SPARK - Gepi tanulas alkalmazasa

In [1]:

```
import json
import re
import string
```

In [2]:

```
df=sc.textFile("jokecomments2014.txt").repartition(8)
```

In [3]:

```
df.take(1)
```

Out[3]:

```
[u"(u'He was decent.', 28, u'1388535896', u't1_ceeekl1', u't1_ceegaq
j', u't3_1u3kq1')"]
```

In [4]:

```
dfALMA=df.map(lambda x: eval(x))
```

In [5]:

```
def stringok(x):
    s=x
    re.sub(r'\W+', '', s)
    s=s.replace('\n'," ")
    exclude = set(string.punctuation)
    s2 = ''.join(ch for ch in s if ch not in exclude)
    s2=s2.lower()
    s2=s2.strip()
    return s2

dfALMA2=dfALMA.map(lambda x: (stringok(x[0]),x[1]))
dfALMA2.take(1)
```

Out[5]:

```
[(u'he was decent', 28)]
```

In [6]:

```
def szo_es_score(x):
    result=[]
    for i in x[0].split(" "):
        if x[1]<0:
            result.append((i,(x[1],0,1)))
        else:
            result.append((i,(0,x[1],1)))
    return result
dfKORTE=dfALMA2.flatMap(szo_es_score)
dfKORTE.take(5)
```

Out[6]:

```
[(u'he', (0, 28, 1)),
 (u'was', (0, 28, 1)),
 (u'decent', (0, 28, 1)),
 (u'well', (0, 12, 1)),
 (u'he', (0, 12, 1))]
```

In [7]:

```
dfCITROM=dfKORTE.reduceByKey(lambda x,y: (x[0]+y[0],x[1]+y[1],x[2]+y[2]))\
.filter(lambda x: x[0]!=u'')
dfCITROM.take(5)
```

Out[7]:

```
[(u'unimaginative', (0, 19, 1)),
 (u'httpwwwredditcomrnewscomments2da3bqrobinwilliamsfounddeadcjnjrb1',
  (0, 66, 1)),
 (u'httpswwwyoutubecomwatchvvarlfj0w6ua', (0, 11, 1)),
 (u'nun\u201d', (0, 14, 1)),
 (u'nun', (-119, 12645, 165))]
```

In [8]:

```
dfEPER=dfCITROM.map(lambda x: [x[0],x[1][0],x[1][1],x[1][2]])
```

In [13]:

```
dfGORIDINNYE=dfEPER.map(lambda x:  x + [(float(x[1]+1))/float(x[2]+1)])
dfGORIDINNYE.take(4)
```

Out[13]:

```
[[u'unimaginative', 0, 19, 1, 0.05],
 [u'httpwwwredditcomrnewscomments2da3bqrobinwilliamsfounddeadcjnjrb1',
  0,
  66,
  1,
  0.014925373134328358],
 [u'httpswwwyoutubecomwatchvvarlfj0w6ua', 0, 11, 1, 0.0833333333333333
3],
 [u'nun\u201d', 0, 14, 1, 0.06666666666666667]]
```

UJ RESZ INNEN JON

In [14]:

```
# Levalogatjuk a szavakat, amik a legerosebb pozitivak
# Ujdonsag: collect() - mindet leszedo take

ww1=dfGORIDINNYE.filter(lambda x: x[3]>300)\
.filter(lambda x: x[4]>-0.03)\
.map(lambda x: x[0])\
.collect()

print("Kivalogatottak:",len(ww1))
```

```
('Kivalogatottak:', 152)
```

In [15]:

```
#Ugyanugy a legjobban negativba hajlokat

ww2=dfGORIDINNYE.filter(lambda x: x[3]>300)\
.filter(lambda x: x[4]<-0.04)\
.map(lambda x: x[0])\
.collect()

print("Kivalogatottak:",len(ww2))
```

```
('Kivalogatottak:', 113)
```

In [16]:

```
#A ket nagy csapat osszerakasa
ww=ww1+ww2
```

In [17]:

```
# Gepi tanulasnal ugynevezett tanulo pontok kellenek
from pyspark.mllib.regression import LabeledPoint
from numpy import array
```

