Homework 1

W261 - Machine Learning at Scale, Section 2

- Amin Venjara
- January 18, 2016

HW1.0.0. Define big data. Provide an example of a big data problem in your domain of expertise.

There are multiple ways to define big data. One way is the 3 Vs: volume, variety, velocity.

```
- volume: that is so large it cannot fit in memory on a single compute
r. A typical laptop can store upto 1TB of data. Laptops have trouble pr
ocessing training data even when sizes are in the tens of GBs of data.
So, big data comes into play when training data sets are >~10-20GBs
- velocity: the best example here is clickstream data that comes in eve
ry second which creates a growing store of data to manage and process
- variety: data in a variety of formats, often unstructured, that needs
to be brought together to draw insight.
```

In my world of HR analytics, we pay 1 in 6 Americans comprising over 20M records (~1KB / record = 20TB) spread over a variety of systems and platforms. Users HR data is regularly updated as pay data can happen at a weekly, bi-weekly, or monthly basis. Employees are hired, change roles, leave. We recently sought to benchmark compensation for specific roles in specific geographies leveraging this data. Managing the diversity was by far the hardest as bringing together the data and finding a way to marry between different role names was very challenging.

HW1.0.1.In 500 words (English or pseudo code or a combination) describe how to estimate the bias, the variance, the irreduciable error for a test dataset T when using polynomial regression models of degree 1, 2,3, 4,5 are considered. How would you select a model?

Using a test data set with T data points

For each data point x *in the test data set compute variance over the variance predictions (50 models give 50 predictions for each data point x*)

Definitions

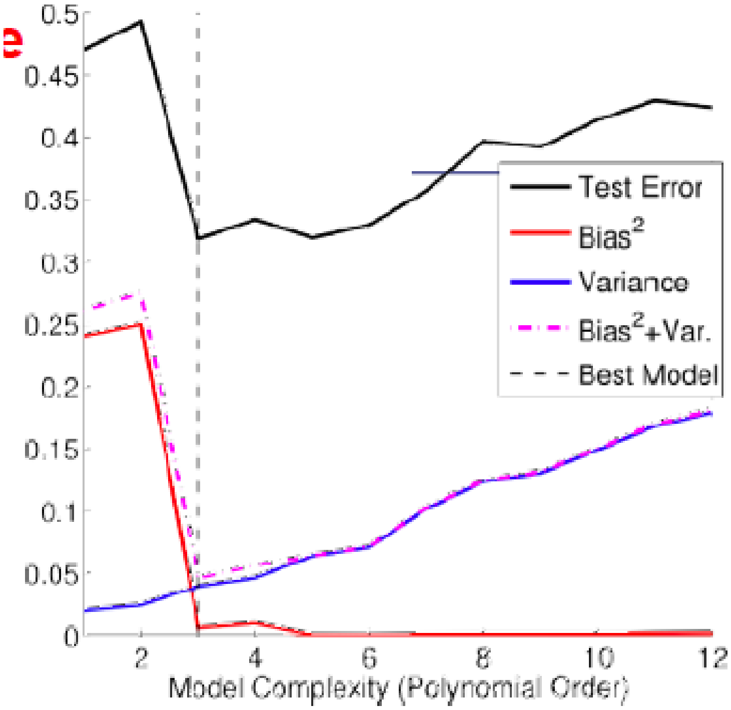
- hD(x*) = model prediction (assume 50 training datesets)
- h(x*) = Average model prediction
- f(x*) = TRUE (Actual function value)
- Y* = Observed target data (noisy)

For each data point x* calculate

- Variance = sum(h(x*) – h(x*))^2/50 ## Describes how much h(x*) varies from one training set S to another
- Bias: h(x*) – f(x*) ## Describes the average error of h(x*).
- Noise: (y – *f(x*))2 ## Describes how much y *varies from f(x*)

Compute Expected prediction error = Variance + Bias^2 + Noise^2 Find the minimum expected prediction error

- This can be done via a plot as illustrated below.



```
In [7]: !perl -pi -e 's/\r/\n/g' enronemail_1h.txt  # to unix
In [8]: !wc -l enronemail_1h.txt
```

```
      99 enronemail_1h.txt
```

