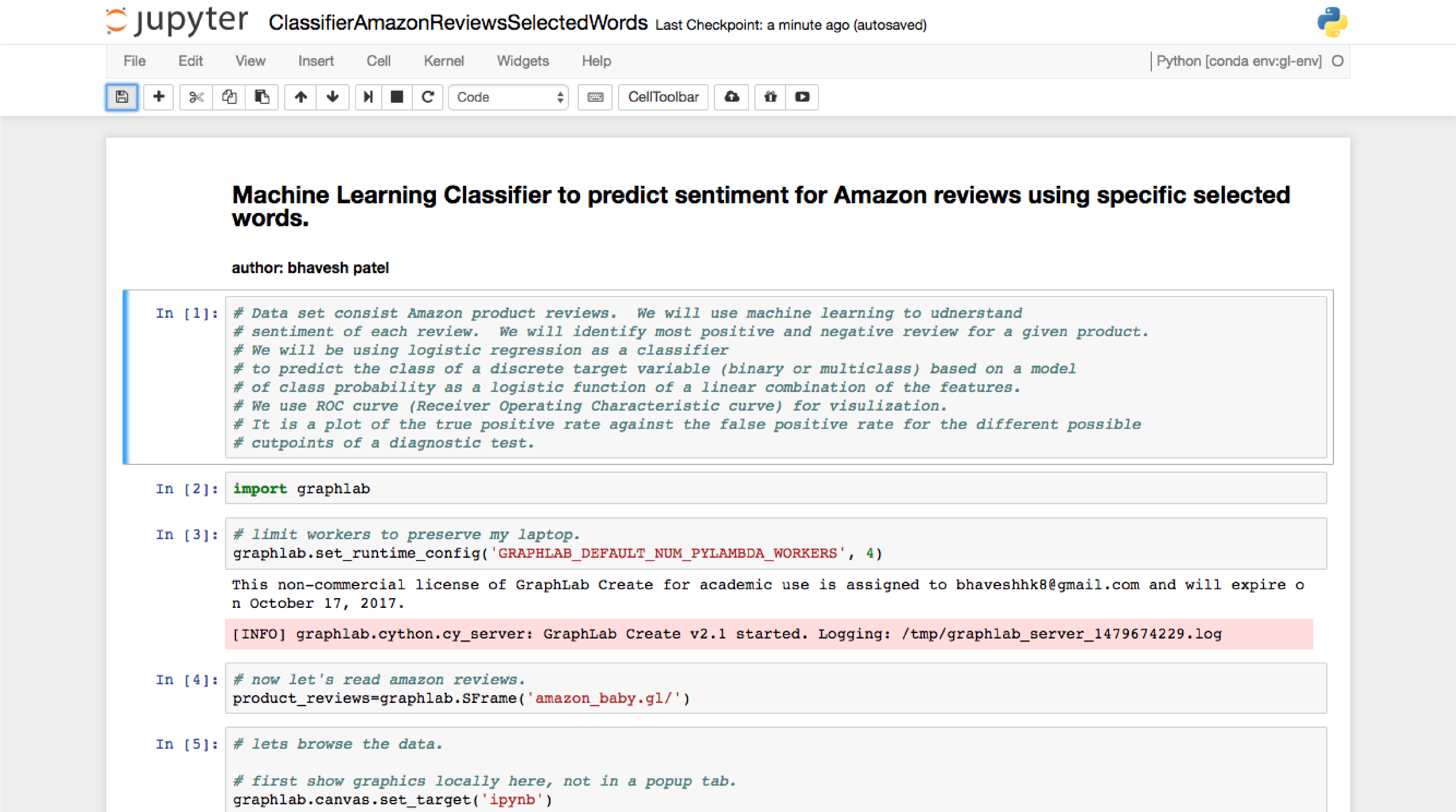

jupyter
ClassifierAmazonReviewsSelectedWords Last Checkpoint: a minute ago (autosaved)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Python [conda env:gl-env]
Code
CellToolbar

Machine Learning Classifier to predict sentiment for Amazon reviews using specific selected words.

author: bhavesh patel

In [1]:

```
# Data set consist Amazon product reviews.  We will use machine learning to udnerstand
# sentiment of each review.  We will identify most positive and negative review for a given product.
# We will be using logistic regression as a classifier
# to predict the class of a discrete target variable (binary or multiclass) based on a model
# of class probability as a logistic function of a linear combination of the features.
# We use ROC curve (Receiver Operating Characteristic curve) for visulization.
# It is a plot of the true positive rate against the false positive rate for the different possible
# cutpoints of a diagnostic test.
```

In [2]:

```
import graphlab
```

In [3]:

```
# limit workers to preserve my laptop.
graphlab.set_runtime_config('GRAPHLAB_DEFAULT_NUM_PYLAMBDA_WORKERS', 4)
```

```
This non-commercial license of GraphLab Create for academic use is assigned to bhaveshhk8@gmail.com and will expire o
n October 17, 2017.
```