In [19]:

```
# Egy fuggveny, ami egy commenthez osszeszedi mely szavak szerepeltek
# a ww litaban.
def genvector(x,ww):
    line=x.split(" ")
    r=[]
    for w in ww:
        if w in line:
            r.append(1)
        else:
            r.append(0)
    return r

dfMATRIX=dfALMA2.map(\
lambda x:\
LabeledPoint(1, genvector(x[0],ww)) if x[1]>0 else LabeledPoint(0, genvector(x[0],w



dfMATRIX.take(3)
```

Out[19]:

```
[LabeledPoint(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,
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.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.
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,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
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,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.
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,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
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,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
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,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]),
 LabeledPoint(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,
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.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
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.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.
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,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
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,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
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,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]),
 LabeledPoint(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,
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.
0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.
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,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
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,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
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.
0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,
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])]
```

In [20]:

```
# LOGISZTIKUS REGRESSZIO MODSZERE

from pyspark.mllib.classification import LogisticRegressionWithLBFGS
from time import time

# SZETSZEDJUK A MODELT TANITORA ES TESZTRE
(trainingData, testData) = dfMATRIX.randomSplit([0.8, 0.2])

# Build the model
t0 = time()
logit_model = LogisticRegressionWithLBFGS.train(trainingData)


tt = time() - t0

print "Classifier trained in {} seconds".format(round(tt,3))
```

```
Classifier trained in 13.135 seconds
```

In [21]:

```
# KIDERUL, HOGY MELYIK SZO A MATRIXBAN MILYEN HATASU
sorted(zip(list(logit_model.weights),ww))
```

Out[21]:

```
[(-0.94527570216630341, u'edit'),
 (-0.88446508791418155, u'saying'),
 (-0.88044693654892192, u'funny'),
 (-0.75388657096097955, u'woman'),
 (-0.62436417116453169, u'sense'),
 (-0.61609291336989491, u'husband'),
 (-0.61373905505114112, u'post'),
 (-0.58062213154820186, u'which'),
 (-0.5527822557230424, u'being'),
 (-0.55217610941151174, u'person'),
 (-0.53628543307765486, u'having'),
 (-0.52013770132361481, u'women'),
 (-0.45799822389046319, u'asks'),
 (-0.45647752165076116, u'whole'),
 (-0.43163711283631251, u'without'),
 (-0.42513596524980612, u'comment'),
 (-0.41227898696170229, u'people'),
 (-0.41109511305873908, u'punchline'),
```

In [36]:

```
#Mire mit mond a modellunk a test halmazban
labels_and_preds =  testData.map(lambda p: (p.label, logit_model.predict(p.features
```

In [37]:

```
#Kiszamitjuk a pontossagot
t0 = time()
test_accuracy = labels_and_preds.filter(lambda (v, p): v == p).count() / float(labe
tt = time() - t0
print(tt,"sec - ACC:",test_accuracy)
```

```
(4.568814992904663, 'sec - ACC:', 0.7808924485125858)
```

In [38]:

```
#RESZLETESEBBEN
```

In [43]:

```
print("ARANYOK")
a = labels_and_preds.filter(lambda (v, p): v == 1 and p == 1).count() / float(labels
print("JO    komment volt, JO    kommentnek tippeltuk:",a)
a = labels_and_preds.filter(lambda (v, p): v == 1 and p == 0).count() / float(labels
print("JO    komment volt, ROSSZ kommentnek tippeltuk:",a)

a = labels_and_preds.filter(lambda (v, p): v == 0 and p == 1).count() / float(labels
print("ROSSZ komment volt, JO    kommentnek tippeltuk:",a)
a = labels_and_preds.filter(lambda (v, p): v == 0 and p == 0).count() / float(labels
print("ROSSZ komment volt, ROSSZ kommentnek tippeltuk:",a)
```

```
ARANYOK
('JO    komment volt, JO    kommentnek tippeltuk:', 0.743707093821510
3)
('JO    komment volt, ROSSZ kommentnek tippeltuk:', 0.1491990846681922
3)
('ROSSZ komment volt, JO    kommentnek tippeltuk:', 0.0699084668192219
7)
('ROSSZ komment volt, ROSSZ kommentnek tippeltuk:', 0.0371853546910755
1)
```

In [ ]: