ILS - Z 604: Assignment #3

Due on Tuesday, April 19, 2016

Prof. Xiazhong Liu

vpatani (Vivek Patani)

Contents

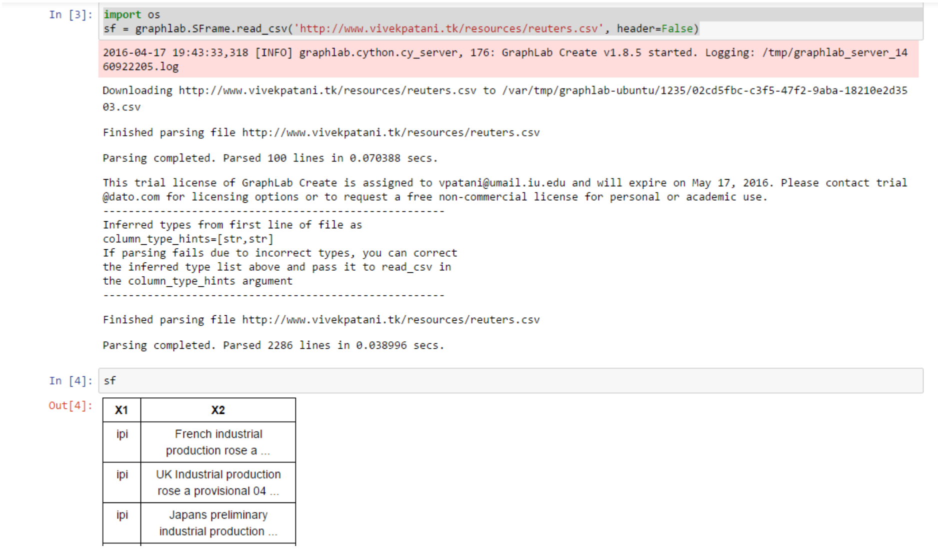
Task 1

Listing 1 Shows the IPynb Script.

Listing 1: Shows the IPynb Script for cleaning

```
#!/usr/bin/perl

#Part 1
import os
import graphlab
graphlab.product_key.set_product_key('Your API Key')
location = 'http://www.vivekpatani.tk/resources/reuters.csv'
sf = graphlab.SFrame.read_csv(location, header=False)
```

- How to configure the graphLab for AWS:
  - Register as a student and Dato will send you a key.
  - Register the key as *graphlab.product − key* function as shown above.
  - Execute in order to register



```
In [3]: import os
        sf = graphlab.SFrame.read_csv('http://www.vivekpatani.tk/resources/reuters.csv', header=False)

2016-04-17 19:43:33,318 [INFO] graphlab.cython.cy_server, 176: GraphLab Create v1.8.5 started. Logging: /tmp/graphlab_server_14
60922205.log

Downloading http://www.vivekpatani.tk/resources/reuters.csv to /var/tmp/graphlab-ubuntu/1235/02cd5fbc-c3f5-47f2-9aba-18210e2d35
03.csv

Finished parsing file http://www.vivekpatani.tk/resources/reuters.csv

Parsing completed. Parsed 100 lines in 0.070388 secs.

This trial license of GraphLab Create is assigned to vpatani@umail.iu.edu and will expire on May 17, 2016. Please contact trial
@dato.com for licensing options or to request a free non-commercial license for personal or academic use.
------------------------------------------------------
Inferred types from first line of file as
column_type_hints=[str,str]
If parsing fails due to incorrect types, you can correct
the inferred type list above and pass it to read_csv in
the column_type_hints argument
------------------------------------------------------

Finished parsing file http://www.vivekpatani.tk/resources/reuters.csv

Parsing completed. Parsed 2286 lines in 0.038996 secs.

In [4]: sf

Out[4]:
```

X1	X2
ipi	French industrial production rose a ...
ipi	UK Industrial production rose a provisional 04 ...
ipi	Japans preliminary industrial production ...

Listing 2 Shows the IPynb Script.

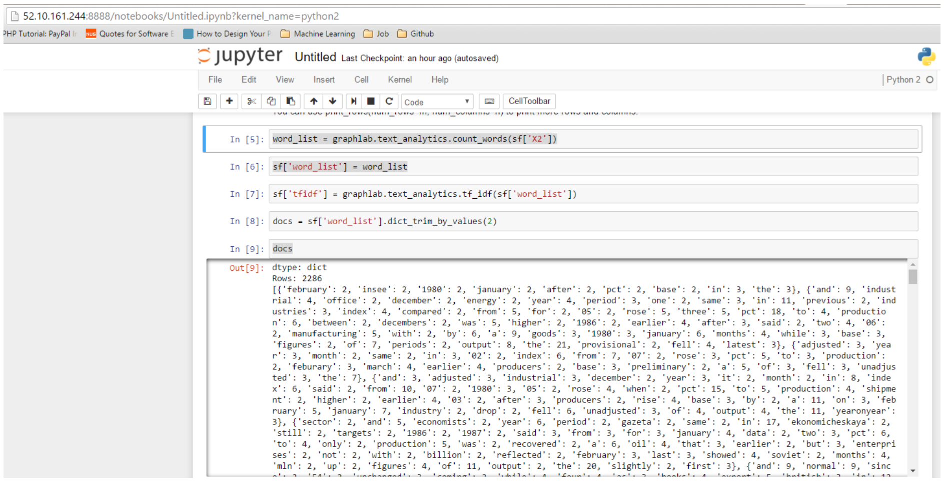
Listing 2: Shows the IPynb Script for counting

```

#Part 2
word_list = graphlab.text_analytics.count_words(sf['X2'])
docs = sf['word_list'].dict_trim_by_values(2)
docs = docs.dict_trim_by_keys(graphlab.text_analytics.stopwords(),exclude=True)
```

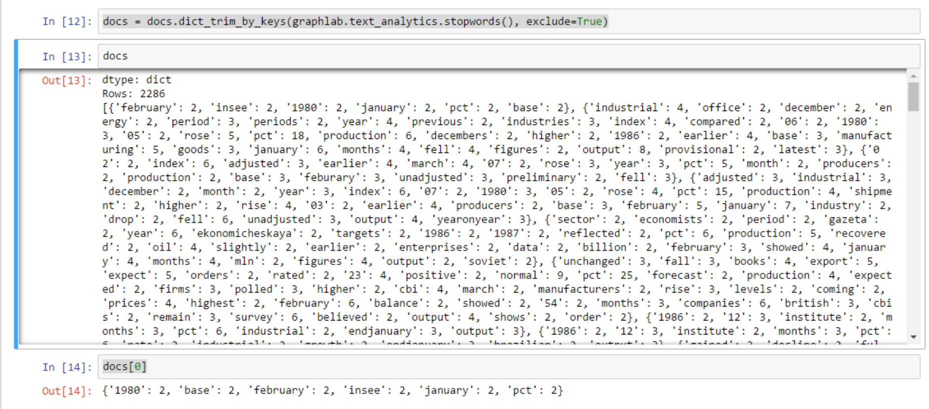
- Now we count the words and eliminate the stop words and words not crossing threshold
  - Use the *SFrame.read − csv* function giving input of the location of your file.
  - View sf to confirm the number of rows.

Task 1 continued on next page. . .