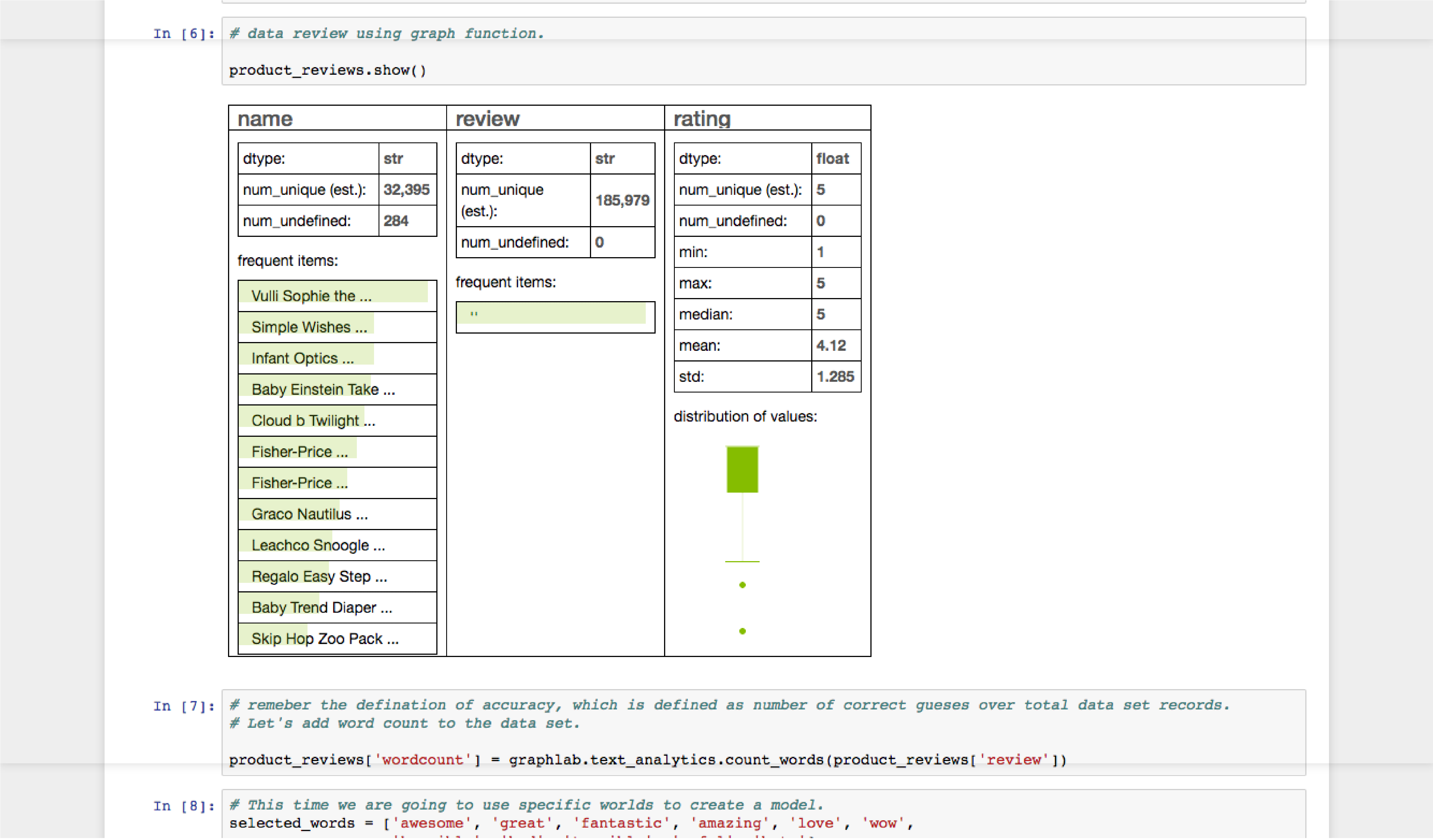
```
[INFO] graphlab.cython.cy_server: GraphLab Create v2.1 started. Logging: /tmp/graphlab_server_1479674229.log
```

In [4]:

```
# now let's read amazon reviews.
product_reviews=graphlab.SFrame('amazon_baby.gl/')
```

In [5]:

```
# lets browse the data.

# first show graphics locally here, not in a popup tab.
graphlab.canvas.set_target('ipynb')
```



In [6]:

```
# data review using graph function.

product_reviews.show()
```

name

dtype:	str
num_unique (est.):	32,395
num_undefined:	284

frequent items:

Vulli Sophie the ...
Simple Wishes ...
Infant Optics ...
Baby Einstein Take ...
Cloud b Twilight ...
Fisher-Price ...
Fisher-Price ...
Graco Nautilus ...
Leachco Snoogle ...
Regalo Easy Step ...
Baby Trend Diaper ...
Skip Hop Zoo Pack ...

review

dtype:	str
num_unique (est.):	185,979
num_undefined:	0

frequent items:

''

rating

dtype:	float
num_unique (est.):	5
num_undefined:	0
min:	1
max:	5
median:	5
mean:	4.12
std:	1.285

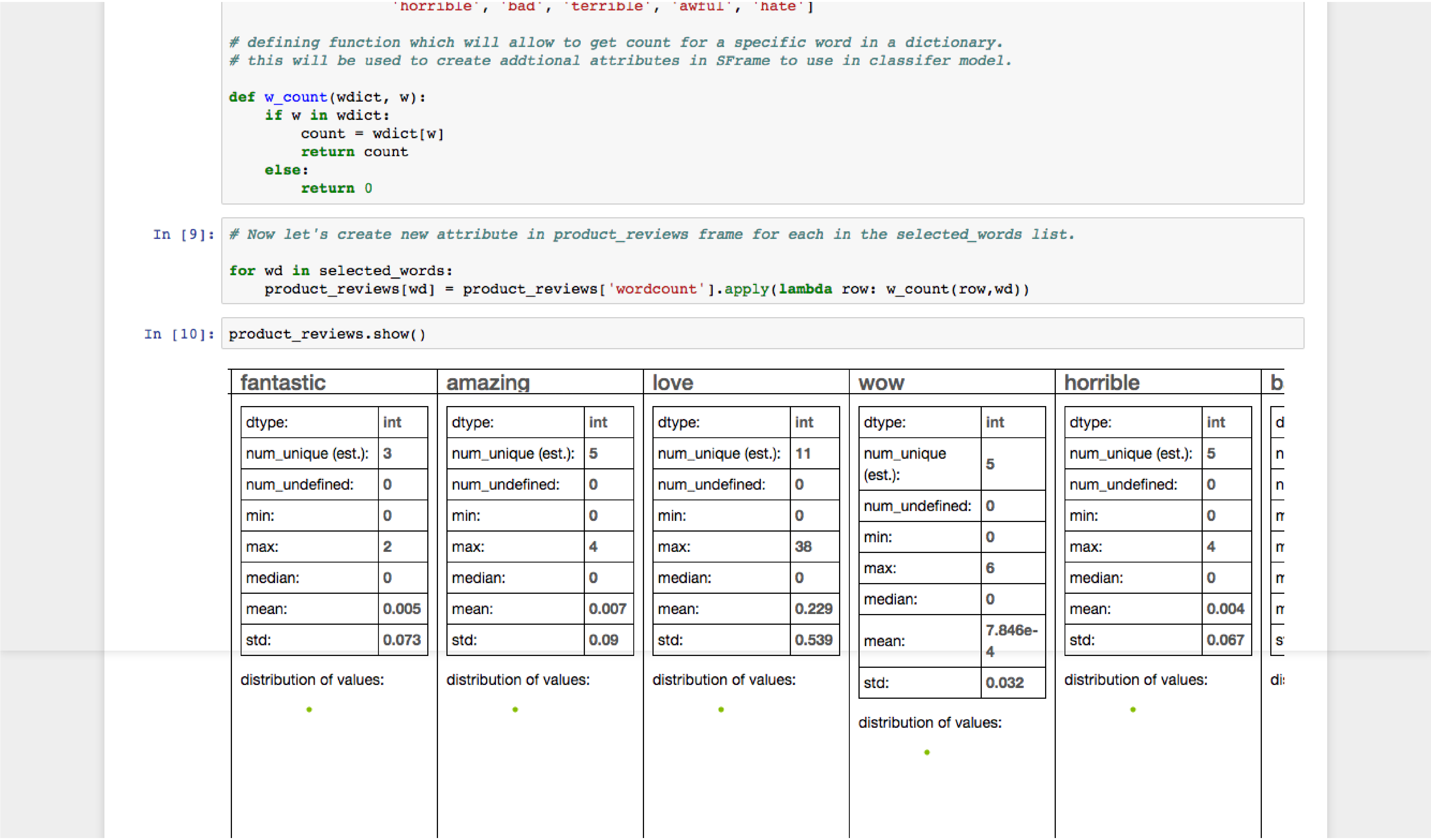
distribution of values:

