NYU Tandon - CS6923 Fall 2016
Machine Learning Professor Sandoval

Homework 3

Name: Jubin Soni (jas1464)

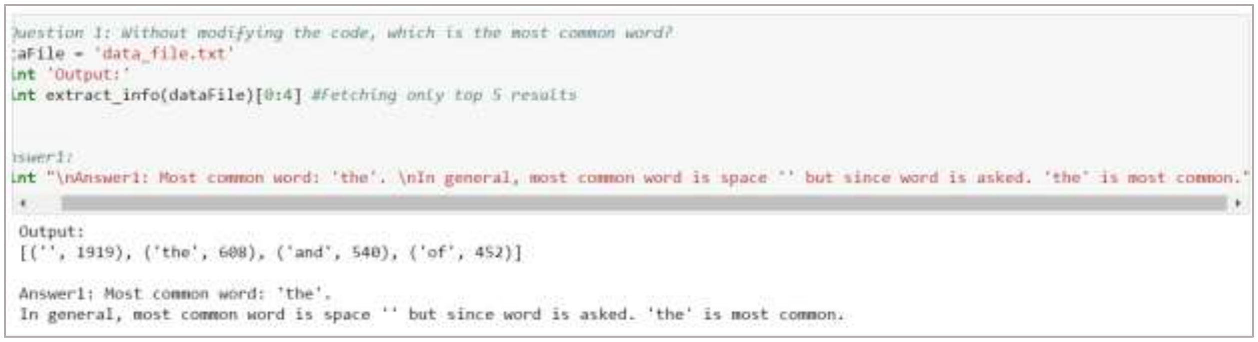
Part-1: Coding Part (Using Python for NLP)

Section A: Term Frequency with unigrams
For code and output of all questions below see my IPython file or click here to view it on my GitHub. I have also attached screenshots here.

Answer 1:

Without any changes, most common word: 'the'. In general, most common word is space '' but since word is asked. 'the' is most common.

Output screenshot:



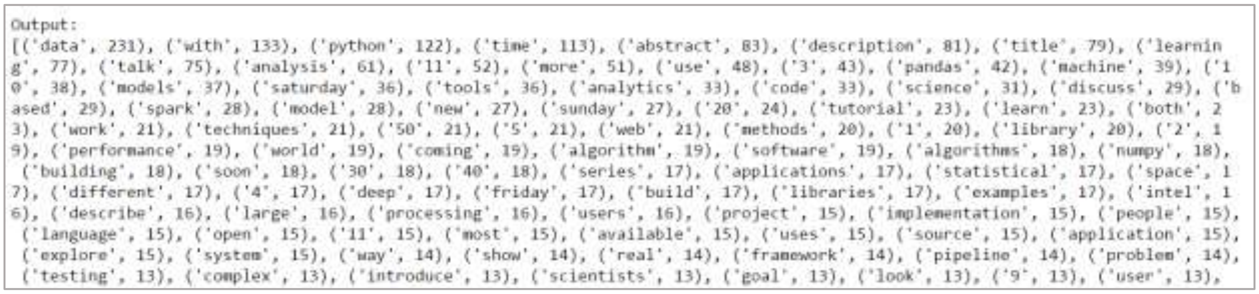
```
Question 1: Without modifying the code, which is the most common word?
:aFile = 'data_file.txt'
int 'Output:'
int extract_info(dataFile)[0:4] #Fetching only top 5 results


iswer1:
int "\nAnswer1: Most common word: 'the'. \nIn general, most common word is space '' but since word is asked. 'the' is most common."
```

```
Output:
[('', 1919), ('the', 608), ('and', 540), ('of', 452)]

Answer1: Most common word: 'the'.
In general, most common word is space '' but since word is asked. 'the' is most common.
```

Answer 2:

After removing stop words, most common 3 words are: 'data', 'with', 'python'

Output screenshot:



```
Output:
[('data', 231), ('with', 133), ('python', 122), ('time', 113), ('abstract', 83), ('description', 81), ('title', 79), ('learnin
g', 77), ('talk', 75), ('analysis', 61), ('11', 52), ('more', 51), ('use', 48), ('3', 43), ('pandas', 42), ('machine', 39), ('1
0', 38), ('models', 37), ('saturday', 36), ('tools', 36), ('analytics', 33), ('code', 33), ('science', 31), ('discuss', 29), ('b
ased', 29), ('spark', 28), ('model', 28), ('new', 27), ('sunday', 27), ('20', 24), ('tutorial', 23), ('learn', 23), ('both', 2
3), ('work', 21), ('techniques', 21), ('50', 21), ('5', 21), ('web', 21), ('methods', 20), ('1', 20), ('library', 20), ('2', 1
9), ('performance', 19), ('world', 19), ('coming', 19), ('algorithm', 19), ('software', 19), ('algorithms', 18), ('numpy', 18),
 ('building', 18), ('soon', 18), ('30', 18), ('40', 18), ('series', 17), ('applications', 17), ('statistical', 17), ('space', 1
7), ('different', 17), ('4', 17), ('deep', 17), ('friday', 17), ('build', 17), ('libraries', 17), ('examples', 17), ('intel', 1
6), ('describe', 16), ('large', 16), ('processing', 16), ('users', 16), ('project', 15), ('implementation', 15), ('people', 15),
 ('language', 15), ('open', 15), ('11', 15), ('most', 15), ('available', 15), ('uses', 15), ('source', 15), ('application', 15),
 ('explore', 15), ('system', 15), ('way', 14), ('show', 14), ('real', 14), ('framework', 14), ('pipeline', 14), ('problem', 14),
 ('testing', 13), ('complex', 13), ('introduce', 13), ('scientists', 13), ('goal', 13), ('look', 13), ('9', 13), ('user', 13),
```

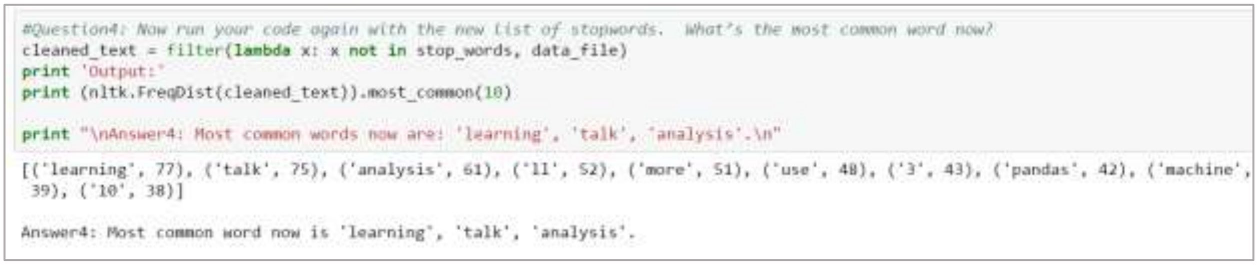
Answer 3:

Since 'with', 'data', 'python' occur in every document, they don't add much value. However, if we look at 50 most common words we should notice more words and numbers like 'll', '30' which are common and don't seem to add much value. But this is subjective to data we want.

Answer 4:

After removing words: 'with', 'data', 'python'. Most common words now are: 'learning', 'talk', 'analysis'.

Output screenshot:



```
#Question4: Now run your code again with the new list of stopwords.  What's the most common word now?
cleaned_text = filter(lambda x: x not in stop_words, data_file)
print 'Output:'
print (nltk.FreqDist(cleaned_text)).most_common(10)

print "\nAnswer4: Most common words now are: 'learning', 'talk', 'analysis'.\n"
```

