AI Lab Project Report

for

Classifying cyber-attacks in network traffic

Prepared by:

Umair Ayaz Aslam (2018484)
Hamza Mawaz Khan (2018132)

10th May 2021

Table of Contents

1. Abstract

This project aims to develop classifiers for the prediction of attack types, given their attributes. The dataset is pre-processed where 23 different attack categories are converted to 5-classes representing attack types, followed by feature engineering to discover the representations needed for classification from raw data. The processed data is then classified using ANN and RandomForest classifiers which are then ensembled by voting method. Finally, the performance of the classified data is evaluated using accuracy by comparing it to test data labels.

2. Introduction

Cyber-attacks are carried out in network traffic all the time, and they have constantly been increasing in the past years due to shifting to remote workforces. Hackers use these to gain access to users' personal information and exploit them. To tackle this vital challenge of cyber-attacks, this project has been carried out to develop classifiers to predict attack types. This project will then help identify various cyber-attacks by classifying them, reducing the chances of information exploitation.

3. Data-preprocessing

Data-preprocessing is the technique of extracting and cleaning data to make it suitable for training Machine Learning models so it transforms raw data into understandable and readable format. Firstly, various libraries are used such as Pandas, NumPy, Sklearn, Matploblib, Tensorflow etc. The data is extracted from two files i.e Dataset.txt and Attack_types.txt. The Dataset file contains 23 different classes of attack_category. The attack_types attribute from Attack-types.txt file is mapped onto the attack_category attribute in Dataset file by comparing the attack_category attribute in both files and hence converted to 5 classes.

4. Feature Engineering

Feature engineering is used to prepare proper input dataset and to improve the performance of machine learning models. Machine learning algorithms like ANN along with clusters such as K-Means clusters need to be scaled before they are fed to the algorithm. Normalization has been performed on the dataset so that the values are shifted and rescaled and they end up ranging between 0 and 1.

Here's the formula for normalization:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Normalization has been performed because our data does not follow a Gaussian distribution and it is good for ANN and K-means clustering.

We have used one hot encoding on input data (three attributes) and labels to esnure there arent any biases within ANN model which inturn increases performance.

5. Classification & Clustering Algorithms

In order to classify the data and identify the attack-types, two classifiers have been used namely ANN and RandomForest. For part b we have used 3 ANN and RandomForest classifiers followed by ensemble learning. Random forest operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes of the individual trees. However, in part c K-means clustering is used to predict the labels which are then utilized to train ANN models.

These models are then ensembled together by using ensemble learning techniques. The first technique we used was max voting in part b, while in part c, the technique used was to take the classification probabilities for each respective attack type and output the maximum among them.

First ANN Model:

```
model = Sequential()
model.add(Dense(33, input_dim=42, activation='relu'))
model.add(Dense(34, activation='relu'))
model.add(Dense(35, activation='relu'))
model.add(Dense(5,activation='softmax'))

model.compile(Adam(lr=0.01), loss = "categorical_crossentropy", metrics = ['accuracy'])

print(model.summary())
```

Second ANN Model:

```
model2 = Sequential()
model2.add(Dense(31, input_dim=42, activation='sigmoid'))
model2.add(Dense(36, activation='sigmoid'))
model2.add(Dense(41, activation='relu'))
model2.add(Dense(5,activation='softmax'))

model2.compile(Adam(lr=0.01), loss = "categorical_crossentropy", metrics = ['accuracy'])

print(model2.summary())
```

Third ANN Model:

```
model3 = Sequential()
model3.add(Dense(20, input_dim=42, activation='relu'))
model3.add(LeakyReLU(alpha=0.05))
model3.add(Dense(30, activation='relu'))
model3.add(LeakyReLU(alpha=0.05))
model3.add(Dense(40, activation='relu'))
model3.add(Dense(5,activation='softmax'))

model3.compile(Adam(lr=0.01), loss = "categorical_crossentropy", metrics = ['accuracy'])

print(model3.summary())
```

RandomForest Model:

```
clf=RandomForestClassifier(n_estimators=100)
clf.fit(train_x,train_y)
pred4 = clf.predict(test_x)

from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(test_y, pred4))
```