In [7]:

```
# remeber the defination of accuracy, which is defined as number of correct gueses over total data set records.
# Let's add word count to the data set.

product_reviews['wordcount'] = graphlab.text_analytics.count_words(product_reviews['review'])
```

In [8]:

```
# This time we are going to use specific worlds to create a model.
selected_words = ['awesome', 'great', 'fantastic', 'amazing', 'love', 'wow',
```



```
                  'horrible', 'bad', 'terrible', 'awful', 'hate']

# defining function which will allow to get count for a specific word in a dictionary.
# this will be used to create addtional attributes in SFrame to use in classifer model.

def w_count(wdict, w):
    if w in wdict:
        count = wdict[w]
        return count
    else:
        return 0
```

In [9]:

```
# Now let's create new attribute in product_reviews frame for each in the selected_words list.

for wd in selected_words:
    product_reviews[wd] = product_reviews['wordcount'].apply(lambda row: w_count(row,wd))
```

In [10]:

```
product_reviews.show()
```

fantastic

dtype:	int
num_unique (est.):	3
num_undefined:	0
min:	0
max:	2
median:	0
mean:	0.005
std:	0.073

distribution of values:

amazing

dtype:	int
num_unique (est.):	5
num_undefined:	0
min:	0
max:	4
median:	0
mean:	0.007
std:	0.09

distribution of values:

love

dtype:	int
num_unique (est.):	11
num_undefined:	0
min:	0
max:	38
median:	0
mean:	0.229
std:	0.539

distribution of values:

wow

dtype:	int
num_unique (est.):	5
num_undefined:	0
min:	0
max:	6
median:	0
mean:	7.846e-4
std:	0.032

distribution of values:

horrible

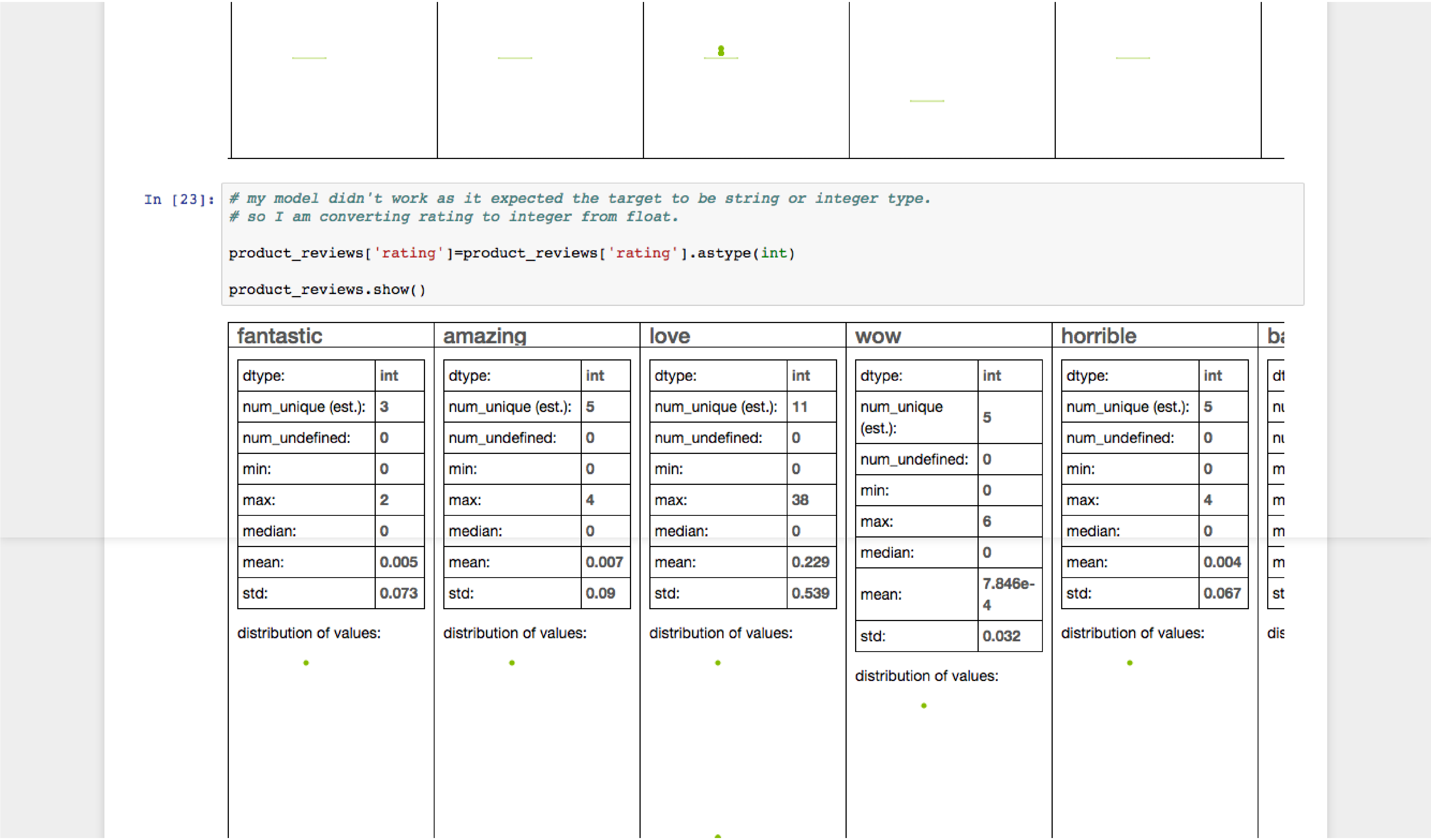
dtype:	int
num_unique (est.):	5
num_undefined:	0
min:	0
max:	4
median:	0
mean:	0.004
std:	0.067