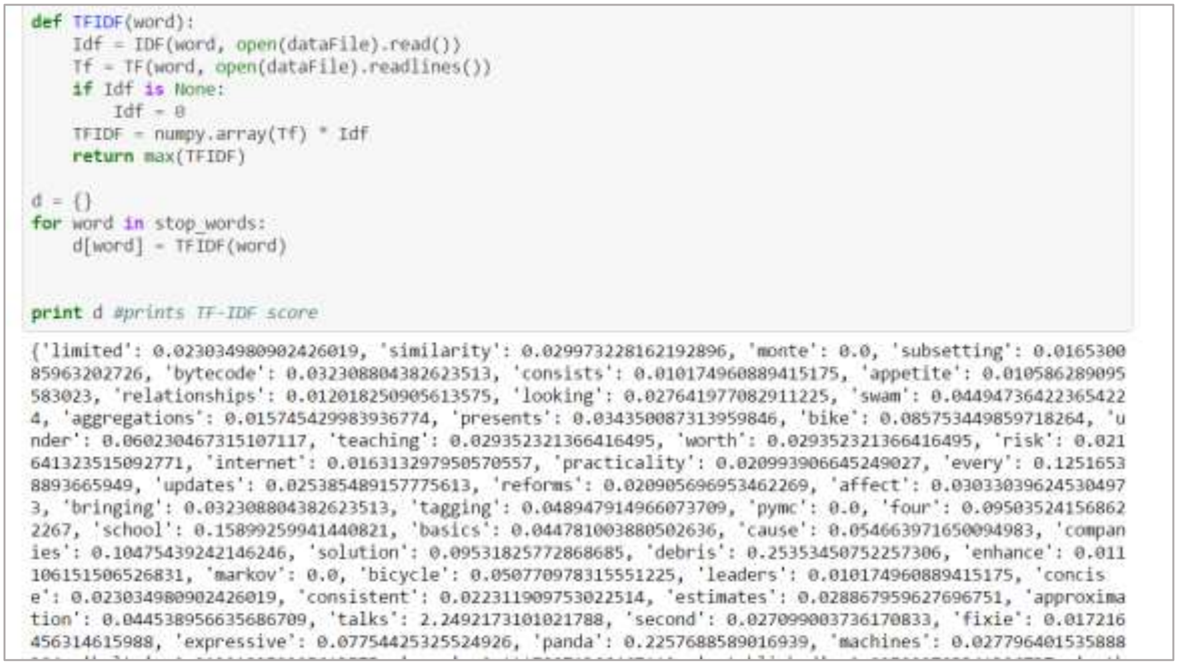
```
[('learning', 77), ('talk', 75), ('analysis', 61), ('ll', 52), ('more', 51), ('use', 48), ('3', 43), ('pandas', 42), ('machine',
 39), ('10', 38)]

Answer4: Most common word now is 'learning', 'talk', 'analysis'.
```

Section B: Using TF-IDF

Answer 5:

After computing TF-IDF score below is the output screenshot. The output is big so entire output is available in attached IPython file and can also be viewed by clicking here, from my GitHub repository.

Output Screenshot:

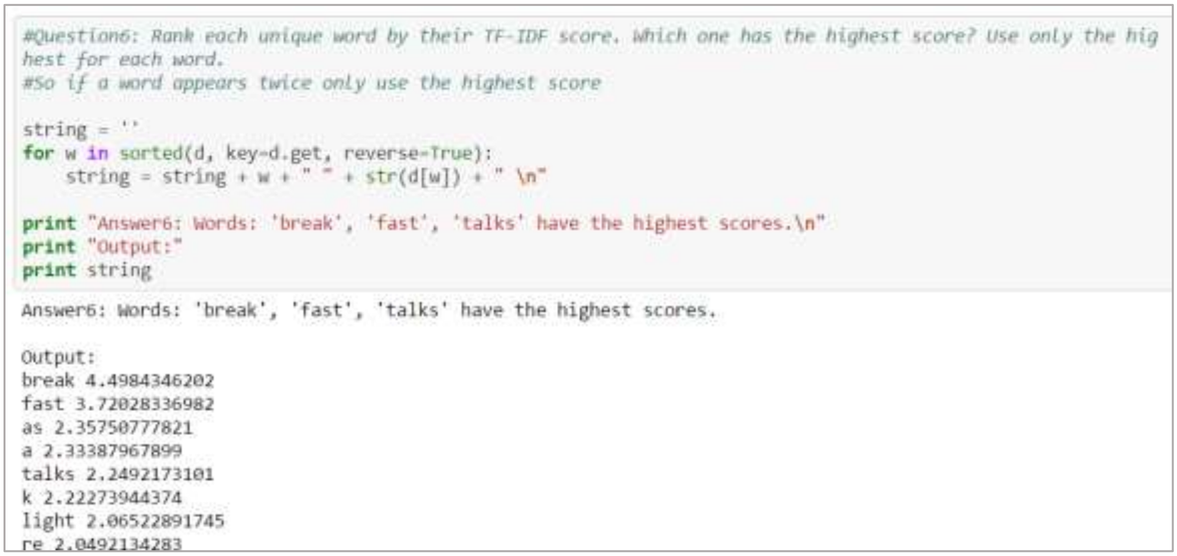


```
def TFIDF(word):
    Idf = IDF(word, open(dataFile).read())
    Tf = TF(word, open(dataFile).readlines())
    if Idf is None:
        Idf = 0
    TFIDF = numpy.array(Tf) * Idf
    return max(TFIDF)

d = {}
for word in stop_words:
    d[word] = TFIDF(word)


print d #prints TF-IDF score
```

```
{'limited': 0.023034980902426019, 'similarity': 0.029973228162192896, 'monte': 0.0, 'subsetting': 0.0165300
85963202726, 'bytecode': 0.032308804382623513, 'consists': 0.010174960889415175, 'appetite': 0.010586289095
583023, 'relationships': 0.012018250905613575, 'looking': 0.027641977082911225, 'swam': 0.04494736422365422
4, 'aggregations': 0.015745429983936774, 'presents': 0.034350087313959846, 'bike': 0.085753449859718264, 'u
nder': 0.060230467315107117, 'teaching': 0.029352321366416495, 'worth': 0.029352321366416495, 'risk': 0.021
641323515092771, 'internet': 0.016313297950570557, 'practicality': 0.020993906645249027, 'every': 0.1251653
8893665949, 'updates': 0.025385489157775613, 'reforms': 0.020905696953462269, 'affect': 0.03033039624530497
3, 'bringing': 0.032308804382623513, 'tagging': 0.048947914966073709, 'pymc': 0.0, 'four': 0.09503524156862
2267, 'school': 0.15899259941440821, 'basics': 0.044781003880502636, 'cause': 0.054663971650094983, 'compan
ies': 0.10475439242146246, 'solution': 0.09531825772868685, 'debris': 0.25353450752257306, 'enhance': 0.011
106151506526831, 'markov': 0.0, 'bicycle': 0.050770978315551225, 'leaders': 0.010174960889415175, 'concis
e': 0.023034980902426019, 'consistent': 0.022311909753022514, 'estimates': 0.028867959627696751, 'approxima
tion': 0.044538956635686709, 'talks': 2.2492173101021788, 'second': 0.027099003736170833, 'fixie': 0.017216
456314615988, 'expressive': 0.07754425325524926, 'panda': 0.2257688589016939, 'machines': 0.027796401535888
```

