further analysis

May 16, 2016

```
In [1]: %matplotlib inline

        import pandas as pd
        import matplotlib.pyplot as plt
        import numpy as np

        # Make the graphs a bit prettier, and bigger
        pd.set_option('display.mpl_style', 'default')
        plt.rcParams['figure.figsize'] = (15, 5)
        plt.rcParams['font.family'] = 'sans-serif'

        # This is necessary to show lots of columns in pandas 0.12.
        # Not necessary in pandas 0.13.
        pd.set_option('display.width', 5000)
        pd.set_option('display.max_columns', 60)
```

```
In [2]: cd md
```

```
/home/raisa/md
```

I download non-overlapping randomly selected samples from file auth.txt.gz so that each file contains roughly the same number of successes and fails.

```
In [54]: all_df=[]
         nfiles=15
         for i in range(nfiles):
             filename = 'msample%d.csv' % i
             print i
             all_df.append(pd.read_csv(filename, header=None))
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
```

In [55]:
```
all_df[0]
```

Out[55]:

	0	1	2	3	4	
0	2	U26@DOM1	U26@DOM1	C616	U26	
1	9	U101@DOM1	U101@DOM1	C1862	C1862	
2	33	C2025$@DOM1	C2025$@DOM1	C467	C467	
3	47	C2653$@DOM1	C2653$@DOM1	C2653	C2653	
4	54	C2653$@DOM1	C2653$@DOM1	C2653	C586	
5	55	C2653$@DOM1	C2653$@DOM1	C2653	C2653	
6	95	U66@DOM1	U66@DOM1	C832	C832	
7	128	C1114$@DOM1	C1114$@DOM1	C1115	C1114	
8	164	C1114$@DOM1	C1114$@DOM1	C1115	C1114	
9	174	C2692$@DOM1	C2692$@DOM1	C528	C528	
10	205	U252@DOM1	U252@DOM1	C2627	C1315	
11	213	C599$@DOM1	C599$@DOM1	C553	C553	
12	239	C3390$@DOM1	C3390$@DOM1	C3392	C3392	
13	243	U22@DOM1	U22@DOM1	C477	U22	
14	286	C1607$@DOM1	C1607$@DOM1	C457	C457	
15	308	C2096$@?	C2096$@?	C25240	C25240	
16	335	U4@DOM1	U4@DOM1	C229	C229	
17	355	C1714$@DOM1	C1714$@DOM1	C612	C612	
18	363	C1527$@DOM1	C1527$@DOM1	C1527	C612	
19	454	C2096$@?	C2096$@?	C457	C457	
20	489	C2902$@DOM1	C2902$@DOM1	C2902	C1065	
21	523	C4334$@DOM1	C4334$@DOM1	C4334	C2106	
22	554	C2653$@DOM1	C2653$@DOM1	C2653	C2653	
23	571	U1825@?	U1825@?	C612	C612	
24	623	U22@DOM1	U22@DOM1	C506	U22	
25	641	C860$@DOM1	C860$@DOM1	C860	C457	
26	673	C2043$@DOM1	C2043$@DOM1	C529	C529	
27	677	C2759$@DOM1	C2759$@DOM1	C2759	C2759	
28	773	LOCAL SERVICE@C3049	LOCAL SERVICE@C3049	C3049	C3049	
29	834	C2480$@DOM1	C2480$@DOM1	C2479	C2479	MICROSOFT_AUTH
...	...	...	...	...	...	
400682	5010840	U9@DOM1	C586$@DOM1	C586	C586	
400683	5010841	U59@?	U59@?	C1634	C1634	
400684	5010861	U8929@?	U8929@?	C19037	C19037	
400685	5010873	U59@?	U59@?	C1634	C1634	
400686	5010874	U9@?	U9@?	C222	C222	
400687	5010879	U22@DOM1	U22@DOM1	C849	U22	
400688	5010879	U9@DOM1	U9@DOM1	C222	C222	
400689	5010884	NETWORK SERVICE@C25102	NETWORK SERVICE@C25102	C25102	C25102	
400690	5010900	C23484$@DOM1	C23484$@DOM1	C23484	C586	
400691	5010907	C1692$@DOM1	C1692$@DOM1	C1692	C1692	
400692	5010916	C743$@DOM1	C743$@DOM1	C586	C586	
400693	5010938	U9@?	U9@?	C222	C222	
400694	5010963	C2344$@DOM1	C2344$@DOM1	C457	C457	
400695	5010970	U22@DOM1	U22@DOM1	C246	U22	
400696	5011005	U59@?	U59@?	C1634	C1634	
400697	5011008	C3188$@DOM1	C3188$@DOM1	C3188	C3188	
400698	5011014	U101@?	U101@?	C3415	C3415	
400699	5011015	U59@?	U59@?	C589	C589	
400700	5011043	U10107@DOM1	U10107@DOM1	C419	C419	
400701	5011067	C27118$@DOM1	C27118$@DOM1	C1369	C1369	

```
400702  5011071         C21596$@DOM1         C21596$@DOM1  C21596    C612
400703  5011083                 U9@?                 U9@?    C222    C222
400704  5011087             U9@DOM1              U9@DOM1    C222    C222
400705  5011110          C398$@DOM1           C398$@DOM1   C1767   C1767
400706  5011116         C1791$@DOM1          C1791$@DOM1   C1065   C1065
400707  5011120         C1617$@DOM1          C1617$@DOM1   C1618    C457
400708  5011157         C7780$@DOM1          C7780$@DOM1   C7780    C528
400709  5011161            U22@DOM1             U22@DOM1    C965     U22
400710  5011167          U6715@DOM1           U6715@DOM1  C10781  C10781
400711  5011195           U199@DOM1           U1825@DOM1   C1929   C1929

[400712 rows x 9 columns]
```