distribution of values:

b

d
n
n
m
m
m
m
s

di



In [23]:

```
# my model didn't work as it expected the target to be string or integer type.
# so I am converting rating to integer from float.

product_reviews['rating']=product_reviews['rating'].astype(int)

product_reviews.show()
```

fantastic

dtype:	int
num_unique (est.):	3
num_undefined:	0
min:	0
max:	2
median:	0
mean:	0.005
std:	0.073

distribution of values:

amazing

dtype:	int
num_unique (est.):	5
num_undefined:	0
min:	0
max:	4
median:	0
mean:	0.007
std:	0.09

distribution of values:

love

dtype:	int
num_unique (est.):	11
num_undefined:	0
min:	0
max:	38
median:	0
mean:	0.229
std:	0.539

distribution of values:

wow

dtype:	int
num_unique (est.):	5
num_undefined:	0
min:	0
max:	6
median:	0
mean:	7.846e-4
std:	0.032

distribution of values:

horrible

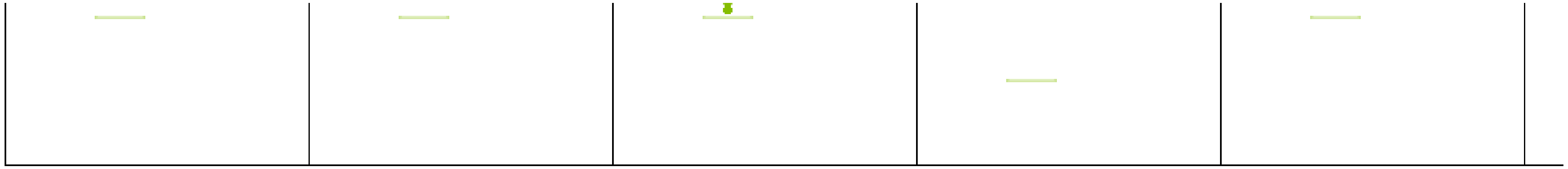
dtype:	int
num_unique (est.):	5
num_undefined:	0
min:	0
max:	4
median:	0
mean:	0.004
std:	0.067

distribution of values:

ba

dt
nu
nu
m
m
m
m
st

dis



In [47]:
```
# now we need to figure out sentiment.  That is based on rating.
# There is column rating, which has 5 values.  For now, we are going
# look into linear classifier which has binary value of 1 or 0.
# for that, we can define that any rating above 4 and 5 is positive aka 1
# any rating below 2 is negative aka 0.
# First, I don't like middle of th road rating 3, so ignore it.
product_reviews = product_reviews[product_reviews['rating'] !=3]

len(product_reviews)
```

