333333333333, 0.4, 0.3636363636363
6365, 0.4166666666666667, 0.46153846153846156, 0.5, 0.5333333333333333, 0.5625, 0.5294117647058824, 0.555555555555
5556, 0.5263157894736842, 0.55, 0.5238095238095238, 0.5454545454545454, 0.5217391304347826, 0.5416666666666666, 0.
56, 0.5769230769230769, 0.5555555555555556, 0.5357142857142857, 0.5172413793103449, 0.5, 0.4838709677419355, 0.5,
0.48484848484848486, 0.5, 0.4857142857142857, 0.4722222222222222, 0.4864864864864865, 0.5, 0.48717948717948717, 0.
5, 0.5121951219512195, 0.5238095238095238, 0.5116279069767442, 0.5, 0.4888888888888889, 0.4782608695652174, 0.4893
617021276596, 0.4791666666666667, 0.4897959183673469, 0.48, 0.49019607843137253, 0.5, 0.49056603773584906, 0.5,
0.509090909090909, 0.5, 0.5087719298245614, 0.5172413793103449, 0.5084745762711864, 0.5166666666666667, 0.50819672
13114754, 0.5, 0.5079365079365079, 0.5, 0.49230769230769234, 0.48484848484848486, 0.4925373134328358, 0.4852941176
470588, 0.4927536231884058, 0.4857142857142857, 0.49295774647887325, 0.4861111111111111, 0.4931506849315068, 0.486
4864864864865, 0.49333333333333335, 0.4868421052631579, 0.4805194805194805, 0.48717948717948717, 0.493670886075949
4, 0.5, 0.49382716049382713, 0.5, 0.4939759036144578, 0.4880952380952381, 0.4823529411764706, 0.47674418604651164,
0.47126436781609193, 0.4659090909090909, 0.4606741573033708, 0.4666666666666667, 0.46153846153846156, 0.4565217391
3043476, 0.46236559139784944, 0.4574468085106383, 0.45263157894736844, 0.4479166666666667, 0.44329896907216493, 0.
4489795918367347, 0.4444444444444444, 0.44, 0.43564356435643564, 0.43137254901960786, 0.42718446601941745]
```

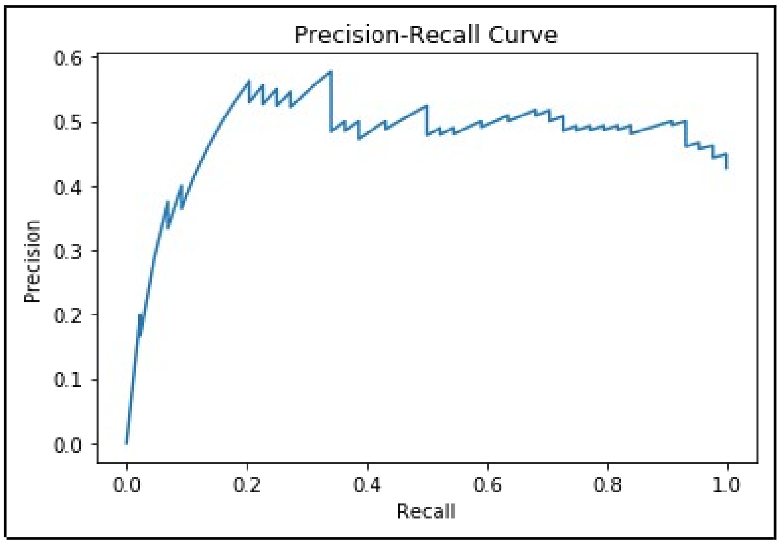
- Recall:



```