- Once we have the data ready, we use the libraries to count the words that exist in the dataset. We use an Sarray.SArray to store the word and their count.
- The next step is to see what words have appeared lesser than 2 times and we pass a parameter as 2 to the function $trim - by - value$.
- Next step is to eliminate the stop word. We pass *doc* as the input and the the output is over ridden on *doc* itself by passing it through *stopwords* function.
- Just to make sure we've got it right, just check *doc*[0].



```
In [5]: word_list = graphlab.text_analytics.count_words(sf['X2'])

In [6]: sf['word_list'] = word_list

In [7]: sf['tfidf'] = graphlab.text_analytics.tf_idf(sf['word_list'])

In [8]: docs = sf['word_list'].dict_trim_by_values(2)

In [9]: docs

Out[9]: dtype: dict
        Rows: 2286
        [{'february': 2, 'insee': 2, '1980': 2, 'january': 2, 'after': 2, 'pct': 2, 'base': 2, 'in': 3, 'the': 3}, {'and': 9, 'indust
        rial': 4, 'office': 2, 'december': 2, 'energy': 2, 'year': 4, 'period': 3, 'one': 2, 'same': 3, 'in': 11, 'previous': 2, 'ind
        ustries': 3, 'index': 4, 'compared': 2, 'from': 5, 'for': 2, '05': 2, 'rose': 5, 'three': 5, 'pct': 18, 'to': 4, 'productio
        n': 6, 'between': 2, 'decembers': 2, 'was': 5, 'higher': 2, '1986': 2, 'earlier': 4, 'after': 3, 'said': 2, 'two': 4, '06':
        2, 'manufacturing': 5, 'with': 2, 'by': 6, 'a': 9, 'goods': 3, '1980': 3, 'january': 6, 'months': 4, 'while': 3, 'base': 3,
        'figures': 2, 'of': 7, 'periods': 2, 'output': 8, 'the': 21, 'provisional': 2, 'fell': 4, 'latest': 3}, {'adjusted': 3, 'yea
        r': 3, 'month': 2, 'same': 2, 'in': 3, '02': 2, 'index': 6, 'from': 7, '07': 2, 'rose': 3, 'pct': 5, 'to': 3, 'production':
        2, 'feburary': 3, 'march': 4, 'earlier': 4, 'producers': 2, 'base': 3, 'preliminary': 2, 'a': 5, 'of': 3, 'fell': 3, 'unadjus
        ted': 3, 'the': 7}, {'and': 3, 'adjusted': 3, 'industrial': 3, 'december': 2, 'year': 3, 'it': 2, 'month': 2, 'in': 8, 'inde
        x': 6, 'said': 2, 'from': 10, '07': 2, '1980': 3, '05': 2, 'rose': 4, 'when': 2, 'pct': 15, 'to': 5, 'production': 4, 'shipme
        nt': 2, 'higher': 2, 'earlier': 4, '03': 2, 'after': 3, 'producers': 2, 'rise': 4, 'base': 3, 'by': 2, 'a': 11, 'on': 3, 'feb
        ruary': 5, 'january': 7, 'industry': 2, 'drop': 2, 'fell': 6, 'unadjusted': 3, 'of': 4, 'output': 4, 'the': 11, 'yearonyear':
        3}, {'sector': 2, 'and': 5, 'economists': 2, 'year': 6, 'period': 2, 'gazeta': 2, 'same': 2, 'in': 17, 'ekonomicheskaya': 2,
        'still': 2, 'targets': 2, '1986': 2, '1987': 2, 'said': 3, 'from': 3, 'for': 3, 'january': 4, 'data': 2, 'two': 3, 'pct': 6,
        'to': 4, 'only': 2, 'production': 5, 'was': 2, 'recovered': 2, 'a': 6, 'oil': 4, 'that': 3, 'earlier': 2, 'but': 3, 'enterpri
        ses': 2, 'not': 2, 'with': 2, 'billion': 2, 'reflected': 2, 'february': 3, 'last': 3, 'showed': 4, 'soviet': 2, 'months': 4,
        'mln': 2, 'up': 2, 'figures': 4, 'of': 11, 'output': 2, 'the': 20, 'slightly': 2, 'first': 3}, {'and': 9, 'normal': 9, 'sinc
```



```
In [12]: docs = docs.dict_trim_by_keys(graphlab.text_analytics.stopwords(), exclude=True)

In [13]: docs

Out[13]: dtype: dict
         Rows: 2286
         [{'february': 2, 'insee': 2, '1980': 2, 'january': 2, 'pct': 2, 'base': 2}, {'industrial': 4, 'office': 2, 'december': 2, 'en
         ergy': 2, 'period': 3, 'periods': 2, 'year': 4, 'previous': 2, 'industries': 3, 'index': 4, 'compared': 2, '06': 2, '1980':
         3, '05': 2, 'rose': 5, 'pct': 18, 'production': 6, 'decembers': 2, 'higher': 2, '1986': 2, 'earlier': 4, 'base': 3, 'manufact
         uring': 5, 'goods': 3, 'january': 6, 'months': 4, 'fell': 4, 'figures': 2, 'output': 8, 'provisional': 2, 'latest': 3}, {'0
         2': 2, 'index': 6, 'adjusted': 3, 'earlier': 4, 'march': 4, '07': 2, 'rose': 3, 'year': 3, 'pct': 5, 'month': 2, 'producers':
         2, 'production': 2, 'base': 3, 'feburary': 3, 'unadjusted': 3, 'preliminary': 2, 'fell': 3}, {'adjusted': 3, 'industrial': 3,
         'december': 2, 'month': 2, 'year': 3, 'index': 6, '07': 2, '1980': 3, '05': 2, 'rose': 4, 'pct': 15, 'production': 4, 'shipme
         nt': 2, 'higher': 2, 'rise': 4, '03': 2, 'earlier': 4, 'producers': 2, 'base': 3, 'february': 5, 'january': 7, 'industry': 2,
         'drop': 2, 'fell': 6, 'unadjusted': 3, 'output': 4, 'yearonyear': 3}, {'sector': 2, 'economists': 2, 'period': 2, 'gazeta':
         2, 'year': 6, 'ekonomicheskaya': 2, 'targets': 2, '1986': 2, '1987': 2, 'reflected': 2, 'pct': 6, 'production': 5, 'recovere
         d': 2, 'oil': 4, 'slightly': 2, 'earlier': 2, 'enterprises': 2, 'data': 2, 'billion': 2, 'february': 3, 'showed': 4, 'januar
         y': 4, 'months': 4, 'mln': 2, 'figures': 4, 'output': 2, 'soviet': 2}, {'unchanged': 3, 'fall': 3, 'books': 4, 'export': 5,
         'expect': 5, 'orders': 2, 'rated': 2, '23': 4, 'positive': 2, 'normal': 9, 'pct': 25, 'forecast': 2, 'production': 4, 'expect
         ed': 2, 'firms': 3, 'polled': 3, 'higher': 2, 'cbi': 4, 'march': 2, 'manufacturers': 2, 'rise': 3, 'levels': 2, 'coming': 2,
         'prices': 4, 'highest': 2, 'february': 6, 'balance': 2, 'showed': 2, '54': 2, 'months': 3, 'companies': 6, 'british': 3, 'cbi
         s': 2, 'remain': 3, 'survey': 6, 'believed': 2, 'output': 4, 'shows': 2, 'order': 2}, {'1986': 2, '12': 3, 'institute': 2, 'm
         onths': 3, 'pct': 6, 'industrial': 2, 'endjanuary': 3, 'output': 3}, {'1986': 2, '12': 3, 'institute': 2, 'months': 3, 'pct':

In [14]: docs[0]

Out[14]: {'1980': 2, 'base': 2, 'february': 2, 'insee': 2, 'january': 2, 'pct': 2}
```