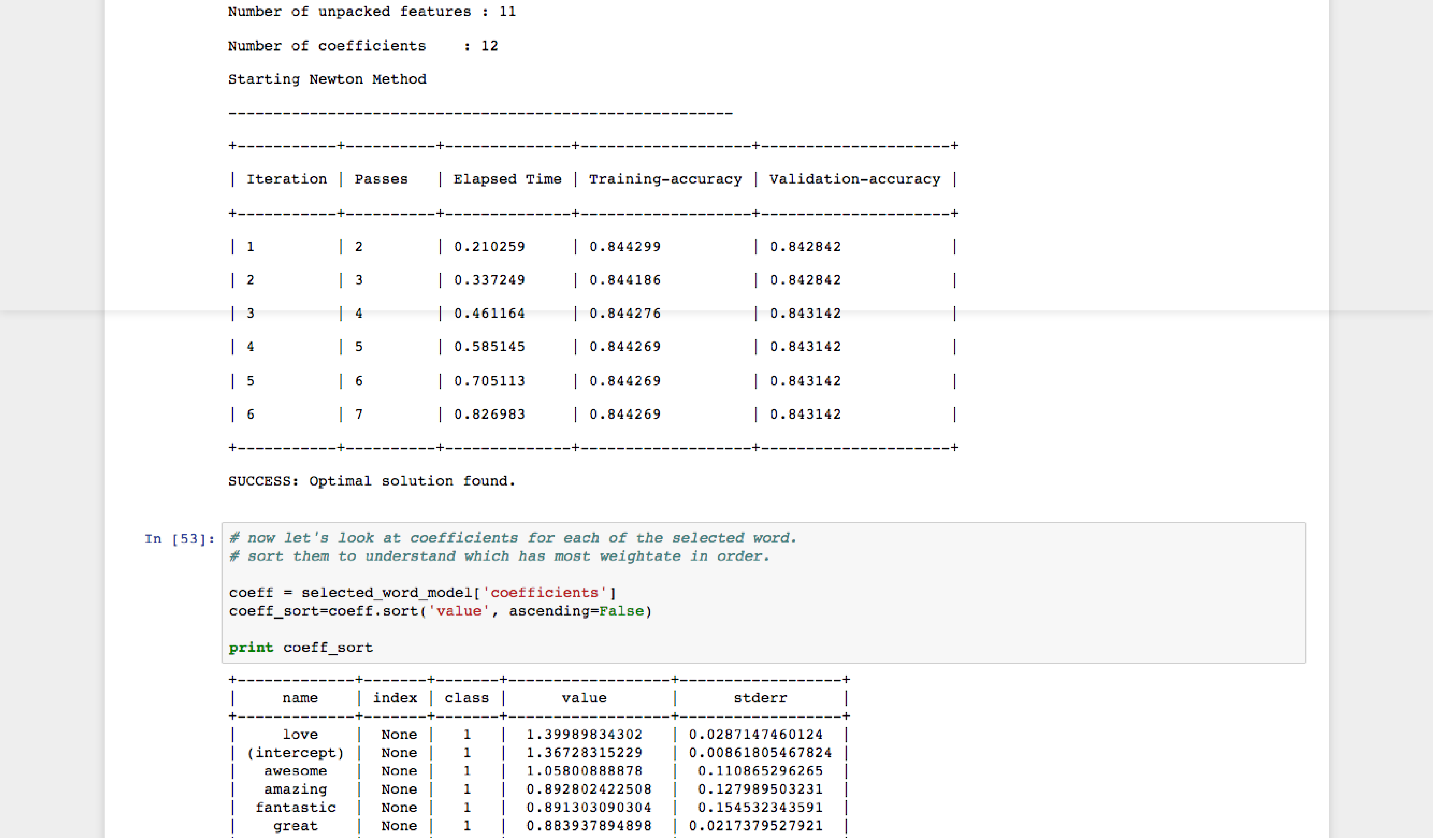
Out[47]:
```
166752
```

In [50]:
```
# now let's add directional column as we dsicussed above.

product_reviews['binrating'] = product_reviews['rating'] >= 4
```

In [51]:
```
# let's create training and test dataset.

train_data, test_data = product_reviews.random_split(0.8, seed=0)
```

In [52]:
```
# now let's create the classifer model.

selected_word_model = graphlab.logistic_classifier.create (train_data,
                                                           target='binrating',
                                                           features=selected_words,
                                                           validation_set=test_data)
```

```
Logistic regression:

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

Number of examples          : 133448

Number of classes           : 2

Number of feature columns   : 11
```

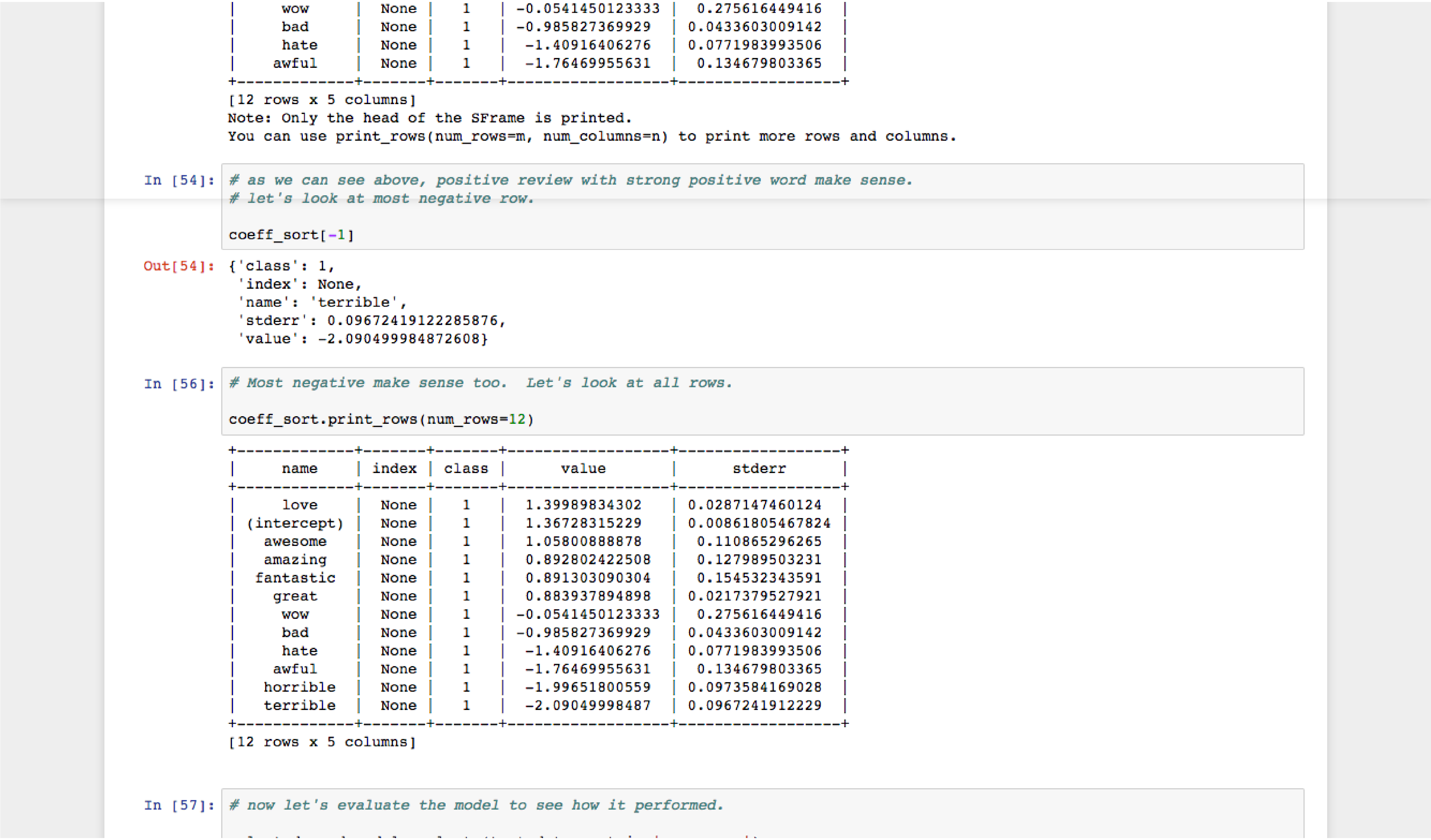


```
Number of unpacked features : 11

Number of coefficients    : 12

Starting Newton Method

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

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

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

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

| 1         | 2        | 0.210259     | 0.844299          | 0.842842            |

| 2         | 3        | 0.337249     | 0.844186          | 0.842842            |

| 3         | 4        | 0.461164     | 0.844276          | 0.843142            |

| 4         | 5        | 0.585145     | 0.844269          | 0.843142            |

| 5         | 6        | 0.705113     | 0.844269          | 0.843142            |

| 6         | 7        | 0.826983     | 0.844269          | 0.843142            |

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

SUCCESS: Optimal solution found.
```

In [53]:

```
# now let's look at coefficients for each of the selected word.
# sort them to understand which has most weightate in order.

coeff = selected_word_model['coefficients']
coeff_sort=coeff.sort('value', ascending=False)

print coeff_sort
```

```
+-------------+-------+-------+-----------------+------------------+
|     name    | index | class |      value      |      stderr      |
+-------------+-------+-------+-----------------+------------------+
|     love    |  None |   1   |  1.39989834302  | 0.0287147460124  |
| (intercept) |  None |   1   |  1.36728315229  | 0.00861805467824 |
|   awesome   |  None |   1   |  1.05800888878  |  0.110865296265  |
|   amazing   |  None |   1   |  0.892802422508 |  0.127989503231  |
|  fantastic  |  None |   1   |  0.891303090304 |  0.154532343591  |
|    great    |  None |   1   |  0.883937894898 | 0.0217379527921  |
```



```
|     wow     |  None |   1   | -0.0541450123333 |  0.275616449416  |
|     bad     |  None |   1   | -0.985827369929  | 0.0433603009142  |
|     hate    |  None |   1   |  -1.40916406276  | 0.0771983993506  |
|    awful    |  None |   1   |  -1.76469955631  |  0.134679803365  |
+-------------+-------+-------+------------------+------------------+
[12 rows x 5 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.
```

In [54]:

```
# as we can see above, positive review with strong positive word make sense.
# let's look at most negative row.

coeff_sort[-1]
```

Out[54]:

```
{'class': 1,
 'index': None,
 'name': 'terrible',
 'stderr': 0.09672419122285876,
 'value': -2.090499984872608}
```

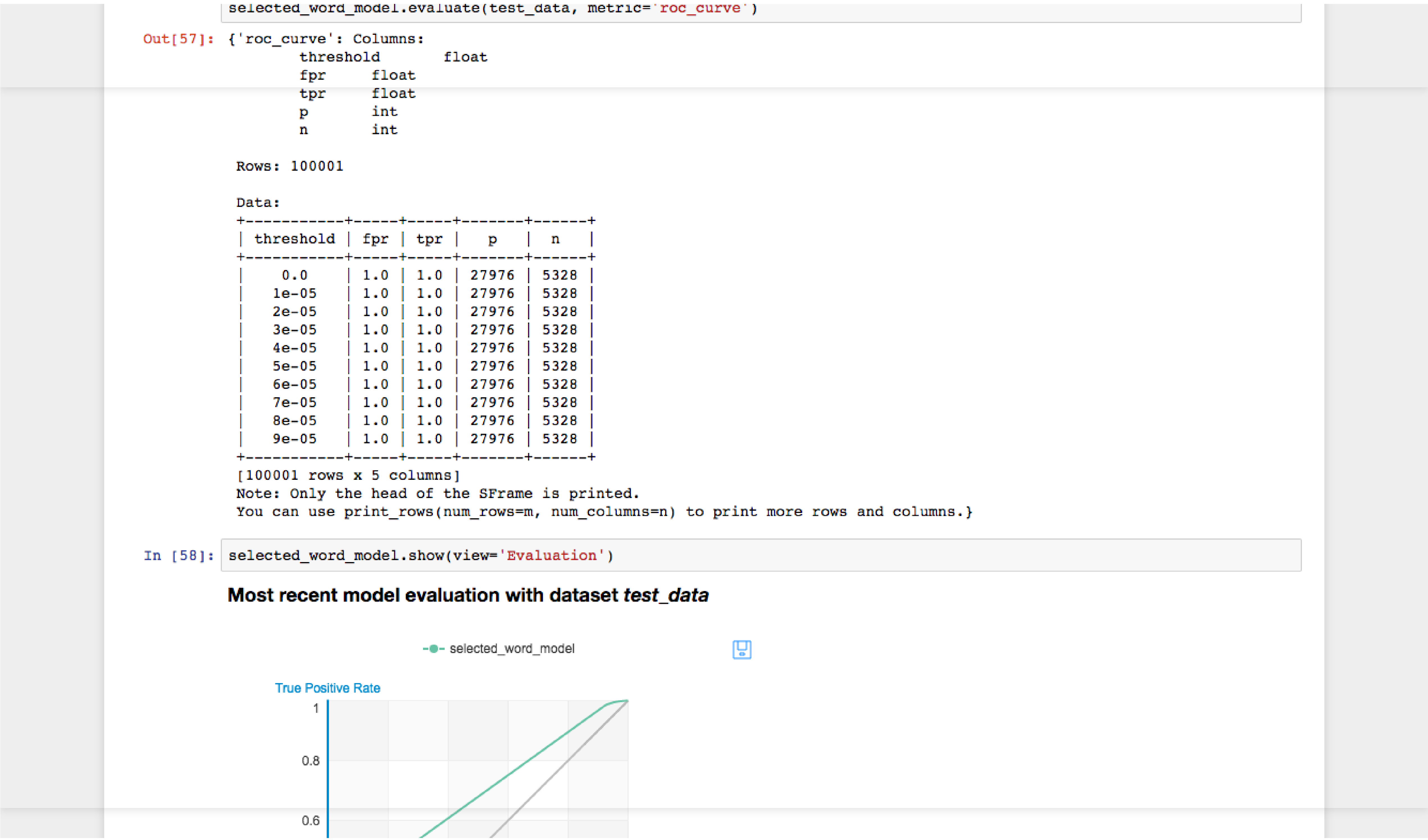
In [56]:

```
# Most negative make sense too.  Let's look at all rows.

coeff_sort.print_rows(num_rows=12)
```

```
+-------------+-------+-------+------------------+------------------+
|     name    | index | class |      value       |      stderr      |
+-------------+-------+-------+------------------+------------------+
|     love    |  None |   1   |  1.39989834302   |  0.0287147460124 |
| (intercept) |  None |   1   |  1.36728315229   | 0.00861805467824 |
|   awesome   |  None |   1   |  1.05800888878   |  0.110865296265  |
|   amazing   |  None |   1   |  0.892802422508  |  0.127989503231  |
|  fantastic  |  None |   1   |  0.891303090304  |  0.154532343591  |
|    great    |  None |   1   |  0.883937894898  | 0.0217379527921  |
|     wow     |  None |   1   | -0.0541450123333 |  0.275616449416  |
|     bad     |  None |   1   | -0.985827369929  | 0.0433603009142  |
|     hate    |  None |   1   |  -1.40916406276  | 0.0771983993506  |
|    awful    |  None |   1   |  -1.76469955631  |  0.134679803365  |
|   horrible  |  None |   1   |  -1.99651800559  | 0.0973584169028  |
|   terrible  |  None |   1   |  -2.09049998487  | 0.0967241912229  |
+-------------+-------+-------+------------------+------------------+
[12 rows x 5 columns]
```