HW1.1. Read through the provided control script (pNaiveBayes.sh) and all of its comments. When you are comfortable with their purpose and function, respond to the remaining homework questions below. A simple cell in the notebook with a print statmement with a "done" string will suffice here. (dont forget to include the Question Number and the quesition in the cell as a multiline comment!)

```
In [12]: print 'done'
```

```
done
```

HW1.2. Provide a mapper/reducer pair that, when executed by pNaiveBayes.sh will determine the number of occurrences of a single, user-specified word. Examine the word “assistance” and report your results.

```
In [75]: %%writefile mapper.py
#!/usr/bin/python
## mapper.py
## Author: Amin Venjara
## Description: mapper code for HW1.2

import sys
import re
import string
count = 0
from collections import Counter

## collect user input
filename = sys.argv[1]
findwords = re.split(" ",sys.argv[2].lower()) # will only be single word
c = Counter()
with open (filename, "r") as myfile:
    for line in myfile:
        c.update(line.translate(None, string.punctuation).split())
    for word in findwords:
        print '%s\t%s'% (word, c[word])
```

```
Overwriting mapper.py
```

In [72]:

```
%%writefile reducer.py
#!/usr/bin/python
## reducer.py
## Author: Amin Venjara
## Description: reducer code for HW1.2

import sys
import re
sum = 0

# capture the list of mapped files as a list
mapped_files = sys.argv[1:]

# stores cumulative count of words across mapped files
word_count = {}

# input comes from mapper.py
for f in mapped_files:
    with open (f, "r") as myfile:
        for line in myfile:
            # remove leading and trailing whitespace
            line = line.strip()

            # parse the input we got from mapper.py
            word, count = line.split('\t', 1)

            # convert count (currently a string) to int
            try:
                count = int(count)
            except ValueError:
                continue

            try:
                word_count[word] = word_count[word]+count
            except:
                word_count[word] = count

# write the tuples to stdout
# Note: they are unsorted
for word in word_count.keys():
    print '%s\t%s'% ( word, word_count[word] )
```

```
Overwriting reducer.py
```

In [76]:

```
!chmod +x mapper.py; chmod +x reducer.py; chmod +x pNaiveBayes.sh;
```

In [81]:

```
!./pNaiveBayes.sh 4 "assistance"
!head enronemail_1h.txt.output
```

```
assistance      10
```

HW1.3. Provide a mapper/reducer pair that, when executed by pNaiveBayes.sh will classify the email messages by a single, user-specified word using the multinomial Naive Bayes Formulation. Examine the word “assistance” and report your results. To do so, make sure that mapper.py and reducer.py that performs a single word Naive Bayes classification.

For multinomial Naive Bayes, the Pr(X=“assistance”|Y=SPAM) is calculated as follows: the number of times “assistance” occurs in SPAM labeled documents / the number of words in documents labeled SPAM NOTE: if “assistance” occurs 5 times in all of the documents Labeled SPAM, and the length in terms of the number of words in all documents labeld as SPAM (when concatenated) is 1,000. Then Pr(X=“assistance”|Y=SPAM) = 5/1000. Note this is a multinomial estimated of the class conditional for a Naive Bayes Classifier. No smoothing is needed in this HW.**

Multinomial NB works as follows for a single word:

P(spam|"assistance") = prior_spam x P("assistance"|spam) x count_of_word

So need to compute:

- prior_spam = # of spam emails / total number of emails
- P ("assistance"| SPAM) = # of times “assistance” occurs in SPAM labeled documents / the number of words in documents labeled SPAM
- count_of_word is the number of time the given word appears in the email

Alternatively:

- P(NOT spam|word in email) = prior_NOTspam x P(word in email |NOT spam) x count_of_word**