[37]: print(recall)
```

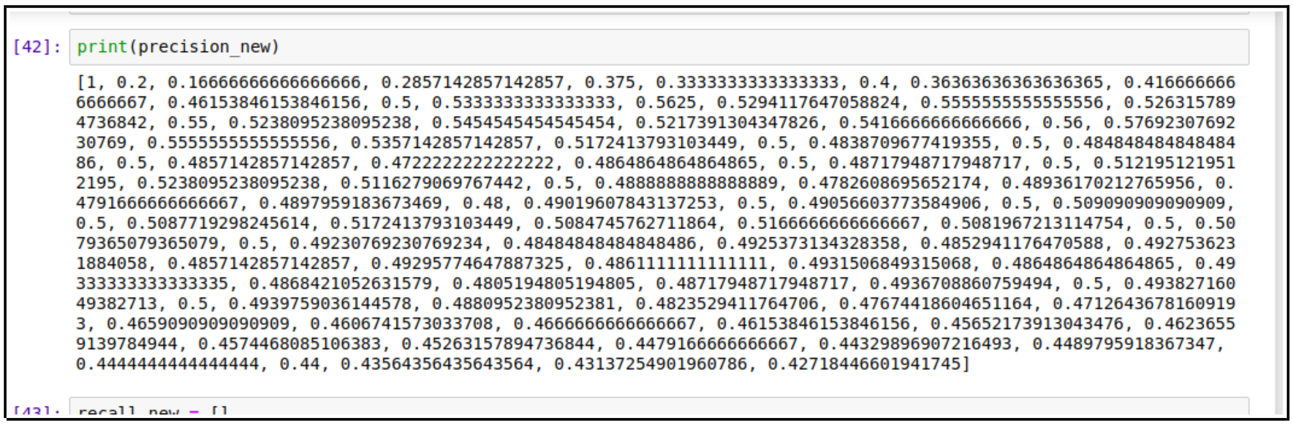
```
[0.0, 0.0, 0.0, 0.0, 0.022727272727272728, 0.022727272727272728, 0.045454545454545456, 0.06818181818181818, 0.0681
8181818181818, 0.09090909090909091, 0.09090909090909091, 0.11363636363636363, 0.13636363636363635, 0.1590909090909
091, 0.18181818181818182, 0.20454545454545456, 0.20454545454545456, 0.22727272727272727, 0.22727272727272727, 0.2
5, 0.25, 0.2727272727272727, 0.2727272727272727, 0.29545454545454547, 0.3181818181818182, 0.3409090909090909, 0.34
09090909090909, 0.3409090909090909, 0.3409090909090909, 0.3409090909090909, 0.3409090909090909, 0.3636363636363636
5, 0.36363636363636365, 0.38636363636363635, 0.38636363636363635, 0.38636363636363635, 0.4090909090909091, 0.43181
81818181818, 0.4318181818181818, 0.45454545454545453, 0.4772727272727273, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5227272727272
727, 0.5227272727272727, 0.5454545454545454, 0.5454545454545454, 0.5681818181818182, 0.5909090909090909, 0.5909090
909090909, 0.6136363636363636, 0.6363636363636364, 0.6363636363636364, 0.6590909090909091, 0.6818181818181818, 0.6
818181818181818, 0.7045454545454546, 0.7045454545454546, 0.7045454545454546, 0.7272727272727273, 0.727272727272727
3, 0.7272727272727273, 0.7272727272727273, 0.75, 0.75, 0.7727272727272727, 0.7727272727272727, 0.7954545454545454,
0.7954545454545454, 0.8181818181818182, 0.8181818181818182, 0.8409090909090909, 0.8409090909090909, 0.840909090909
0909, 0.8636363636363636, 0.8863636363636364, 0.9090909090909091, 0.9090909090909091, 0.9318181818181818, 0.931818
1818181818, 0.9318181818181818, 0.9318181818181818, 0.9318181818181818, 0.9318181818181818, 0.9318181818181818, 0.
9318181818181818, 0.9545454545454546, 0.9545454545454546, 0.9545454545454546, 0.9772727272727273, 0.97727272727272
73, 0.9772727272727273, 0.9772727272727273, 0.9772727272727273, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
```

- PR curve:



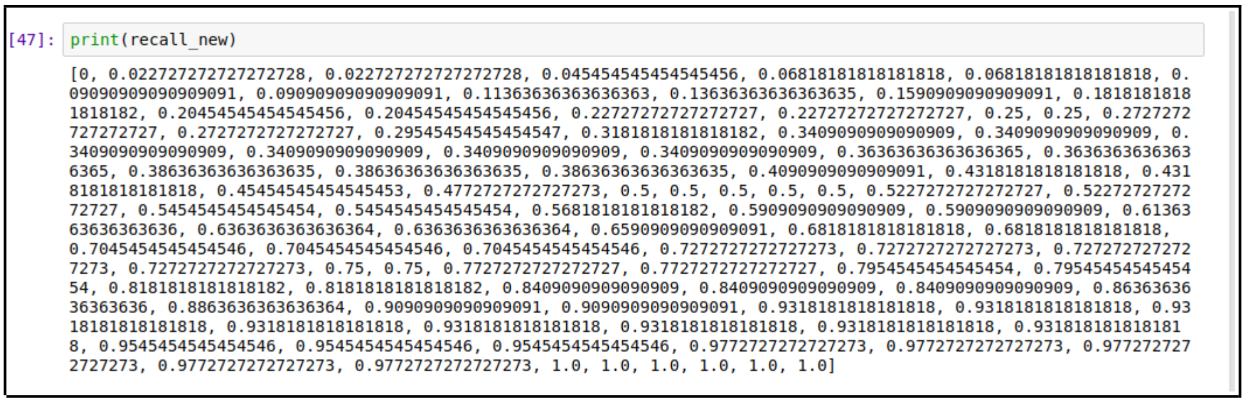
- After appending 1 in precision and 0 in recall data and removing first 4 irreleveant documents:

- Precision:



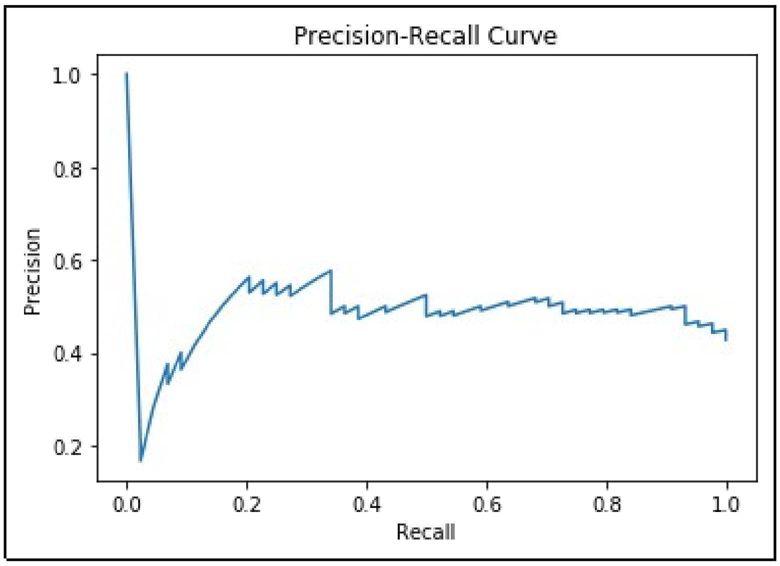
```
[42]: print(precision_new)

      [1, 0.2, 0.16666666666666666, 0.2857142857142857, 0.375, 0.3333333333333333, 0.4, 0.36363636363636365, 0.416666666
      6666667, 0.46153846153846156, 0.5, 0.5333333333333333, 0.5625, 0.5294117647058824, 0.5555555555555556, 0.526315789
      4736842, 0.55, 0.5238095238095238, 0.5454545454545454, 0.5217391304347826, 0.5416666666666666, 0.56, 0.57692307692
      30769, 0.5555555555555556, 0.5357142857142857, 0.5172413793103449, 0.5, 0.4838709677419355, 0.5, 0.484848484848484
      86, 0.5, 0.4857142857142857, 0.4722222222222222, 0.4864864864864865, 0.5, 0.48717948717948717, 0.5, 0.512195121951
      2195, 0.5238095238095238, 0.5116279069767442, 0.5, 0.4888888888888889, 0.4782608695652174, 0.48936170212765956, 0.