In [57]:

```
# now let's evaluate the model to see how it performed.
```

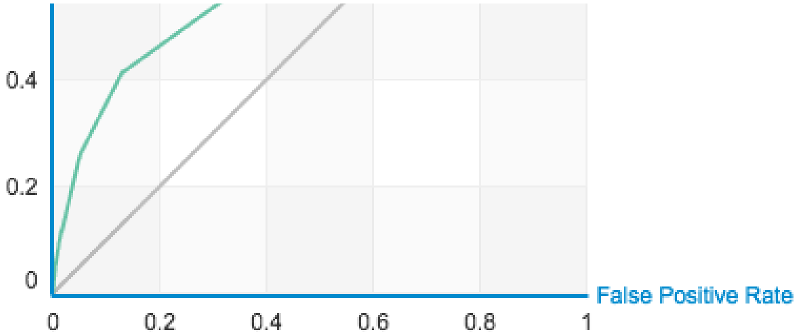


```
selected_word_model.evaluate(test_data, metric='roc_curve')
```

```
Out[57]: {'roc_curve': Columns:
                threshold       float
                fpr     float
                tpr     float
                p       int
                n       int

         Rows: 100001

         Data:
         +-----------+-----+-----+-------+------+
         | threshold | fpr | tpr |   p   |  n   |
         +-----------+-----+-----+-------+------+
         |    0.0    | 1.0 | 1.0 | 27976 | 5328 |
         |   1e-05   | 1.0 | 1.0 | 27976 | 5328 |
         |   2e-05   | 1.0 | 1.0 | 27976 | 5328 |
         |   3e-05   | 1.0 | 1.0 | 27976 | 5328 |
         |   4e-05   | 1.0 | 1.0 | 27976 | 5328 |
         |   5e-05   | 1.0 | 1.0 | 27976 | 5328 |
         |   6e-05   | 1.0 | 1.0 | 27976 | 5328 |
         |   7e-05   | 1.0 | 1.0 | 27976 | 5328 |
         |   8e-05   | 1.0 | 1.0 | 27976 | 5328 |
         |   9e-05   | 1.0 | 1.0 | 27976 | 5328 |
         +-----------+-----+-----+-------+------+
         [100001 rows x 5 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.}
```

```
In [58]: selected_word_model.show(view='Evaluation')
```

Most recent model evaluation with dataset *test_data*



True Positive	False Negative	Accuracy	Precision
27836	**140**	**0.843**	**0.845**
False Positive	True Negative	Recall	F1 Score
5094	**234**	**0.995**	**0.914**

Threshold
0.501

AUC
0.665

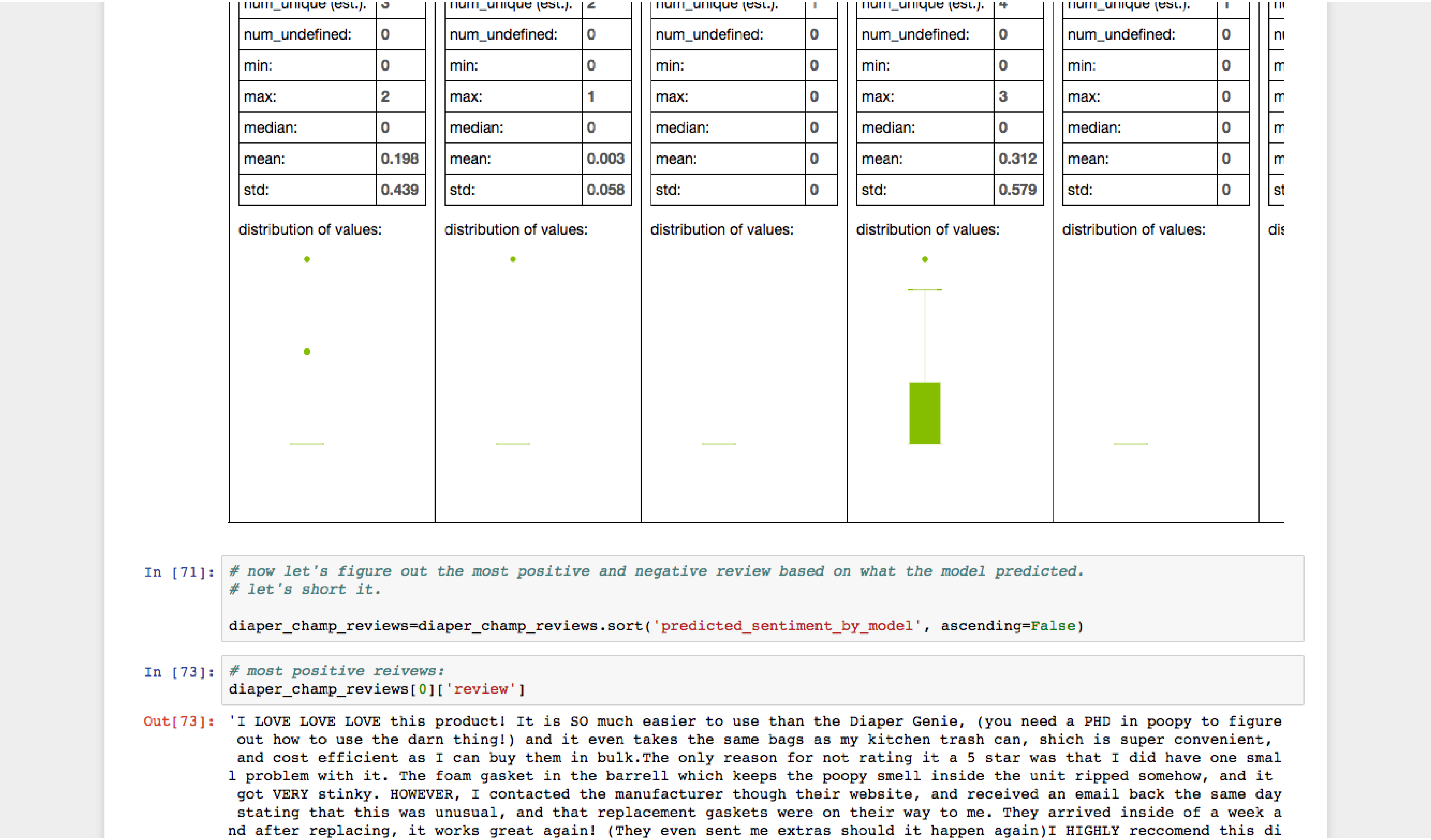
now let's use this model to find out most positive/negative reviews for a product.

```
In [62]: # get another SFrame for the product.

         diaper_champ_reviews = product_reviews[product_reviews['name']=='Baby Trend Diaper Champ']
```

```
In [63]: len(diaper_champ_reviews)

Out[63]: 298
```

```
In [69]: # now that model is ready, let's use it.
         # let's see how it predict each review sentiment for the diaper champ product.
         # we will add a column for each review. That will hold predicted sentiment by the model we built.
         diaper_champ_reviews['predicted_sentiment_by_model']=selected_word_model.predict(diaper_champ_reviews, output_type='pro

         diaper_champ_reviews.show()
```