Max Voting Ensembling:

```
print(test_y.shape)                              #Ensembling through voting
result = [0] * test_y.shape[0]
for i in range(test_y.shape[0]):

   for x in range(5):
       arr = [0] * 5
       arr[prediction[i]]+=1
       arr[prediction2[i]]+=1
       arr[prediction3[i]]+=1
       arr[prediction4[i]]+=1

       result[i]  = np.argmax(arr)
```

Part c Ensembling:

```
print(y_test.shape)
result = [0] * y_test.shape[0]
for i in range(y_test.shape[0]):
  sum = 0
  sum = pred[i] + pred2[i] + pred3[i]
  #print(sum)
  sum = np.array(sum)
  result[i] = np.argmax(sum)
  #print(result)
```

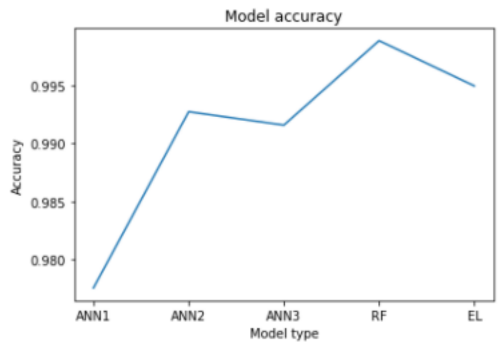
K-Means clustering:

```
kmeans = KMeans(n_clusters=5, max_iter=5).fit(cll)
label = kmeans.predict(cll)
centroids = kmeans.cluster_centers_
```

6. Comparison & Performance Evaluation

The graph computes accuracy by comparing the original label with the output of each respective model as well as with the ensembled model output.

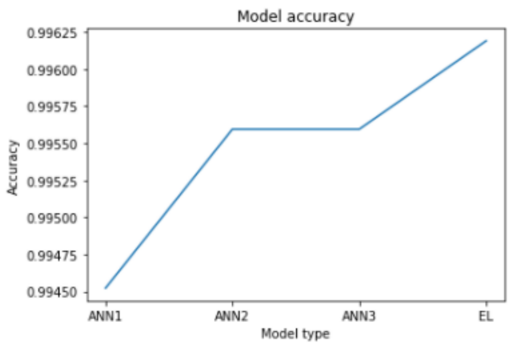
Part b

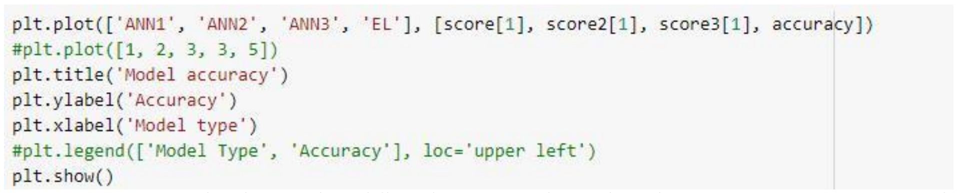


```
plt.plot(['ANN1', 'ANN2', 'ANN3', 'RF', 'EL'], [score[1], score2[1], score3[1], score4, accuracy])
#plt.plot([1, 2, 3, 3, 5])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Model type')
#plt.legend(['Model Type', 'Accuracy'], loc='upper left')
plt.show()
```

As we can see the best classification was done by the RandomForest model as compared to the three ANN models. Our overall accuracy after ensembling came to 99.456%.

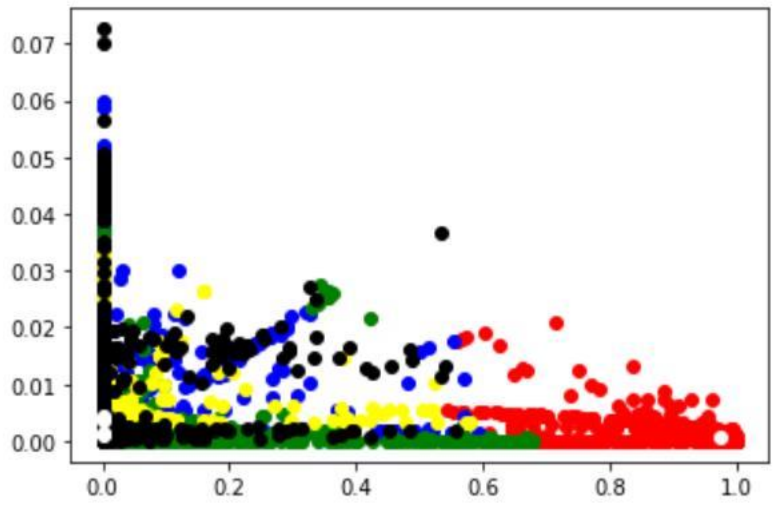
Part c

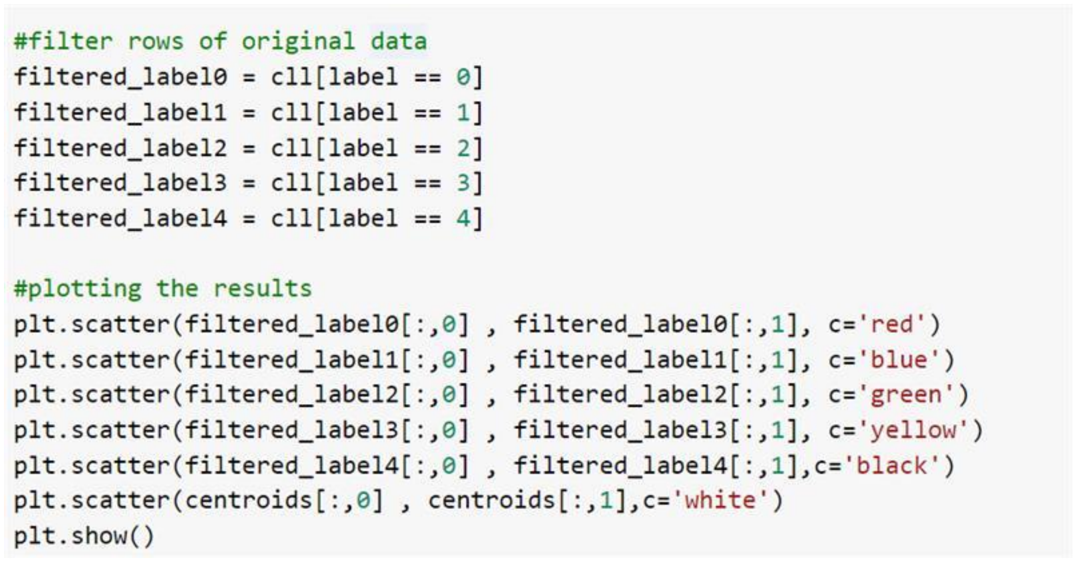




```
plt.plot(['ANN1', 'ANN2', 'ANN3', 'EL'], [score[1], score2[1], score3[1], accuracy])
#plt.plot([1, 2, 3, 3, 5])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Model type')
#plt.legend(['Model Type', 'Accuracy'], loc='upper left')
plt.show()
```

As we can see the best classification was done by the ANN3 as compared to the two other ANN models. Our overall accuracy after ensembling came to 99.54%





```
#filter rows of original data
filtered_label0 = cll[label == 0]
filtered_label1 = cll[label == 1]
filtered_label2 = cll[label == 2]
filtered_label3 = cll[label == 3]
filtered_label4 = cll[label == 4]

#plotting the results
plt.scatter(filtered_label0[:,0] , filtered_label0[:,1], c='red')
plt.scatter(filtered_label1[:,0] , filtered_label1[:,1], c='blue')
plt.scatter(filtered_label2[:,0] , filtered_label2[:,1], c='green')
plt.scatter(filtered_label3[:,0] , filtered_label3[:,1], c='yellow')
plt.scatter(filtered_label4[:,0] , filtered_label4[:,1],c='black')
plt.scatter(centroids[:,0] , centroids[:,1],c='white')
plt.show()
```

K-means clustering has clustered the labels into 5 different categories with each category represented by a different color.

7. Conclusion

In conclusion we initially extracted the data from txt file retrieved the relevant information by combining the files. Preprocessed the data, performed feature engineering, built and ran ANN and random forest model. Next, we performed clustering, trained models again and evaluated the accuracy.