In [148]:

```
%%writefile mapper.py
#!/usr/bin/python
## mapper.py
## Author: Amin Venjara
## Description: mapper code for HW1.3

import sys
import re
from collections import Counter
import string
count = 0

## collect user input
filename = sys.argv[1]
findwords = re.split(" ",sys.argv[2].lower()) # will only be single word
c = Counter()

with open (filename, "r") as myfile:
    for line in myfile:
        email = line.translate(None, string.punctuation).split()
        c = Counter(email[2:])
        (email_id, SPAM_flag, target_word_count, email_word_count) = (line.split()[0], int(email[1]), c[findwords[0]], sum(c.values()))
        print '%s\t%d\t%d\t%d'% (email_id, SPAM_flag, target_word_count, email_word_count)
```

```
Overwriting mapper.py
```

In [168]:

```
%%writefile reducer.py
#!/Applications/anaconda/bin/python
## reducer.py
## Author: Amin Venjara
## Description: reducer code for HW1.3

import sys
import re
import pandas as pd
pd.options.mode.chained_assignment = None
sum = 0

# capture the list of mapped files as a list
mapped_files = sys.argv[1:]

####################
# gather the data  #
####################

# stores email data across mapped files
col_names = ['email_id', 'SPAM_flag', 'target_word_count', 'email_w
ord_count']
email_parser = pd.DataFrame()

# create dataframe that combines data from mapper.py
for f in mapped_files:
    with open (f, "r") as myfile:
        email_parser = email_parser.append(pd.read_csv(myfile, se
p='\t', names=col_names), ignore_index = True)

#print email_parser

####################
# process the data #
####################

# create a df that select just SPAM emails
df_SPAM_emails = email_parser[email_parser.SPAM_flag == 1]

total_emails = len(email_parser.index)
print "total_emails = ", total_emails

SPAM_emails = len(df_SPAM_emails.index)
print "SPAM_emails = ", SPAM_emails

target_word_count_in_SPAM_emails = df_SPAM_emails['target_word_coun
t'].sum()
print "target_word_count_in_SPAM_emails = ", target_word_count_in_S
PAM_emails

SPAM_word_count = df_SPAM_emails['email_word_count'].sum()
print "SPAM_word_count = ", SPAM_word_count
```

```
#compute the elements of the bayesian calculation
prior_spam = float(SPAM_emails)/float(total_emails)
print prior_spam

prob_targetword_given_SPAM = float(target_word_count_in_SPAM_emails) / float(SPAM_word_count)
print prob_targetword_given_SPAM

output = email_parser[['email_id', 'SPAM_flag']]
output['classification'] = (prior_spam*prob_targetword_given_SPAM*email_parser['target_word_count']).round().astype(int)

####################
# output the data  #
####################
print output.to_csv(sep='\t', header = False, index = False)
```

```
Overwriting reducer.py
```

In [169]:

```
!chmod +x mapper.py; chmod +x reducer.py; chmod +x pNaiveBayes.sh;
```