- This completes the first part of **cleaning data** and **generating features**.

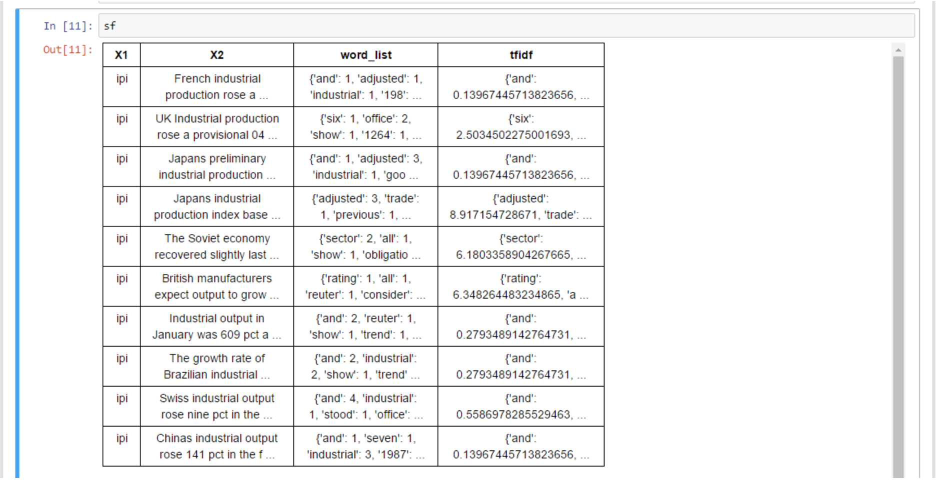
Task 2

Listing 3 Shows the IPynb Script.

Listing 3: Shows the IPynb Script for generating Features

```

#Part 3
word_list = graphlab.text_analytics.count_words(sf['X2'])
sf['word_list'] = word_list
sf['tfidf'] = graphlab.text_analytics.tf_idf(sf['word_list'])
```

- Here the we simply create a list(bag) of words for each document.
- This is straightforward since we just generate a word list by the $count_words$ function by giving input as the the feature column over which we need to do this.
- Next step is to add it the frame generated by us.
- Next run the TF-IDF from the analytics library of graph lab over the word list generated.
- We generated two things
  1. Word List
  2. TF-IDF Values
- This is how the Frame will look like



```
In [11]: sf
Out[11]:
```

X1	X2	word_list	tfidf
ipi	French industrial production rose a ...	{'and': 1, 'adjusted': 1, 'industrial': 1, '198': ...	{'and': 0.13967445713823656, ...
ipi	UK Industrial production rose a provisional 04 ...	{'six': 1, 'office': 2, 'show': 1, '1264': 1, ...	{'six': 2.5034502275001693, ...
ipi	Japans preliminary industrial production ...	{'and': 1, 'adjusted': 3, 'industrial': 1, 'goo ...	{'and': 0.13967445713823656, ...
ipi	Japans industrial production index base ...	{'adjusted': 3, 'trade': 1, 'previous': 1, ...	{'adjusted': 8.917154728671, 'trade': ...
ipi	The Soviet economy recovered slightly last ...	{'sector': 2, 'all': 1, 'show': 1, 'obligatio ...	{'sector': 6.1803358904267665, ...
ipi	British manufacturers expect output to grow ...	{'rating': 1, 'all': 1, 'reuter': 1, 'consider': ...	{'rating': 6.348264483234865, 'a ...
ipi	Industrial output in January was 609 pct a ...	{'and': 2, 'reuter': 1, 'show': 1, 'trend': 1, ...	{'and': 0.2793489142764731, ...
ipi	The growth rate of Brazilian industrial ...	{'and': 2, 'industrial': 2, 'show': 1, 'trend' ...	{'and': 0.2793489142764731, ...
ipi	Swiss industrial output rose nine pct in the ...	{'and': 4, 'industrial': 1, 'stood': 1, 'office': ...	{'and': 0.5586978285529463, ...
ipi	Chinas industrial output rose 141 pct in the f ...	{'and': 1, 'seven': 1, 'industrial': 3, '1987': ...	{'and': 0.13967445713823656, ...