Answer 6:
Words: ‘break’, ‘fast’ and ‘talks’ have highest scores.
After ranking each unique word with its TF-IDF score below is the output screenshot.
The output is big so entire output is available in attached IPython file and can also be viewed by clicking here, from my GitHub repository.

Output Screenshot:



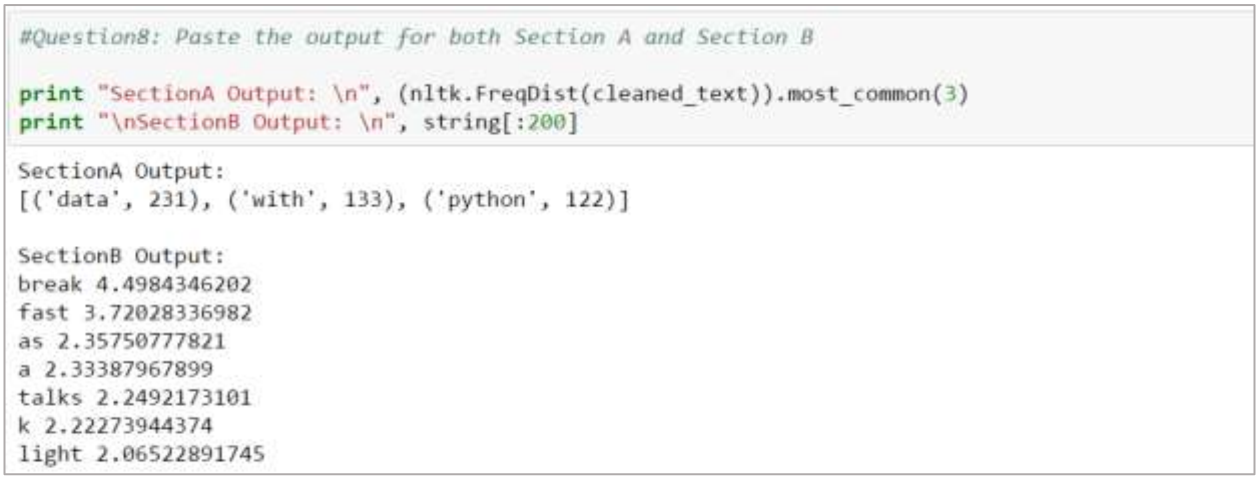
```
#Question6: Rank each unique word by their TF-IDF score. Which one has the highest score? Use only the hig
hest for each word.
#So if a word appears twice only use the highest score

string = ''
for w in sorted(d, key=d.get, reverse=True):
    string = string + w + " " + str(d[w]) + " \n"

print "Answer6: Words: 'break', 'fast', 'talks' have the highest scores.\n"
print "Output:"
print string
```

```
Answer6: Words: 'break', 'fast', 'talks' have the highest scores.

Output:
break 4.4984346202
fast 3.72028336982
as 2.35750777821
a 2.33387967899
talks 2.2492173101
k 2.22273944374
light 2.06522891745
re 2.0492134283
```

Answer 7:
Efficient ways to remove space ‘ ‘ are by using Regex or NLTK word tokenizer. For my code I removed the space using NLTK word_tokenizer, and have extracted just words.

Answer 8:
Below is the output of Section A and Section B. Entire output can be seen in my IPython file or by clicking here, from my GitHub repository.

Output Screenshot:



```
#Question8: Paste the output for both Section A and Section B

print "SectionA Output: \n", (nltk.FreqDist(cleaned_text)).most_common(3)
print "\nSectionB Output: \n", string[:200]
```

```
SectionA Output:
[('data', 231), ('with', 133), ('python', 122)]

SectionB Output:
break 4.4984346202
fast 3.72028336982
as 2.35750777821
a 2.33387967899
talks 2.2492173101
k 2.22273944374
light 2.06522891745
```

Part 2 – Pen and Pencil

Problem 2:
From the given table, True-False Error Condition can be calculated as below:

Book	r (Actual)	y (Predicted)	Errors Condition
War and Peace	+	-	FN
Pattern Recognition	+	+	TP
How to Win Friends and Influence People	-	-	TN
The Philosophical Breakfast Club	+	-	FN
Harry Potter	-	+	FP
Godel Escher Bach	+	+	TP
Photoshop for Dummies	-	-	TN

Now using above calculation, we can total the error conditions for hypothesis h to get:

Error Condition	Total
FN	2
FP	1
TP	2
TN	2

And Recall and Precision can be calculated as follows:

$$\begin{aligned} Recall &= \frac{TP}{TP + FN} \\ &= \frac{2}{2+2} \\ &= \frac{2}{4} \\ &= \frac{1}{2} \\ &\approx 0.5 \end{aligned}$$

$$\begin{aligned} Precision &= \frac{TP}{TP + FP} \\ &= \frac{2}{2+1} \\ &= \frac{2}{3} \\ &\approx 0.66667 \end{aligned}$$

Therefore,

Answer: $\boldsymbol{Recall = \frac{1}{2} = 0.5}$**; and** $\boldsymbol{Precision = \frac{2}{3} = 0.66667.}$

Problem 3:

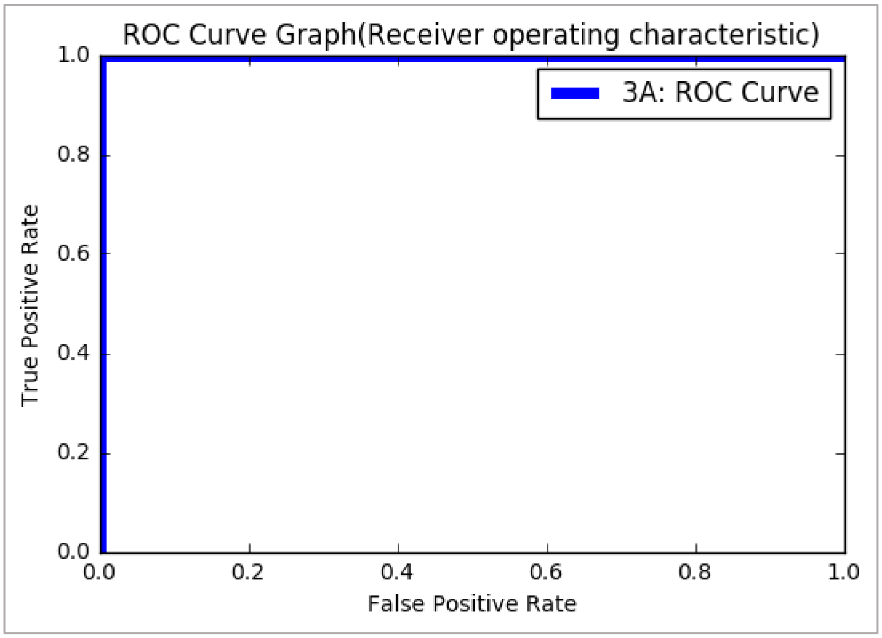
(a) Suppose the labels for the samples, in descending order of values assigned by h are:

$h(x^t)$	x^1	x^2	x^3	x^4	x^5	x^6
$labels$	$+$	$+$	$+$	$-$	$-$	$-$

With different θ, the classifier will predict:

$h(x^t)$	x^1	x^2	x^3	x^4	x^5	x^6	FN	TN	TP	FP	$\boldsymbol{TP\ Rate}$	$\boldsymbol{FP\ Rate}$
θ_0	$-$	$-$	$-$	$-$	$-$	$-$	3	3	0	0	**0**	**0**
θ_1	$+$	$-$	$-$	$-$	$-$	$-$	2	3	1	0	**1/3**	**0**
θ_2	$+$	$+$	$-$	$-$	$-$	$-$	1	3	2	0	**2/3**	**0**
θ_3	$+$	$+$	$+$	$-$	$-$	$-$	0	3	3	0	**1**	**0**
θ_4	$+$	$+$	$+$	$+$	$-$	$-$	0	2	3	1	**1**	**1/3**
θ_5	$+$	$+$	$+$	$+$	$+$	$-$	0	1	3	2	**1**	**2/3**
θ_6	$+$	$+$	$+$	$+$	$+$	$+$	0	0	3	3	**1**	**1**

So, we can plot ROC curve as:



Area under curve can be calculated as:

$$AUC = l * b = 1 * 1 = 1$$

In general, the AUC will be 1 if the predictor h gives all the positive examples higher values than all the negative examples.

Therefore, the AUC (Area under curve) is: 1.

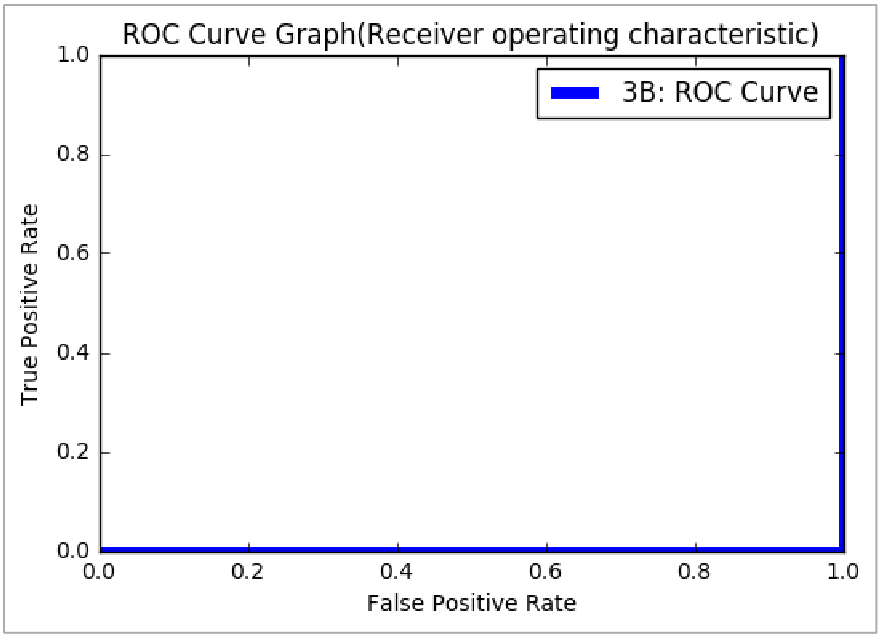
(b) Suppose the labels for the samples, in descending order of values assigned by h are:

$h(x^t)$	x^1	x^2	x^3	x^4	x^5	x^6
$labels$	$-$	$-$	$-$	$+$	$+$	$+$

With different θ, the classifier will predict:

$h(x^t)$	x^1	x^2	x^3	x^4	x^5	x^6	FN	TN	TP	FP	$\boldsymbol{TP\ Rate}$	$\boldsymbol{FP\ Rate}$
θ_0	$-$	$-$	$-$	$-$	$-$	$-$	3	3	0	0	**0**	**0**
θ_1	$+$	$-$	$-$	$-$	$-$	$-$	3	2	0	1	**0**	**1/3**
θ_2	$+$	$+$	$-$	$-$	$-$	$-$	3	1	0	2	**0**	**2/3**
θ_3	$+$	$+$	$+$	$-$	$-$	$-$	3	0	0	3	**0**	**1**
θ_4	$+$	$+$	$+$	$+$	$-$	$-$	2	0	1	3	**1/3**	**1**
θ_5	$+$	$+$	$+$	$+$	$+$	$-$	1	0	2	3	**2/3**	**1**
θ_6	$+$	$+$	$+$	$+$	$+$	$+$	0	0	3	3	**1**	**1**

So, we can plot ROC curve as:



In general, the AUC will be 0 if the predictor h gives all the positive examples lower values than all the negative examples.