In [171]:

```
!./pNaiveBayes.sh 4 "assistance"
!head -100 enronemail_1h.txt.output
```

```
total_emails =  100
SPAM_emails =  44
target_word_count_in_SPAM_emails =  8
SPAM_word_count =  18283
0.44
0.000437564951047
0001.1999-12-10.farmer  0       0
0001.1999-12-10.kaminski        0       0
0001.2000-01-17.beck    0       0
0001.2000-06-06.lokay   0       0
0001.2001-02-07.kitchen 0       0
0001.2001-04-02.williams        0       0
0002.1999-12-13.farmer  0       0
0002.2001-02-07.kitchen 0       0
0002.2001-05-25.SA_and_HP       1       0
0002.2003-12-18.GP      1       0
0002.2004-08-01.BG      1       0
0003.1999-12-10.kaminski        0       0
0003.1999-12-14.farmer  0       0
0003.2000-01-17.beck    0       0
0003.2001-02-08.kitchen 0       0
0003.2003-12-18.GP      1       0
0003.2004-08-01.BG      1       0
0004.1999-12-10.kaminski        0       0
0004.1999-12-14.farmer  0       0
0004.2001-04-02.williams        0       0
0004.2001-06-12.SA_and_HP       1       0
0004.2004-08-01.BG      1       0
0005.1999-12-12.kaminski        0       0
0005.1999-12-14.farmer  0       0
0005.2000-06-06.lokay   0       0
0005.2001-02-08.kitchen 0       0
0005.2001-06-23.SA_and_HP       1       0
0005.2003-12-18.GP      1       0
0006.1999-12-13.kaminski        0       0
0006.2001-02-08.kitchen 0       0
0006.2001-04-03.williams        0       0
0006.2001-06-25.SA_and_HP       1       0
0006.2003-12-18.GP      1       0
0006.2004-08-01.BG      1       0
0007.1999-12-13.kaminski        0       0
0007.1999-12-14.farmer  0       0
0007.2000-01-17.beck    0       0
0007.2001-02-09.kitchen 0       0
0007.2003-12-18.GP      1       0
0007.2004-08-01.BG      1       0
0008.2001-02-09.kitchen 0       0
0008.2001-06-12.SA_and_HP       1       0
0008.2001-06-25.SA_and_HP       1       0
0008.2003-12-18.GP      1       0
0008.2004-08-01.BG      1       0
0009.1999-12-13.kaminski        0       0
0009.1999-12-14.farmer  0       0
```

```
0009.2000-06-07.lokay   0       0
0009.2001-02-09.kitchen 0       0
0009.2001-06-26.SA_and_HP       1       0
0009.2003-12-18.GP      1       0
0010.1999-12-14.farmer  0       0
0010.1999-12-14.kaminski        0       0
0010.2001-02-09.kitchen 0       0
0010.2001-06-28.SA_and_HP       1       0
0010.2003-12-18.GP      1       0
0010.2004-08-01.BG      1       0
0011.1999-12-14.farmer  0       0
0011.2001-06-28.SA_and_HP       1       0
0011.2001-06-29.SA_and_HP       1       0
0011.2003-12-18.GP      1       0
0011.2004-08-01.BG      1       0
0012.1999-12-14.farmer  0       0
0012.1999-12-14.kaminski        0       0
0012.2000-01-17.beck    0       0
0012.2000-06-08.lokay   0       0
0012.2001-02-09.kitchen 0       0
0012.2003-12-19.GP      1       0
0013.1999-12-14.farmer  0       0
0013.1999-12-14.kaminski        0       0
0013.2001-04-03.williams        0       0
0013.2001-06-30.SA_and_HP       1       0
0013.2004-08-01.BG      1       0
0014.1999-12-14.kaminski        0       0
0014.1999-12-15.farmer  0       0
0014.2001-02-12.kitchen 0       0
0014.2001-07-04.SA_and_HP       1       0
0014.2003-12-19.GP      1       0
0014.2004-08-01.BG      1       0
0015.1999-12-14.kaminski        0       0
0015.1999-12-15.farmer  0       0
0015.2000-06-09.lokay   0       0
0015.2001-02-12.kitchen 0       0
0015.2001-07-05.SA_and_HP       1       0
0015.2003-12-19.GP      1       0
0016.1999-12-15.farmer  0       0
0016.2001-02-12.kitchen 0       0
0016.2001-07-05.SA_and_HP       1       0
0016.2001-07-06.SA_and_HP       1       0
0016.2003-12-19.GP      1       0
0016.2004-08-01.BG      1       0
0017.1999-12-14.kaminski        0       0
0017.2000-01-17.beck    0       0
0017.2001-04-03.williams        0       0
```

HW1.4. Provide a mapper/reducer pair that, when executed by pNaiveBayes.sh will classify the email messages by a list of one or more user-specified words. Examine the words “assistance”, “valium”, and “enlargementWithATypo” and report your results. To do so, make sure that

- mapper.py counts all occurrences of a list of words, and
- reducer.py

performs the multiple-word multinomial Naive Bayes classification via the chosen list. No smoothing is needed in this HW.