In [56]:
```python
Y=[]
for i in range(nfiles):
    Y.append(all_df[i][8]=='Success')
```

In [13]:
```python
Y[1]
```

Out[13]:
```
0          False
1          False
2          False
3           True
4           True
5          False
6           True
7           True
8           True
9          False
10         False
11         False
12          True
13         False
14          True
15          True
16         False
17          True
18          True
19          True
20         False
21         False
22          True
23          True
24          True
25          True
26         False
27         False
28          True
29         False
           ...
400276      True
400277     False
400278      True
400279     False
400280     False
```

```
400281     True
400282     True
400283    False
400284     True
400285    False
400286     True
400287     True
400288    False
400289     True
400290    False
400291     True
400292     True
400293    False
400294    False
400295    False
400296     True
400297    False
400298    False
400299     True
400300    False
400301     True
400302    False
400303    False
400304     True
400305    False
Name: 8, dtype: bool
```

I here repeat my procedure for generating labeled data and features for training/test data.

In [57]:
```
def map_user(x):
    if x.startswith('C'):
        return 'C'
    elif x.startswith('U'):
        return 'U'
    else:
        return x
```

In [68]:
```
X=[]
for i in range(nfiles):
    df=all_df[i]
    df["source_user"], df["source_domain"] = zip(*df[1].str.split('@').tolist())
    df["source_user"]=df["source_user"].str.rstrip('$')
    df["destination_user"], df["destination_domain"] = zip(*df[2].str.split('@').tolist())
    df["destination_user"]=df["destination_user"].str.rstrip('$')
    df['source_class']=df['source_user'].map(map_user)
    df['destination_class']=df['destination_user'].map(map_user)
    x=pd.DataFrame.from_items([
    ('time', (df[0]%(24*60*60)).astype(int))])
    x['same_user']= (df['destination_user']==df['source_user'])
    x['same_domain']=(df['destination_domain']==df['source_domain'])
    x['source_user_comp_same']=(df[3]==df['source_user'])
    x['destination_user_comp_same']=(df['destination_user']==df[4])
    x['same_comp']=(df[3]==df[4])
    x['source_domain_comp_same']=(df[3]==df['source_domain'])
    x['destination_domain_comp_same']=(df['destination_domain']==df[4])
```

```
    for j in [5,6, 7]:
        for label in sorted(df[j].unique()):
            if label=='?':
                if j==5:
                    x['?_authentication type']=(df[j]==label)
                elif j==6:
                    x['?_logon type']=(df[j]==label)
            else:
                x[label]=(df[j]==label)
    for cl in ['source_class', 'destination_class']:
        for label in sorted(df[cl].unique()):
            if cl=='source_class':
                x['source_'+label]=(df[cl]==label)
            else:
                x['destination_'+label]=(df[cl]==label)
    X.append(x)
```

In [62]: `X[1]`

Out[62]:

	time	same_user	same_domain	source_user_comp_same	destination_user_comp_same	same_comp	so
0	2	True	True	False	True	False	
1	3	True	True	False	False	True	
2	11	True	True	False	False	False	
3	140	True	True	True	False	False	
4	176	True	True	False	False	True	
5	185	True	True	False	False	True	
6	224	True	True	True	False	False	
7	250	True	True	False	False	True	
8	252	True	True	False	False	False	
9	333	True	True	True	True	True	
10	348	True	True	False	True	False	
11	416	True	True	False	True	False	
12	459	True	True	True	True	True	
13	470	True	True	True	True	True	
14	485	True	True	False	False	False	
15	490	True	True	True	False	False	
16	510	True	True	False	True	False	
17	542	True	True	False	False	True	
18	551	True	True	True	False	False	
19	570	True	True	False	False	True	
20	588	True	True	False	True	False	
21	623	True	True	False	True	False	
22	679	True	True	False	True	False	
23	704	True	True	False	False	True	
24	726	True	True	False	False	True	
25	745	True	True	True	False	False	
26	750	True	True	False	False	True	
27	859	True	True	False	False	True	
28	936	True	True	False	False	True	
29	975	True	True	False	False	True	
...	...	...	...	...	...	...	
400276	86055	True	True	False	False	False	
400277	86057	True	True	False	False	True	
400278	86061	True	True	True	True	True	

```
400279  86073     True       True                 False                     False     True
400280  86089     True       True                 False                     False     True
400281  86093     True       True                 False                     False    False
400282  86104     True       True                 False                     False     True
400283  86127     True       True                 False                     False     True
400284  86131     True       True                 False                     False     True
400285  86151     True       True                 False                     False     True
400286  86179     True       True                  True                     False    False
400287  86251     True       True                 False                     False     True
400288  86259     True       True                 False                     False     True
400289  86267     True       True                  True                      True     True
400290  86267     True       True                 False                     False     True
400291  86275     True       True                 False                     False     True
400292  86280     True       True                 False                     False     True
400293  86281     True       True                 False                     False     True
400294  86281     True       True                 False                      True    False
400295  86287     True       True                 False                     False     True
400296  86298     True       True                 False                     False     True
400297  86341     True       True                 False                     False     True
400298  86347     True       True                 False                     False     True
400299  86354     True       True                 False                     False     True
400300  86356     True       True                 False                     False     True
400301  86372     True       True                 False                     False     True
400302  86373     True       True                 False                     False     True
400303  86374     True       True                 False                     False     True
400304  86391     True       True                 False                     False     True
400305  86393     True       True                 False                     False    False

[400306 rows x 56 columns]
```

In [63]:
```
X[0].columns
```

Out[63]:
```
Index([u'time', u'same_user', u'same_domain', u'source_user_comp_same', u'destination_user_comp_s
```

In [64]:
```
[len(entry.columns) for entry in X]
```

Out[64]:
```
[53, 56, 53, 54, 54, 52, 56, 56, 57, 55, 55, 54, 55, 54, 54]
```

I just discovered that my sample sets do not contain the same number of features. Below I am checking to see what the source of this difference.

In [65]:
```
all_col = set(sum([list(entry.columns) for entry in X], []))
[all_col.difference(list(entry.columns)) for entry in X]
```

Out[65]:
```
[{'ACRONIS_RELOGON_AUTHENTICATION_PACKAGE',
  'CygwinLsa',
  'MICROSOFT_AUTHENTICA',
  'MICROSOFT_AUTHENTICATION_P',
  'MICROSOFT_AUTHENTICATION_PA',
  'MICROSOFT_AUTHENTICATION_PACK'},
 {'CygwinLsa', 'MICROSOFT_AUTHENTICA', 'MICROSOFT_AUTHENTICATION_P'},
 {'ACRONIS_RELOGON_AUTHENTICATION_PACKAGE',
  'CygwinLsa',
  'MICROSOFT_AUTHENTICA',
  'MICROSOFT_AUTHENTICATION_P',
  'MICROSOFT_AUTHENTICATION_PA',
```

```
 'MICROSOFT_AUTHENTICATION_PAC'},
{'ACRONIS_RELOGON_AUTHENTICATION_PACKAGE',
 'CygwinLsa',
 'MICROSOFT_AUTHENTICA',
 'MICROSOFT_AUTHENTICATION_P',
 'MICROSOFT_AUTHENTICATION_PAC'},
{'ACRONIS_RELOGON_AUTHENTICATION_PACKAGE',
 'CygwinLsa',
 'MICROSOFT_AUTHENTICA',
 'MICROSOFT_AUTHENTICATION_P',
 'MICROSOFT_AUTHENTICATION_PAC'},
{'ACRONIS_RELOGON_AUTHENTICATION_PACKAGE',
 'CygwinLsa',
 'MICROSOFT_AUTHENTICA',
 'MICROSOFT_AUTHENTICATION_P',
 'MICROSOFT_AUTHENTICATION_PAC',
 'MICROSOFT_AUTHENTICATION_PACK',
 'Setuid'},
{'MICROSOFT_AUTHENTICA',
 'MICROSOFT_AUTHENTICATION_P',
 'MICROSOFT_AUTHENTICATION_PA'},
{'ACRONIS_RELOGON_AUTHENTICATION_PACKAGE',
 'MICROSOFT_AUTHENTICA',
 'MICROSOFT_AUTHENTICATION_PAC'},
{'CygwinLsa', 'MICROSOFT_AUTHENTICATION_P'},
{'ACRONIS_RELOGON_AUTHENTICATION_PACKAGE',
 'CygwinLsa',
 'MICROSOFT_AUTHENTICA',
 'MICROSOFT_AUTHENTICATION_PACKAGE_V1'},
{'MICROSOFT_AUTHENTICA',
 'MICROSOFT_AUTHENTICATION_P',
 'MICROSOFT_AUTHENTICATION_PA',
 'MICROSOFT_AUTHENTICATION_PAC'},
{'CygwinLsa',
 'MICROSOFT_AUTHENTICA',
 'MICROSOFT_AUTHENTICATION_P',
 'MICROSOFT_AUTHENTICATION_PA',
 'MICROSOFT_AUTHENTICATION_PAC'},
{'CygwinLsa',
 'MICROSOFT_AUTHENTICA',
 'MICROSOFT_AUTHENTICATION_P',
 'MICROSOFT_AUTHENTICATION_PA'},
{'CygwinLsa',
 'MICROSOFT_AUTHENTICA',
 'MICROSOFT_AUTHENTICATION_P',
 'MICROSOFT_AUTHENTICATION_PA',
 'Setuid'},
{'ACRONIS_RELOGON_AUTHENTICATION_PACKAGE',
 'MICROSOFT_AUTHENTICA',
 'MICROSOFT_AUTHENTICATION_P',
 'MICROSOFT_AUTHENTICATION_PA',
 'MICROSOFT_AUTHENTICATION_PAC'}]
```

This is potentially different spelling of two different commands/labels. For now I will just remove all the labels that are not present in 15 files of data I have just downloaded. If the scores for machine learning will change noticeably. I will look into ways to clean and incorporate this data.

```
In [69]: col_set = [set(entry.columns) for entry in X]
         common_subset = set.intersection(*col_set)
         drop_cols = [e.difference(common_subset) for e in col_set]
         for entry, to_drop in zip(X, drop_cols):
             print 'dropping', to_drop
             for item in to_drop:
                 del entry[item]
```

```
dropping set(['Setuid', 'MICROSOFT_AUTHENTICATION_PACKAGE_V1', 'MICROSOFT_AUTHENTICATION_PAC'])
dropping set(['MICROSOFT_AUTHENTICATION_PA', 'Setuid', 'ACRONIS_RELOGON_AUTHENTICATION_PACKAGE', 'MICROSO
dropping set(['Setuid', 'MICROSOFT_AUTHENTICATION_PACK', 'MICROSOFT_AUTHENTICATION_PACKAGE_V1'])
dropping set(['MICROSOFT_AUTHENTICATION_PA', 'Setuid', 'MICROSOFT_AUTHENTICATION_PACKAGE_V1', 'MICROSOFT_
dropping set(['MICROSOFT_AUTHENTICATION_PA', 'Setuid', 'MICROSOFT_AUTHENTICATION_PACKAGE_V1', 'MICROSOFT_
dropping set(['MICROSOFT_AUTHENTICATION_PA', 'MICROSOFT_AUTHENTICATION_PACKAGE_V1'])
dropping set(['Setuid', 'MICROSOFT_AUTHENTICATION_PACK', 'ACRONIS_RELOGON_AUTHENTICATION_PACKAGE', 'Cygwi
dropping set(['MICROSOFT_AUTHENTICATION_PA', 'Setuid', 'MICROSOFT_AUTHENTICATION_P', 'CygwinLsa', 'MICROS
dropping set(['MICROSOFT_AUTHENTICATION_PA', 'Setuid', 'ACRONIS_RELOGON_AUTHENTICATION_PACKAGE', 'MICROSO
dropping set(['MICROSOFT_AUTHENTICATION_PA', 'Setuid', 'MICROSOFT_AUTHENTICATION_P', 'MICROSOFT_AUTHENTIC
dropping set(['Setuid', 'CygwinLsa', 'ACRONIS_RELOGON_AUTHENTICATION_PACKAGE', 'MICROSOFT_AUTHENTICATION_
dropping set(['Setuid', 'MICROSOFT_AUTHENTICATION_PACK', 'ACRONIS_RELOGON_AUTHENTICATION_PACKAGE', 'MICRO
dropping set(['Setuid', 'MICROSOFT_AUTHENTICATION_PACK', 'ACRONIS_RELOGON_AUTHENTICATION_PACKAGE', 'MICRO
dropping set(['MICROSOFT_AUTHENTICATION_PACK', 'ACRONIS_RELOGON_AUTHENTICATION_PACKAGE', 'MICROSOFT_AUTHE
dropping set(['Setuid', 'MICROSOFT_AUTHENTICATION_PACK', 'CygwinLsa', 'MICROSOFT_AUTHENTICATION_PACKAGE_V
```

```
In [70]: col0 = list(X[0].columns)
         for i in range(1,nfiles):
             col_i = list(X[i].columns)
             assert col0 == col_i, 'mismatch in %r:\n%s\n%s' % (i, col0, col_i)
```

1 Machine learning with logistic regression with Lasso

```
In [71]: from sklearn import linear_model
         clf_l1_LR = linear_model.LogisticRegression(C=1000, penalty='l1', tol=0.001).fit(X[0], Y[0])
         scores=[]
         scores.append(clf_l1_LR.score(X[0], Y[0]))
         print 'score for training set', scores[0]
         for i in range(1,nfiles):
             scores.append(clf_l1_LR.score(X[i], Y[i]))
             print 'score for test set', i, scores[i]
```

```
score for training set 0.944072051748
score for test set 1 0.94448247091
score for test set 2 0.943976919929
score for test set 3 0.944386639788
score for test set 4 0.944560448937
score for test set 5 0.943735713999
score for test set 6 0.944166904201
score for test set 7 0.943538001825
score for test set 8 0.944438192553
score for test set 9 0.943566597067
score for test set 10 0.944126539894
```

```
score for test set 11 0.944858468573
score for test set 12 0.944788959785
score for test set 13 0.944127431039
score for test set 14 0.943777194073
```

```
In [72]: print 'mean', np.mean(scores), 'std', np.std(scores)
```

```
mean 0.944173502288 std 0.00039856965612
```

Logistic regression with Lasso (L1 penalty) computed over 15 non-overlapping subsets of auth.txt.gz gave me a score with mean 0.9442 and std 0.0004. I believe I am sampling from a normal distribution, which means I have a very narrow gaussian. This in turn means that further sampling will not change my results significantly.