great		fantastic		amazing		love		wow		h
dtype:	int	dtype:	int	dtype:	int	dtype:	int	dtype:	int	d



num_unique (est.):	3
num_undefined:	0
min:	0
max:	2
median:	0
mean:	0.198
std:	0.439

distribution of values:

num_unique (est.):	2
num_undefined:	0
min:	0
max:	1
median:	0
mean:	0.003
std:	0.058

distribution of values:

num_unique (est.):	1
num_undefined:	0
min:	0
max:	0
median:	0
mean:	0
std:	0

distribution of values:

num_unique (est.):	4
num_undefined:	0
min:	0
max:	3
median:	0
mean:	0.312
std:	0.579

distribution of values:

num_unique (est.):	1
num_undefined:	0
min:	0
max:	0
median:	0
mean:	0
std:	0

distribution of values:

In [71]:

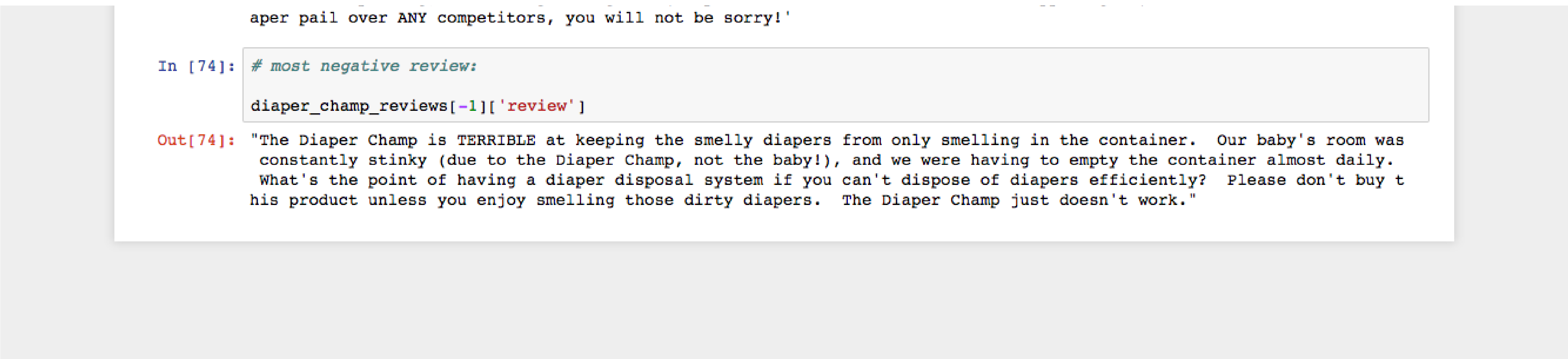
```
# now let's figure out the most positive and negative review based on what the model predicted.
# let's short it.

diaper_champ_reviews=diaper_champ_reviews.sort('predicted_sentiment_by_model', ascending=False)
```

In [73]:

```
# most positive reivews:
diaper_champ_reviews[0]['review']
```

Out[73]:

```
'I LOVE LOVE LOVE this product! It is SO much easier to use than the Diaper Genie, (you need a PHD in poopy to figure
 out how to use the darn thing!) and it even takes the same bags as my kitchen trash can, shich is super convenient,
 and cost efficient as I can buy them in bulk.The only reason for not rating it a 5 star was that I did have one smal
l problem with it. The foam gasket in the barrell which keeps the poopy smell inside the unit ripped somehow, and it
 got VERY stinky. HOWEVER, I contacted the manufacturer though their website, and received an email back the same day
 stating that this was unusual, and that replacement gaskets were on their way to me. They arrived inside of a week a
nd after replacing, it works great again! (They even sent me extras should it happen again)I HIGHLY reccomend this di
```



```
aper pail over ANY competitors, you will not be sorry!'
```

In [74]:

```
# most negative review:

diaper_champ_reviews[-1]['review']
```

Out[74]:

```
"The Diaper Champ is TERRIBLE at keeping the smelly diapers from only smelling in the container.  Our baby's room was
 constantly stinky (due to the Diaper Champ, not the baby!), and we were having to empty the container almost daily.
 What's the point of having a diaper disposal system if you can't dispose of diapers efficiently?  Please don't buy t
his product unless you enjoy smelling those dirty diapers.  The Diaper Champ just doesn't work."
```