In [175]:

```
%%writefile mapper.py
#!/usr/bin/python
## mapper.py
## Author: Amin Venjara
## Description: mapper code for HW1.4

import sys
import re
from collections import Counter
import string
count = 0

## collect user input
filename = sys.argv[1]
findwords = re.split(" ",sys.argv[2].lower()) # will only be single
word
c = Counter()

with open (filename, "r") as myfile:
    for line in myfile:
        email = line.translate(None, string.punctuation).split()
        c = Counter(email[2:])
        (email_id, SPAM_flag, target_word_count, email_word_count)
= (line.split()[0], int(email[1]), c[findwords[0]], sum(c.value
s()))
        print '%s\t%d\t%d\t%d'% (email_id, SPAM_flag, target_word_c
ount, email_word_count)
```

```
Overwriting mapper.py
```

In [176]:

```
%%writefile reducer.py
#!/Applications/anaconda/bin/python
## reducer.py
## Author: Amin Venjara
## Description: reducer code for HW1.4

import sys
import re
import pandas as pd
pd.options.mode.chained_assignment = None
sum = 0

# capture the list of mapped files as a list
mapped_files = sys.argv[1:]

####################
# gather the data  #
####################

# stores email data across mapped files
col_names = ['email_id', 'SPAM_flag', 'target_word_count', 'email_w
ord_count']
email_parser = pd.DataFrame()

# create dataframe that combines data from mapper.py
for f in mapped_files:
    with open (f, "r") as myfile:
        email_parser = email_parser.append(pd.read_csv(myfile, se
p='\t', names=col_names), ignore_index = True)

#print email_parser

####################
# process the data #
####################

# create a df that select just SPAM emails
df_SPAM_emails = email_parser[email_parser.SPAM_flag == 1]

total_emails = len(email_parser.index)
print "total_emails = ", total_emails

SPAM_emails = len(df_SPAM_emails.index)
print "SPAM_emails = ", SPAM_emails

target_word_count_in_SPAM_emails = df_SPAM_emails['target_word_coun
t'].sum()
print "target_word_count_in_SPAM_emails = ", target_word_count_in_S
PAM_emails

SPAM_word_count = df_SPAM_emails['email_word_count'].sum()
print "SPAM_word_count = ", SPAM_word_count
```

```
#compute the elements of the bayesian calculation
prior_spam = float(SPAM_emails)/float(total_emails)
print prior_spam

prob_targetword_given_SPAM = float(target_word_count_in_SPAM_email
s) / float(SPAM_word_count)
print prob_targetword_given_SPAM

output = email_parser[['email_id', 'SPAM_flag']]
output['classification'] = (prior_spam*prob_targetword_given_SPAM*e
mail_parser['target_word_count']).round().astype(int)

####################
# output the data  #
####################
print output.to_csv(sep='\t', header = False, index = False)
```

```
Overwriting reducer.py
```

In [177]:

```
!chmod +x mapper.py; chmod +x reducer.py; chmod +x pNaiveBayes.sh;
```