- Now, we have attached the **features** that will help us analyse the data.

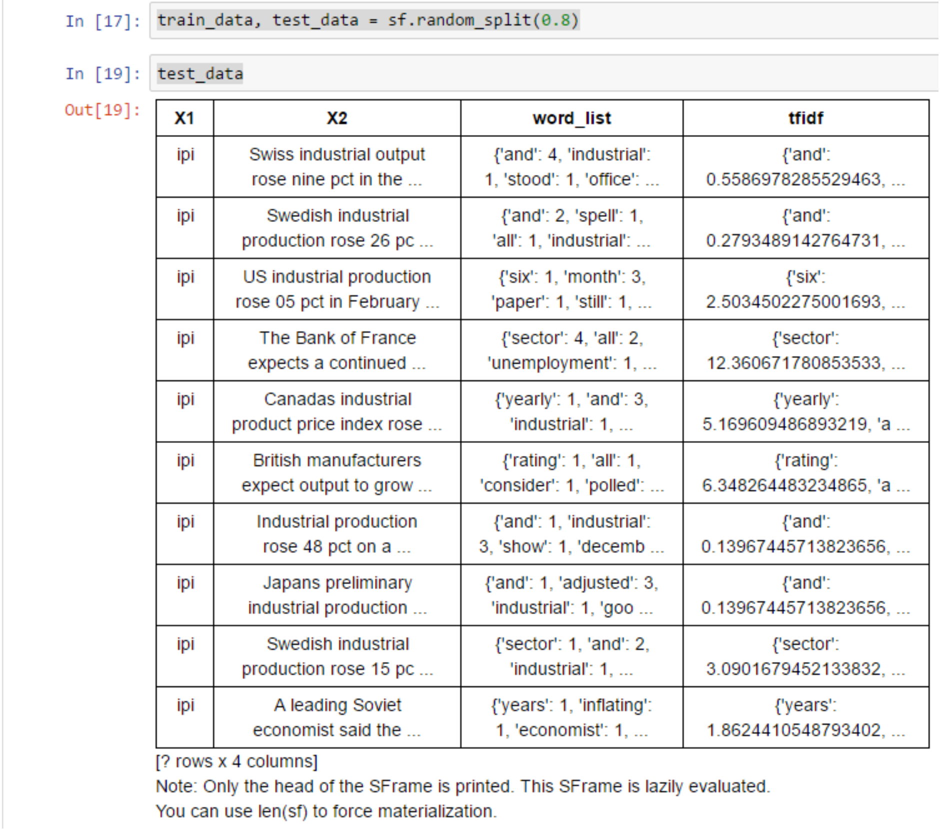
Task 3

Listing 4 Shows the IPynb Script.

Listing 4: Shows the IPynb Script for generating Model & Evaluating it

```
#Part 4
train_data, test_data = sf.random_split(0.8)
test_data
len(sf)
len(test_data)
len(train_data)
```

- In this we have cleaned the data and build the elements required to build a model.
- We first divide the set into **Train** & **Test** in order to build and evaluate the model. We split the data in the ratio of 80:20, where 80 is train and 20 is test.
- This is how the Train & Test Frame will look like

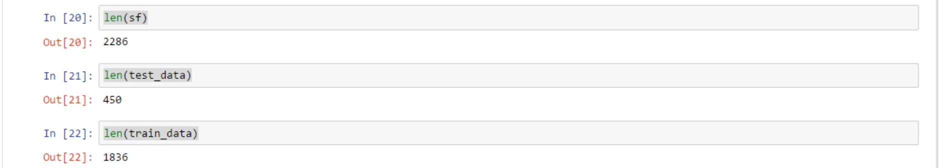


```
In [17]: train_data, test_data = sf.random_split(0.8)

In [19]: test_data

Out[19]:
```

X1	X2	word_list	tfidf
ipi	Swiss industrial output rose nine pct in the ...	{'and': 4, 'industrial': 1, 'stood': 1, 'office': ...	{'and': 0.5586978285529463, ...
ipi	Swedish industrial production rose 26 pc ...	{'and': 2, 'spell': 1, 'all': 1, 'industrial': ...	{'and': 0.2793489142764731, ...
ipi	US industrial production rose 05 pct in February ...	{'six': 1, 'month': 3, 'paper': 1, 'still': 1, ...	{'six': 2.5034502275001693, ...
ipi	The Bank of France expects a continued ...	{'sector': 4, 'all': 2, 'unemployment': 1, ...	{'sector': 12.360671780853533, ...
ipi	Canadas industrial product price index rose ...	{'yearly': 1, 'and': 3, 'industrial': 1, ...	{'yearly': 5.169609486893219, 'a ...
ipi	British manufacturers expect output to grow ...	{'rating': 1, 'all': 1, 'consider': 1, 'polled': ...	{'rating': 6.348264483234865, 'a ...
ipi	Industrial production rose 48 pct on a ...	{'and': 1, 'industrial': 3, 'show': 1, 'decemb ...	{'and': 0.13967445713823656, ...
ipi	Japans preliminary industrial production ...	{'and': 1, 'adjusted': 3, 'industrial': 1, 'goo ...	{'and': 0.13967445713823656, ...
ipi	Swedish industrial production rose 15 pc ...	{'sector': 1, 'and': 2, 'industrial': 1, ...	{'sector': 3.0901679452133832, ...
ipi	A leading Soviet economist said the ...	{'years': 1, 'inflating': 1, 'economist': 1, ...	{'years': 1.8624410548793402, ...

```
[? rows x 4 columns]
Note: Only the head of the SFrame is printed. This SFrame is lazily evaluated.
You can use len(sf) to force materialization.
```

- The length of each is



```
In [20]: len(sf)
Out[20]: 2286

In [21]: len(test_data)
Out[21]: 450

In [22]: len(train_data)
Out[22]: 1836
```

- We use train to learn the model and data.

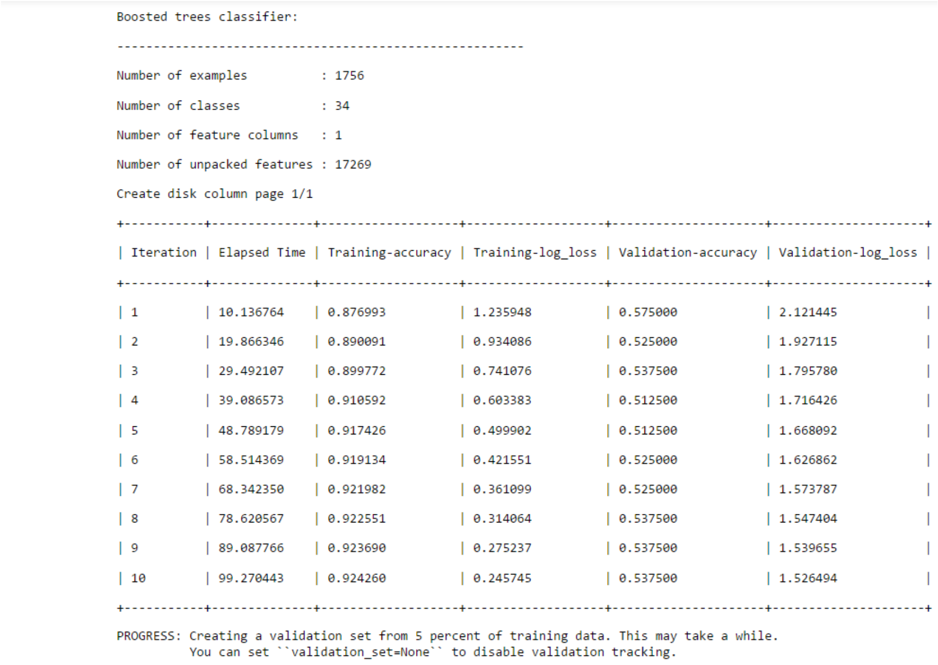
Task 3 continued on next page. . .

- Test will be used to evaluate features of the model such as accuracy.
- Now we start building models with different approaches:
    1. Boosted Tree Classifier: Listing 5 Shows the IPynb Script.