      4791666666666667, 0.4897959183673469, 0.48, 0.49019607843137253, 0.5, 0.49056603773584906, 0.5, 0.509090909090909,
      0.5, 0.5087719298245614, 0.5172413793103449, 0.5084745762711864, 0.5166666666666667, 0.5081967213114754, 0.5, 0.50
      79365079365079, 0.5, 0.49230769230769234, 0.48484848484848486, 0.4925373134328358, 0.4852941176470588, 0.492753623
      1884058, 0.4857142857142857, 0.49295774647887325, 0.4861111111111111, 0.4931506849315068, 0.4864864864864865, 0.49
      333333333333335, 0.4868421052631579, 0.4805194805194805, 0.48717948717948717, 0.4936708860759494, 0.5, 0.493827160
      49382713, 0.5, 0.4939759036144578, 0.4880952380952381, 0.4823529411764706, 0.47674418604651164, 0.4712643678160919
      3, 0.4659090909090909, 0.4606741573033708, 0.4666666666666667, 0.46153846153846156, 0.45652173913043476, 0.4623655
      9139784944, 0.4574468085106383, 0.45263157894736844, 0.4479166666666667, 0.44329896907216493, 0.4489795918367347,
      0.4444444444444444, 0.44, 0.43564356435643564, 0.43137254901960786, 0.42718446601941745]

[43]: recall_new = []
```

- Recall:



```
[47]: print(recall_new)

      [0, 0.022727272727272728, 0.022727272727272728, 0.045454545454545456, 0.06818181818181818, 0.06818181818181818, 0.
      09090909090909091, 0.09090909090909091, 0.11363636363636363, 0.13636363636363635, 0.1590909090909091, 0.1818181818
      1818182, 0.20454545454545456, 0.20454545454545456, 0.22727272727272727, 0.22727272727272727, 0.25, 0.25, 0.2727272
      727272727, 0.2727272727272727, 0.29545454545454547, 0.3181818181818182, 0.3409090909090909, 0.3409090909090909, 0.
      3409090909090909, 0.3409090909090909, 0.3409090909090909, 0.3409090909090909, 0.36363636363636365, 0.363636363636363
      6365, 0.38636363636363635, 0.38636363636363635, 0.38636363636363635, 0.4090909090909091, 0.4318181818181818, 0.431
      8181818181818, 0.4545454545454545453, 0.4772727272727273, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5227272727272727, 0.5227272727272
      72727, 0.5454545454545454, 0.5454545454545454, 0.5681818181818182, 0.5909090909090909, 0.5909090909090909, 0.61363
      63636363636, 0.6363636363636364, 0.6363636363636364, 0.6590909090909091, 0.6818181818181818, 0.6818181818181818,
      0.7045454545454546, 0.7045454545454546, 0.7045454545454546, 0.7272727272727273, 0.7272727272727273, 0.727272727272
      7273, 0.7272727272727273, 0.75, 0.75, 0.7727272727272727, 0.7727272727272727, 0.7954545454545454, 0.79545454545454
      54, 0.8181818181818182, 0.8181818181818182, 0.8409090909090909, 0.8409090909090909, 0.8409090909090909, 0.86363636
      36363636, 0.8863636363636364, 0.9090909090909091, 0.9090909090909091, 0.9318181818181818, 0.9318181818181818, 0.93
      18181818181818, 0.9318181818181818, 0.9318181818181818, 0.9318181818181818, 0.9318181818181818, 0.931818181818181
      8, 0.9545454545454546, 0.9545454545454546, 0.9545454545454546, 0.9772727272727273, 0.9772727272727273, 0.977272727
      2727273, 0.9772727272727273, 0.9772727272727273, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
```

- PR curve :



Q3.

1. Explain the relationship between ROC curve and PR curve.

- For both we use confusion mattrices.
- For Plotting a graph, in case of ROC space we use False Positive Rate(FPR) on x-axis and True Positive Rate(TPR) on y-axis and In PR space, one plots Recall which is same as TPR on the x-axis and Precision on the y-axis.
- When recall is non-zero then there exists a same confusion matrix for both the curves.
- When dataset is fixed,each point ROC space have a unique confusion matrix. Since it's true that PR curve doesn't take TN into the consideration then one might worry that there exists more than one confusion matrix for each point. But because we've fixed number of positive and negative samples and recall is not zero, so it's possible to determine true negative rate by using other parametes such as FP,TP and FN. The value obtained for TN will be same as we considered for ROC curve.Thus, from this we can conclude that tere exists an one-to-one mapping between ROC and PR curve.