In [178]:
```
!./pNaiveBayes.sh 4 "enlargementWithATypo"
!head -100 enronemail_1h.txt.output
```

```
total_emails =  100
SPAM_emails =  44
target_word_count_in_SPAM_emails =  0
SPAM_word_count =  18283
0.44
0.0
0001.1999-12-10.farmer  0       0
0001.1999-12-10.kaminski        0       0
0001.2000-01-17.beck    0       0
0001.2000-06-06.lokay   0       0
0001.2001-02-07.kitchen 0       0
0001.2001-04-02.williams        0       0
0002.1999-12-13.farmer  0       0
0002.2001-02-07.kitchen 0       0
0002.2001-05-25.SA_and_HP       1       0
0002.2003-12-18.GP      1       0
0002.2004-08-01.BG      1       0
0003.1999-12-10.kaminski        0       0
0003.1999-12-14.farmer  0       0
0003.2000-01-17.beck    0       0
0003.2001-02-08.kitchen 0       0
0003.2003-12-18.GP      1       0
0003.2004-08-01.BG      1       0
0004.1999-12-10.kaminski        0       0
0004.1999-12-14.farmer  0       0
0004.2001-04-02.williams        0       0
0004.2001-06-12.SA_and_HP       1       0
0004.2004-08-01.BG      1       0
0005.1999-12-12.kaminski        0       0
0005.1999-12-14.farmer  0       0
0005.2000-06-06.lokay   0       0
0005.2001-02-08.kitchen 0       0
0005.2001-06-23.SA_and_HP       1       0
0005.2003-12-18.GP      1       0
0006.1999-12-13.kaminski        0       0
0006.2001-02-08.kitchen 0       0
0006.2001-04-03.williams        0       0
0006.2001-06-25.SA_and_HP       1       0
0006.2003-12-18.GP      1       0
0006.2004-08-01.BG      1       0
0007.1999-12-13.kaminski        0       0
0007.1999-12-14.farmer  0       0
0007.2000-01-17.beck    0       0
0007.2001-02-09.kitchen 0       0
0007.2003-12-18.GP      1       0
0007.2004-08-01.BG      1       0
0008.2001-02-09.kitchen 0       0
0008.2001-06-12.SA_and_HP       1       0
0008.2001-06-25.SA_and_HP       1       0
0008.2003-12-18.GP      1       0
0008.2004-08-01.BG      1       0
0009.1999-12-13.kaminski        0       0
0009.1999-12-14.farmer  0       0
```

```
0009.2000-06-07.lokay   0       0
0009.2001-02-09.kitchen 0       0
0009.2001-06-26.SA_and_HP       1       0
0009.2003-12-18.GP      1       0
0010.1999-12-14.farmer  0       0
0010.1999-12-14.kaminski        0       0
0010.2001-02-09.kitchen 0       0
0010.2001-06-28.SA_and_HP       1       0
0010.2003-12-18.GP      1       0
0010.2004-08-01.BG      1       0
0011.1999-12-14.farmer  0       0
0011.2001-06-28.SA_and_HP       1       0
0011.2001-06-29.SA_and_HP       1       0
0011.2003-12-18.GP      1       0
0011.2004-08-01.BG      1       0
0012.1999-12-14.farmer  0       0
0012.1999-12-14.kaminski        0       0
0012.2000-01-17.beck    0       0
0012.2000-06-08.lokay   0       0
0012.2001-02-09.kitchen 0       0
0012.2003-12-19.GP      1       0
0013.1999-12-14.farmer  0       0
0013.1999-12-14.kaminski        0       0
0013.2001-04-03.williams        0       0
0013.2001-06-30.SA_and_HP       1       0
0013.2004-08-01.BG      1       0
0014.1999-12-14.kaminski        0       0
0014.1999-12-15.farmer  0       0
0014.2001-02-12.kitchen 0       0
0014.2001-07-04.SA_and_HP       1       0
0014.2003-12-19.GP      1       0
0014.2004-08-01.BG      1       0
0015.1999-12-14.kaminski        0       0
0015.1999-12-15.farmer  0       0
0015.2000-06-09.lokay   0       0
0015.2001-02-12.kitchen 0       0
0015.2001-07-05.SA_and_HP       1       0
0015.2003-12-19.GP      1       0
0016.1999-12-15.farmer  0       0
0016.2001-02-12.kitchen 0       0
0016.2001-07-05.SA_and_HP       1       0
0016.2001-07-06.SA_and_HP       1       0
0016.2003-12-19.GP      1       0
0016.2004-08-01.BG      1       0
0017.1999-12-14.kaminski        0       0
0017.2000-01-17.beck    0       0
0017.2001-04-03.williams        0       0
```

In [179]:

```
!./pNaiveBayes.sh 4 "valium"
!head -100 enronemail_1h.txt.output
```

```
total_emails =  100
SPAM_emails =  44
target_word_count_in_SPAM_emails =  3
SPAM_word_count =  18283
0.44
0.000164086856643
0001.1999-12-10.farmer  0       0
0001.1999-12-10.kaminski        0       0
0001.2000-01-17.beck    0       0
0001.2000-06-06.lokay   0       0
0001.2001-02-07.kitchen 0       0
0001.2001-04-02.williams        0       0
0002.1999-12-13.farmer  0       0
0002.2001-02-07.kitchen 0       0
0002.2001-05-25.SA_and_HP       1       0
0002.2003-12-18.GP      1       0
0002.2004-08-01.BG      1       0
0003.1999-12-10.kaminski        0       0
0003.1999-12-14.farmer  0       0
0003.2000-01-17.beck    0       0
0003.2001-02-08.kitchen 0       0
0003.2003-12-18.GP      1       0
0003.2004-08-01.BG      1       0
0004.1999-12-10.kaminski        0       0
0004.1999-12-14.farmer  0       0
0004.2001-04-02.williams        0       0
0004.2001-06-12.SA_and_HP       1       0
0004.2004-08-01.BG      1       0
0005.1999-12-12.kaminski        0       0
0005.1999-12-14.farmer  0       0
0005.2000-06-06.lokay   0       0
0005.2001-02-08.kitchen 0       0
0005.2001-06-23.SA_and_HP       1       0
0005.2003-12-18.GP      1       0
0006.1999-12-13.kaminski        0       0
0006.2001-02-08.kitchen 0       0
0006.2001-04-03.williams        0       0
0006.2001-06-25.SA_and_HP       1       0
0006.2003-12-18.GP      1       0
0006.2004-08-01.BG      1       0
0007.1999-12-13.kaminski        0       0
0007.1999-12-14.farmer  0       0
0007.2000-01-17.beck    0       0
0007.2001-02-09.kitchen 0       0
0007.2003-12-18.GP      1       0
0007.2004-08-01.BG      1       0
0008.2001-02-09.kitchen 0       0
0008.2001-06-12.SA_and_HP       1       0
0008.2001-06-25.SA_and_HP       1       0
0008.2003-12-18.GP      1       0
0008.2004-08-01.BG      1       0
0009.1999-12-13.kaminski        0       0
0009.1999-12-14.farmer  0       0
```

```
0009.2000-06-07.lokay   0       0
0009.2001-02-09.kitchen 0       0
0009.2001-06-26.SA_and_HP       1       0
0009.2003-12-18.GP      1       0
0010.1999-12-14.farmer  0       0
0010.1999-12-14.kaminski        0       0
0010.2001-02-09.kitchen 0       0
0010.2001-06-28.SA_and_HP       1       0
0010.2003-12-18.GP      1       0
0010.2004-08-01.BG      1       0
0011.1999-12-14.farmer  0       0
0011.2001-06-28.SA_and_HP       1       0
0011.2001-06-29.SA_and_HP       1       0
0011.2003-12-18.GP      1       0
0011.2004-08-01.BG      1       0
0012.1999-12-14.farmer  0       0
0012.1999-12-14.kaminski        0       0
0012.2000-01-17.beck    0       0
0012.2000-06-08.lokay   0       0
0012.2001-02-09.kitchen 0       0
0012.2003-12-19.GP      1       0
0013.1999-12-14.farmer  0       0
0013.1999-12-14.kaminski        0       0
0013.2001-04-03.williams        0       0
0013.2001-06-30.SA_and_HP       1       0
0013.2004-08-01.BG      1       0
0014.1999-12-14.kaminski        0       0
0014.1999-12-15.farmer  0       0
0014.2001-02-12.kitchen 0       0
0014.2001-07-04.SA_and_HP       1       0
0014.2003-12-19.GP      1       0
0014.2004-08-01.BG      1       0
0015.1999-12-14.kaminski        0       0
0015.1999-12-15.farmer  0       0
0015.2000-06-09.lokay   0       0
0015.2001-02-12.kitchen 0       0
0015.2001-07-05.SA_and_HP       1       0
0015.2003-12-19.GP      1       0
0016.1999-12-15.farmer  0       0
0016.2001-02-12.kitchen 0       0
0016.2001-07-05.SA_and_HP       1       0
0016.2001-07-06.SA_and_HP       1       0
0016.2003-12-19.GP      1       0
0016.2004-08-01.BG      1       0
0017.1999-12-14.kaminski        0       0
0017.2000-01-17.beck    0       0
0017.2001-04-03.williams        0       0
```

In [ ]: