Recommended

Stupid solution to Malware classification

February 8, 2015February 10, 2015 ThierryS

Spoiler Alert: I'm not using any of this for my best solution

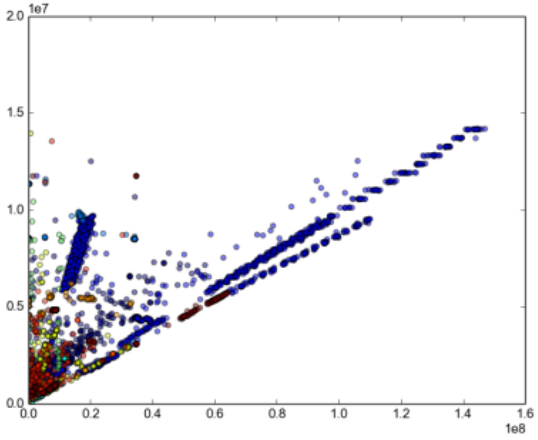
Here was my first submission for the Malware classification challenge (https://www.kaggle.com/c/malware-classification). I always like to submit the stupidest model I can think of to see how the test data compare with my cross-validated approach.

This solution is quiet fast and funny. The whole code runs in less than 5 mn on one core.

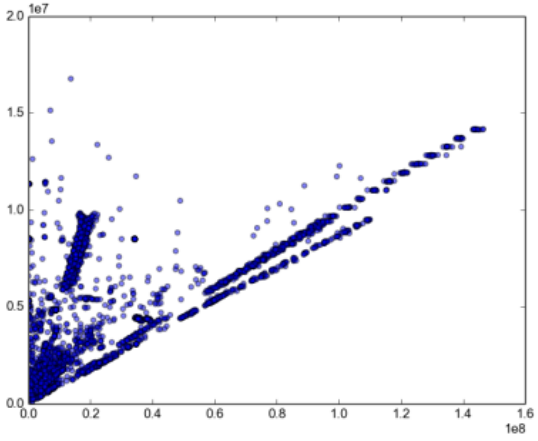
I would say that my approach leverage some data leakage. (Is it really? I guess someone could easily manipulate the file size and make this method useless but hey it's working here so why not)

I'm only using the size of the asm and bytes file as features. Below are scatter plots representing those 2 features.

Train data (First one, each color represents a class) and test data (Second one). Beautiful clusters right? (You can get those plots by downloading and running the python file I distribute at the end of the post)



(https://thierrysilbermann.files.wordpress.com/2015/01/figure_1_v2.png)



(https://thierrysilbermann.files.wordpress.com/2015/01/figure_2_v2.png)

So after extracting the file sizes, I run 2 differents ExtraTreesClassifier from sklearn that will help segment the feature space and capture those clusters.

I end up with this confusion matrix when using 20CV fold.

```
Extra Tree: 0.251
[[1353   35    0    4   10   53    6   56   24]
 [  58 2377    0    6    1   25    2    4    5]
 [   0    5 2937    0    0    0    0    0    0]
 [   5   13    0  447    0    5    0    3    2]
 [  18    2    0    0   17    4    0    1    0]
 [  60   19    1    8    2  639    0   20    2]
 [   4    5    0    0    0    0  389    0    0]
 [  94   12    3    7    1   28    2 1076    5]
 [  26    8    0    3    1    5    0    3  967]]
Extra Tree: 0.218
[[1341   42    0    4    9   51    7   64   23]
 [  49 2372    0    8    2   32    2    7    6]
 [   0    4 2937    0    0    1    0    0    0]
 [   5   11    0  448    0    5    0    4    2]
 [  21    4    1    0   13    1    0    2    0]
 [  57   24    2    9    0  634    0   21    4]
 [   2   10    0    0    0    0  386    0    0]
 [  81   17    4    7    0   28    2 1083    6]
 [  25    5    0    3    1    6    0    3  970]]
```

(https://thierrysilbermann.files.wordpress.com/2015/02/selection_046.png)

Averaging those two models will give you a score around 0.2 with 20CV and a score below 0.2 on the leaderboard.

By the way, I did a small grid search on the parameters, don't waste time changing, you will not increase that much (and forget about RandomForestClassifier too)

The files need to be unzipped.

```
1  __author__ = &quot;Silbermann Thierry&quot;
2  __license__ = &quot;WTFPL&quot;
3
4  import matplotlib.pyplot as plt
5  from os import listdir
6  from os.path import isfile, join
7
8  from sklearn.metrics import log_loss, confusion_matrix
9  from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
```

```
10  from sklearn.cross_validation import cross_val_score, KFold
11
12  import numpy as np
13  import os
14
15  dt = np.dtype([('Id', 'a30'), ('Class', 'u2')])
16  data = np.loadtxt(&quot;trainLabels.csv&quot;, skiprows=1, delimiter = &quot;,&quot;, dty
17
18  X = np.zeros((data.shape[0], 2))
19  Y = data['Class']
20
21
22  for i, (Id, Class) in enumerate(data):
23      X[i][0] = os.path.getsize('train/'+Id[1:-1]+'.asm')
24      X[i][1] = os.path.getsize('train/'+Id[1:-1]+'.bytes')
25
26  mypath = 'test'
27  list_files = [ f for f in listdir(mypath) if isfile(join(mypath,f)) ]
28  set_files = set([ os.path.splitext(el)[0] for el in list_files ])
29
30  Id_list = list(set_files)
31
32  X_test = np.zeros(( len(set_files), 2 ))
33
34  print X_test.shape
35
36  for i, Id in enumerate(Id_list):
37      X_test[i][0] = os.path.getsize('test/'+Id+'.asm')
38      X_test[i][1] = os.path.getsize('test/'+Id+'.bytes')
39
40  ################
41  #     Plot     #
42  ################
43
44  plt.axis((0,1.6*10**8, 0, 2*10**7))
45  plt.scatter(X[:,0], X[:,1], c=Y, alpha=0.5)
46  plt.show()
47
48  plt.axis((0,1.6*10**8, 0, 2*10**7))
49  plt.scatter(X_test[:,0], X_test[:,1], alpha=0.5)
```

```
plt.show()

def run_cv(X,y, clf):

    # Construct a kfolds object
    kf = KFold(len(y),n_folds=10,shuffle=True)
    y_prob = np.zeros((len(y),9))
    y_pred = np.zeros(len(y))

    # Iterate through folds
    for train_index, test_index in kf:
        X_train, X_test = X[train_index], X[test_index]
        y_train = y[train_index]

        clf.fit(X_train,y_train)
        y_prob[test_index] = clf.predict_proba(X_test)
        y_pred[test_index] = clf.predict(X_test)

    return y_prob, y_pred

################
# Get CV score #
################

print &quot;Extra Tree:&quot;,
clf1 = ExtraTreesClassifier(n_estimators=2000, max_features=None, min_samples_leaf=1,
        min_samples_split=9, n_jobs=1, criterion='gini')
p2, pred2 = run_cv(X,Y,clf1)
print &quot;%.3f&quot; % log_loss(Y, p2)
cm = confusion_matrix(Y, pred2)
print(cm)

print &quot;Extra Tree:&quot;
clf2 = ExtraTreesClassifier(n_estimators=2000, max_features=None, min_samples_leaf=2,
        min_samples_split=3, n_jobs=1, criterion='gini')
p3, pred3 = run_cv(X,Y,clf2)
print &quot;%.3f&quot; % log_loss(Y, p3)
cm = confusion_matrix(Y, pred3)
print(cm)

```

```
print 'Combination:',
p6 = (p2 + p3) / 2.
print &quot;%.3f&quot; % log_loss(Y, p6)

'''
clf = ExtraTreesClassifier(n_estimators=2000, max_features=None, min_samples_leaf=1,
        min_samples_split=9, n_jobs=2, criterion='gini')
clf = ExtraTreesClassifier(n_estimators=2000, max_features=None, min_samples_leaf=2,
        min_samples_split=3, n_jobs=2, criterion='gini')
Extra Tree: 0.251
[[1353   35    0    4   10   53    6   56   24]
 [  58 2377    0    6    1   25    2    4    5]
 [   0    5 2937    0    0    0    0    0    0]
 [   5   13    0  447    0    5    0    3    2]
 [  18    2    0    0   17    4    0    1    0]
 [  60   19    1    8    2  639    0   20    2]
 [   4    5    0    0    0    0  389    0    0]
 [  94   12    3    7    1   28    2 1076    5]
 [  26    8    0    3    1    5    0    3  967]]
Extra Tree: 0.218
[[1341   42    0    4    9   51    7   64   23]
 [  49 2372    0    8    2   32    2    7    6]
 [   0    4 2937    0    0    1    0    0    0]
 [   5   11    0  448    0    5    0    4    2]
 [  21    4    1    0   13    1    0    2    0]
 [  57   24    2    9    0  634    0   21    4]
 [   2   10    0    0    0    0  386    0    0]
 [  81   17    4    7    0   28    2 1083    6]
 [  25    5    0    3    1    6    0    3  970]]
'''

###################
# Get prediction #
###################

clf1.fit(X, Y)
result1 = clf1.predict_proba(X_test)

clf2.fit(X, Y)
result2 = clf2.predict_proba(X_test)
```

```
130
131 result = (result1 + result2) / 2.0
132
133 w = open(&amp;quot;stupid_submission.csv&amp;quot;, &amp;quot;w&amp;quot;)
134 w.write('&amp;quot;Id&amp;quot;,&amp;quot;Prediction1&amp;quot;,&amp;quot;Prediction2&amp;quot;,&amp;quot;Prediction3
135 w.write('&amp;quot;Prediction5&amp;quot;,&amp;quot;Prediction6&amp;quot;,&amp;quot;Prediction7&amp;quot;,&amp;quot;Pr
136
137 assert(len(set_files) == X_test.shape[0])
138
139 for i, sample in enumerate(result):
140     string = '&amp;quot;%s&amp;quot;,%s\n' %(Id_list[i], ','.join(map(str, sample)) )
141     w.write(string)
142
143 w.close()
```

Kaggle ï kaggle, python2.7, sklearn