EE379K Enterprise Network Security Lab 4 Report

Student: Sean Wang, szw87
Professor: Mohit Tiwari, Antonio Espinoza
Department of Electrical & Computer Engineering
The University of Texas at Austin

November 10, 2019

Exercise 1

Part 1

For this part, some additional data processing was needed before the pipeline. The following are the steps taken:

```
# get data from CSVs
train = pd.read_csv('data/exercise1/UNSW_NB15_training-set.csv',index_col='id')
test = pd.read_csv('data/exercise1/UNSW_NB15_testing-set.csv',index_col='id')
# remove all rows that are not data on 'Normal' or 'Fuzzers' attacks and make a trai
train_data = train.loc[(train['attack_cat'] == 'Normal') | (train['attack_cat'] == '
test_data = test.loc[(test['attack_cat'] == 'Normal') | (test['attack_cat'] == 'Fuzz
train_labels = train_data['attack_cat']
test_labels = test_data['attack_cat']
# keep only features between 'spkts' and 'is_sm_ips_ports'
train_data = train_data.iloc[:,4:42]
test_data = test_data.iloc[:,4:42]
```

Part 2

In this part, the code was similar except the train and test CSVs were combined and the train/test split was created by randomly dividing the combined DataFrame.

```
# combine CSVs into one DataFrame
com = pd.concat([train, test], ignore_index=True)
```

```
...
# after filtering for columns and rows like in Part 1, create train/test split rando
X_train, X_test, y_train, y_test = train_test_split(com_data, com_labels, test_size=
```

Questions

The inference and results section was similar for both parts, since the process of creating a model, fitting it to the data, and then predicting on test data is similar besides the data used to fit the model. The positive label is 'Fuzzers' and the results are as follows:

```
TRAIN PREDS  part1                part2
precision = 0.9963600264725347 0.9744233104006476
   recall = 0.9934015176509403 0.9851076016692578
f1measure = 0.9948785726086238 0.979736328125
 accuracy = 0.9985602155032279 0.9915050406836907

TEST PREDS  part1                part2
precision = 0.8503908380294491 0.7202686685104702
   recall = 0.25725912890453145 0.758004158004158
f1measure = 0.3950179438463163 0.7386547811993517
 accuracy = 0.8068451418095546 0.8899749245176808
```

1) In part 1, the precision and recall of the predicted training labels are about 0.9964 and 0.9934, respectively. On the other hand, the precision and recall scores of the predicted testing labels are about 0.8504 and 0.2573, respectively.

2) The results on the training data were all above 0.99 while the results on the test data were much lower for recall and f1 measure. The decrease of each score in general is likely since the test data has entries that are dissimilar to the entries in the training data. In particular, the low recall score in the test predictions for part 1 is likely due to a much greater number of false negatives than false positives, since the only difference in the formulas for precision and recall is false positives and false negatives, respectively. This means that in the test data, the model trained on the training data had a significant number of predictions of 'Normal' attacks that were actually 'Fuzzers' (false negatives), in comparison to the number of false positives, where 'Fuzzers' was predicted, but the true label was 'Normal.'

3) The test predictions have a much more balanced spread of scores since the classifier was exposed to data from the test set while training. Overall, this produces a better classifier, since accuracy (a ratio of the correctly predicted labels to the total number of data points) has gone up, despite precision going down. This simply means that in part 2, the number of false positives out of all the positive label predictions is higher compared to the ratio in part 1. In this case, if the organization is worried about fuzzing attacks, then it might be better to have more false positives and be more wary. Additionally, the recall score went up in part 2, meaning that there are relatively less false negatives in the predictions than in part 1. In other words, there are less occurrences of an attack being labeled as 'Normal' when it actually is 'Fuzzers.' Again, this is a good thing when the goal is to separate fuzzing attacks from normal attacks.

Exercise 2

This exercise was similar to the previous exercise in terms of building the model and getting predictions. However, this time accuracy score and a confusion matrix were used to measure performance:

```
from sklearn import metrics
labels_uniq = csv_data['label'].unique()
acc = metrics.accuracy_score(y_test, preds)
matrix = metrics.confusion_matrix(y_test, preds, labels=labels_uniq)
mat_df = pd.DataFrame(
    matrix,
    index=['t:'+x for x in labels_uniq],
    columns=['p:'+x for x in labels_uniq]
)
```

Printing these out resulted in the following (p for predicted, t for true):

```
Accuracy: 0.808055693684734
Confusion Matrix:
                p:AUDIO  p:BROWSING  p:CHAT  p:FILE-TRANSFER  p:MAIL  p:P2P  p:Video
t:AUDIO             263          66       4                2       7      4        3
t:BROWSING           75         614      36                9      15      1       40
t:CHAT               11          89      61                2       0      0        5
t:FILE-TRANSFER       4          20       2              365       8      4       17
t:MAIL               10          55       2               13      65      0        7
```

t:P2P	8	5	1	2	1	553	1
t:VIDEO	24	88	10	24	16	11	256
t:VOIP	10	21	9	4	2	3	5

	p:VIDEO	p:VOIP
t:AUDIO	3	0
t:BROWSING	46	2
t:CHAT	5	1
t:FILE-TRANSFER	17	1
t:MAIL	7	0
t:P2P	7	0
t:VIDEO	256	0
t:VOIP	5	1073

References