- Thus, both can be converted into each other.

2. Prove that a curve dominates in ROC space if and only if it dominates in PR space.

- There exists a both ways relation:

- *A curve dominates in ROC then it dominates in PR space:*

  Consider two curves i.e curve 1 and curve 2 in ROC space.In ROC space curve 1 dominates over curve 2 when it's converted to PR space curve 1 no longer dominates. Now there exists a point x2 in curve 2 with higher precision value than point x1 on curve 1 with same recall value.
  Since Recall is same for both the points then, TPR(x1)= TPR(x2).
  Now, in ROC space curve 1 dominates over curve 2 then FPR(x2)>=FPR(x1) so,
  TPR(x1) = TP(x1)/Total positive —- (1)
  TPR(x2) = TP(x2)/Total positive —- (2)

  As we defined earlier that TPR(x1) = TPR(x1) thus from above two equations (1) and (2) we can deduce that TP(x1)=TP(x1)=TP.
  Now,
  FPR(x1) = FP(x1)/Total negative —- (3)
  FPR(x2) = FP(x2)/Total negative —- (4)

  Similarly from above relation FP(x2)>=FP(x1), Since FPR(x2)>=FPR(x1)
  Precision for x1 and x2 given as:

Precision(x1) = TP/TP + FP(x1) —- (5)
Precision (x2) = TP/TP + FP(x2)—- (6)

We know that FP(x2) >= FP(x1) Therefore, Precision(x1)> Precision (x2). This contradicts our assumption.

- *A curve dominates in PR then it dominates in ROC space:*

  Consider two curves i.e curve 1 and curve 2 in PR space.In PR space curve 1 dominates over curve 2 when it's converted to ROC space curve 1 no longer dominates. Now there exists a point x2 in curve 2 with higher precision value than point x1 on curve 1 with same recall value.
  Sice recall of x1 and x2 are same, then TPR(x2) = TPR(x1) .
  But FPR(x2) < FPR(x1).
  Precision of x2 will always be less than precision of x1 because curve 1 dominates over curve 2 in PR space.
  so,
  TPR(x1) = TP(x1)/Total positive —- (1)
  TPR(x2) = TP(x2)/Total positive —- (2)
  As we defined earlier that TPR(x1) = TPR(x1) thus from above two equations (1) and (2) we can deduce that TP(x1)=TP(x1)=TP.

  Precision for x1 and x2 given as:
  Precision(x1) = TP/TP + FP(x1) —- (3)
  Precision (x2) = TP/TP + FP(x2)—- (4)

  We know that precision of x2 will be less than precision of x1, False Positive for x2 will be greater than False Positive for x1.

  Now,
  FPR(x1) = FP(x1)/Total negative —- (5)
  FPR(x2) = FP(x2)/Total negative —- (6)

  We know that FP(x2) >= FP(x1) , Since FPR(x2)>= FPR(x1). This contradicts our assumption.

Therefore, we can say that a curve dominates in ROC space if and only if it dominates in PR space.

3. It is incorrect to interpolate between points in PR space. When and why does this happen? How will you tackle this problem?

- When we interpolate between the points in PR space, it's not mandatory that precision will always change linearly with a change in Recall. This is due in

Precision that fact that FP replaced by FN in denominator. Interpolating in such cases is incorrect as it gives an optimal result of a performance. That’s why it’s wrong to interpolate between points in PR space.

- Consider two points x1 and x2 that are far away from each other in PR space. Now to compute some intermediate value, we need to interpolate between these points i.e x1 and x2 and require the counts of TP and FP as it gives us the point on a plane.We can see the number of negative examples it takes to reach one positive example, which can be given as:

$$FP(x2) – FP(x1) / TP(x2) – TP(x1)$$

- Now we can find all the points ina plane as TP(x1) + x (interpolating between x1 and x2) where x belongs to a range [1, TP(x2) – TP(x1)] and then we can compute FP by linearly increasing the above equation for each new point.