Therefore, the AUC (Area under curve) is: 0.

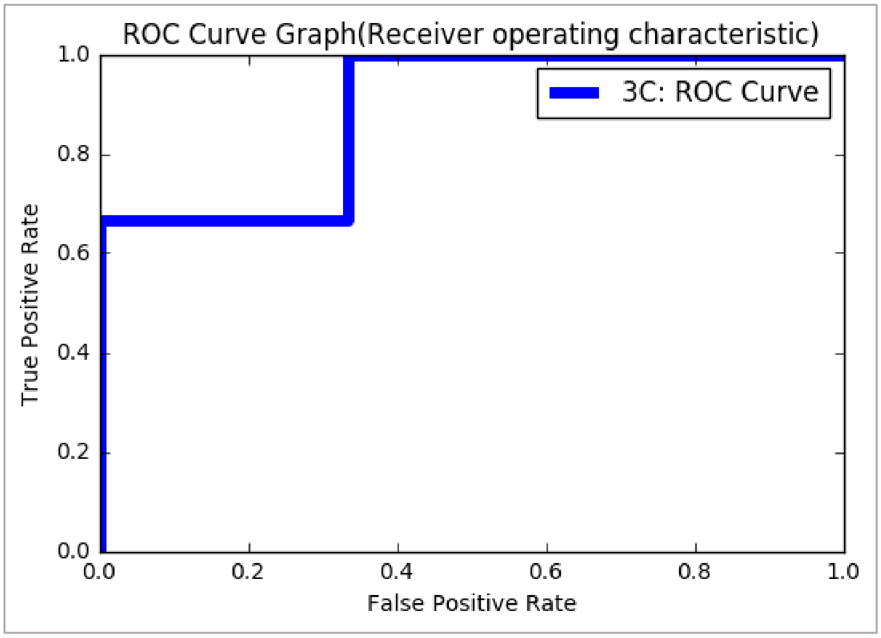
(c) Suppose the labels for the samples, in descending order of values assigned by h are:

$h(x^t)$	x^1	x^2	x^3	x^4	x^5	x^6
$labels$	$+$	$+$	$-$	$+$	$-$	$-$

With different θ, the classifier will predict:

$h(x^t)$	x^1	x^2	x^3	x^4	x^5	x^6	FN	TN	TP	FP	$\boldsymbol{TP\ Rate}$	$\boldsymbol{FP\ Rate}$
θ_0	$-$	$-$	$-$	$-$	$-$	$-$	3	3	0	0	**0**	**0**
θ_1	$+$	$-$	$-$	$-$	$-$	$-$	2	3	1	0	**1/3**	**0**
θ_2	$+$	$+$	$-$	$-$	$-$	$-$	1	3	2	0	**2/3**	**0**
θ_3	$+$	$+$	$+$	$-$	$-$	$-$	1	2	2	1	**2/3**	**1/3**
θ_4	$+$	$+$	$+$	$+$	$-$	$-$	0	2	3	1	**1**	**1/3**
θ_5	$+$	$+$	$+$	$+$	$+$	$-$	0	1	3	2	**1**	**2/3**
θ_6	$+$	$+$	$+$	$+$	$+$	$+$	0	0	3	3	**1**	**1**

So, we can plot ROC curve as:



Area under curve can be calculated as:

$$\begin{aligned} AUC &= (l_1 * b_1) + (l_2 * b_2) + (l_3 * b_3) \\ &= \left(\frac{2}{3} * \frac{2}{3}\right) + \left(\frac{1}{3} * \frac{2}{3}\right) + \left(\frac{1}{3} * \frac{2}{3}\right) \\ &= \frac{4}{9} + 2 * \frac{2}{9} = \frac{4}{9} + \frac{4}{9} \\ &= \frac{8}{9} \end{aligned}$$

The AUC (Area under curve) is: 8/9.