Listing 5: Shows the IPynb Script for generating Model & Evaluating it

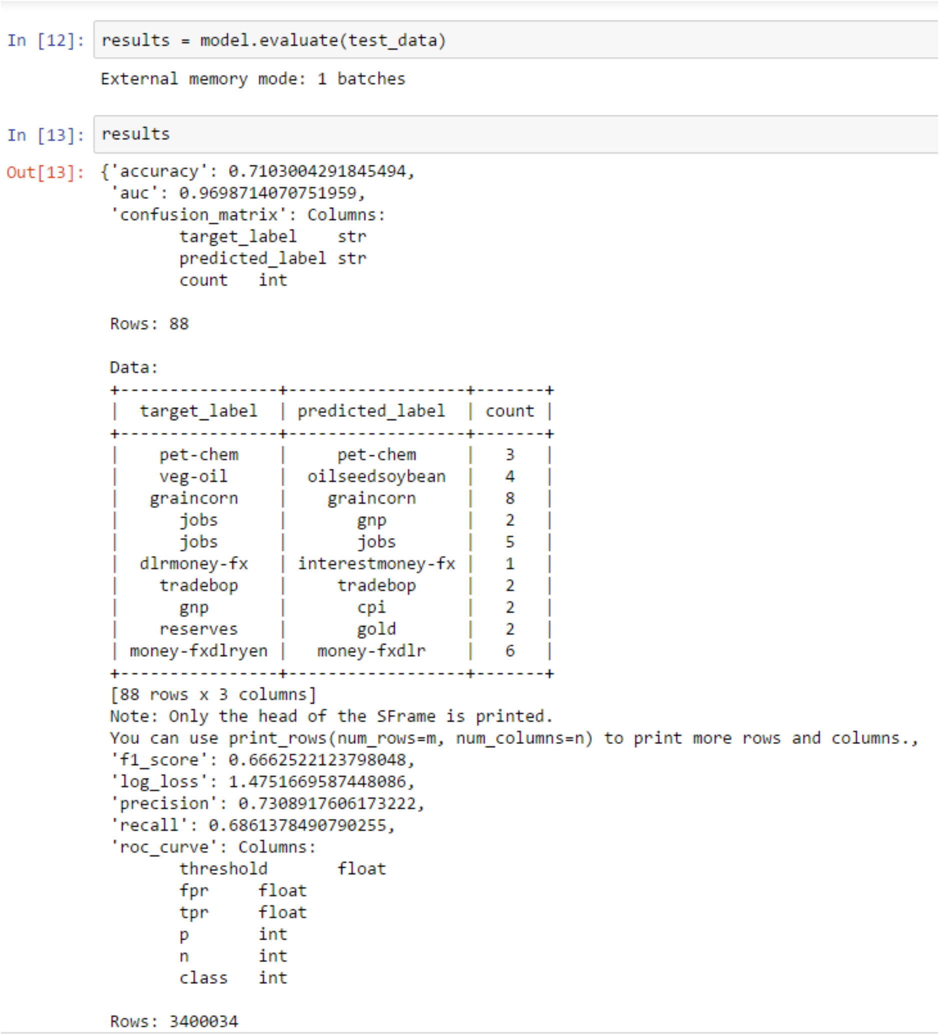
```
model = graphlab.boosted_trees_classifier.create(train_data, target=’X1’,
                                           max_iterations=2,
                                           max_depth = 3)
predictions = model.classify(test_data)
results = model.evaluate(test_data)
```

- After training the data with the given model.
- Output:



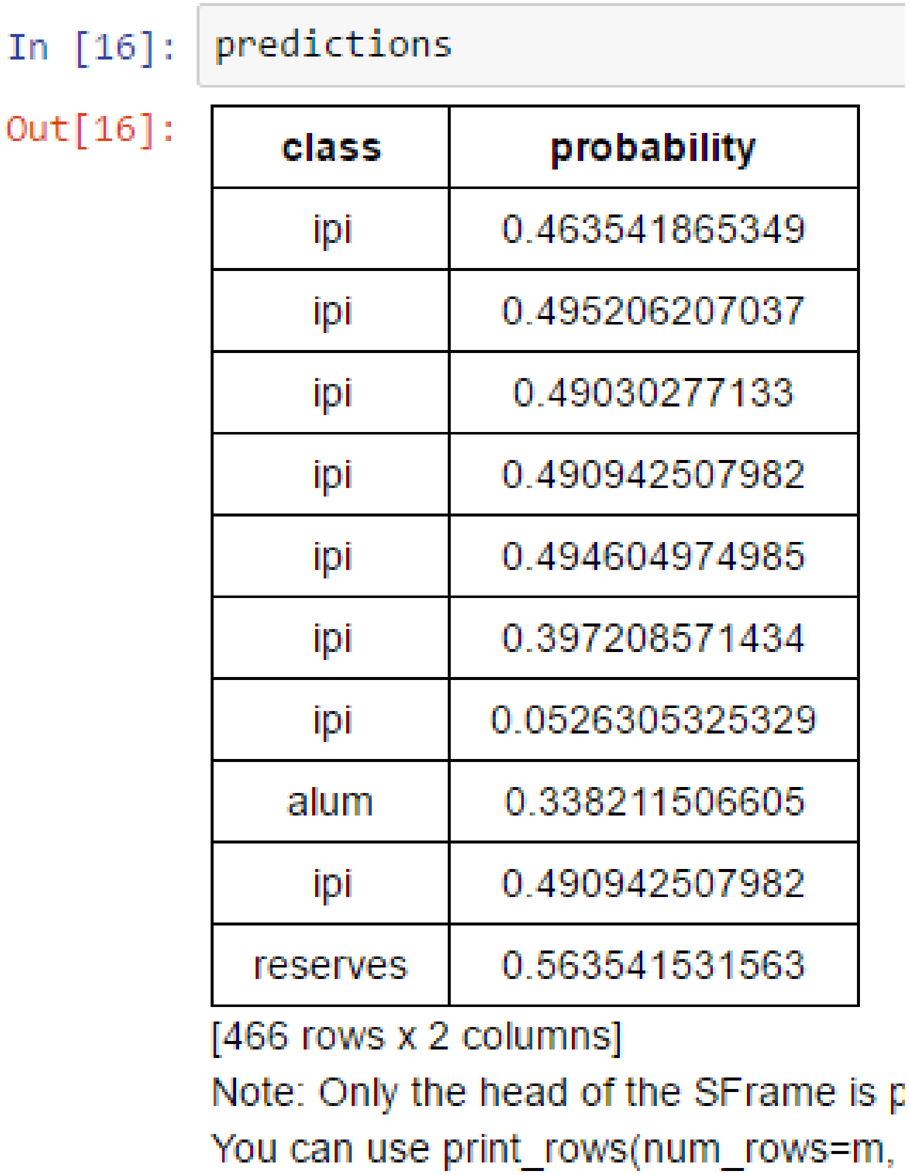
```
Boosted trees classifier:

--------------------------------------------------------

Number of examples          : 1756

Number of classes           : 34

Number of feature columns   : 1

Number of unpacked features : 17269

Create disk column page 1/1

+-----------+--------------+-------------------+-------------------+---------------------+---------------------+

| Iteration | Elapsed Time | Training-accuracy | Training-log_loss | Validation-accuracy | Validation-log_loss |

+-----------+--------------+-------------------+-------------------+---------------------+---------------------+

| 1         | 10.136764    | 0.876993          | 1.235948          | 0.575000            | 2.121445            |

| 2         | 19.866346    | 0.890091          | 0.934086          | 0.525000            | 1.927115            |

| 3         | 29.492107    | 0.899772          | 0.741076          | 0.537500            | 1.795780            |

| 4         | 39.086573    | 0.910592          | 0.603383          | 0.512500            | 1.716426            |

| 5         | 48.789179    | 0.917426          | 0.499902          | 0.512500            | 1.668092            |

| 6         | 58.514369    | 0.919134          | 0.421551          | 0.525000            | 1.626862            |

| 7         | 68.342350    | 0.921982          | 0.361099          | 0.525000            | 1.573787            |

| 8         | 78.620567    | 0.922551          | 0.314064          | 0.537500            | 1.547404            |

| 9         | 89.087766    | 0.923690          | 0.275237          | 0.537500            | 1.539655            |

| 10        | 99.270443    | 0.924260          | 0.245745          | 0.537500            | 1.526494            |

+-----------+--------------+-------------------+-------------------+---------------------+---------------------+

PROGRESS: Creating a validation set from 5 percent of training data. This may take a while.
          You can set ``validation_set=None`` to disable validation tracking.
```

- After running the test, the accuracy is 71%:
- Output:



```
In [12]: results = model.evaluate(test_data)

         External memory mode: 1 batches

In [13]: results

Out[13]: {'accuracy': 0.7103004291845494,
          'auc': 0.9698714070751959,
          'confusion_matrix': Columns:
                 target_label    str
                 predicted_label str
                 count   int

          Rows: 88

          Data:
          +----------------+-----------------+-------+
          |  target_label  | predicted_label | count |
          +----------------+-----------------+-------+
          |    pet-chem    |     pet-chem    |   3   |
          |    veg-oil     |  oilseedsoybean |   4   |
          |   graincorn    |    graincorn    |   8   |
          |      jobs      |       gnp       |   2   |
          |      jobs      |       jobs      |   5   |
          |  dlrmoney-fx   | interestmoney-fx |   1   |
          |    tradebop    |     tradebop    |   2   |
          |      gnp       |       cpi       |   2   |
          |    reserves    |       gold      |   2   |
          | money-fxdlryen |   money-fxdlr   |   6   |
          +----------------+-----------------+-------+
          [88 rows x 3 columns]
          Note: Only the head of the SFrame is printed.
          You can use print_rows(num_rows=m, num_columns=n) to print more rows and columns.,
          'f1_score': 0.6662522123798048,
          'log_loss': 1.4751669587448086,
          'precision': 0.7308917606173222,
          'recall': 0.6861378490790255,
          'roc_curve': Columns:
                 threshold       float
                 fpr     float
                 tpr     float
                 p       int
                 n       int
                 class   int

          Rows: 3400034
```

- The prediction excerpt
- Output:



```
In [16]: predictions
```

Out[16]:

class	probability
ipi	0.463541865349
ipi	0.495206207037
ipi	0.49030277133
ipi	0.490942507982
ipi	0.494604974985
ipi	0.397208571434
ipi	0.0526305325329
alum	0.338211506605
ipi	0.490942507982
reserves	0.563541531563

[466 rows x 2 columns]

Note: Only the head of the SFrame is p

You can use print_rows(num_rows=m,