Trojan Detection

In [1]:

```
import pandas as pd
```

In [2]:

```
df = pd.read_csv("../input/trojan-detection/Trojan_Detection.csv", sep = r',', skipini

df.head()
```

Out[2]:

	Unnamed: 0	Flow ID	Source IP	Source Port	Destination IP	Destination Port	Protocol	Timestamp	Du
0	73217	10.42.0.42-121.14.255.84-49975-80-6	10.42.0.42	49975	121.14.255.84	80	6	17/07/2017 01:18:33	107
1	72089	172.217.6.226-10.42.0.42-443-49169-17	10.42.0.42	49169	172.217.6.226	443	17	17/07/2017 10:25:25	2
2	96676	10.42.0.1-10.42.0.42-53-37749-17	10.42.0.42	37749	10.42.0.1	53	17	30/06/2017 07:16:12	10
3	42891	10.42.0.1-10.42.0.42-53-41352-17	10.42.0.42	41352	10.42.0.1	53	17	13/07/2017 03:48:44	2
4	169326	10.42.0.151-107.22.241.77-44353-443-6	10.42.0.151	44353	107.22.241.77	443	6	05/07/2017 10:47:35	656

5 rows × 86 columns

Pre-Cleaning data

In [3]:

```
df = df.dropna()

df.drop(["Unnamed: 0"], axis = 1).values

df = df.replace("Trojan", 1)
df = df.replace("Benign", 0)

df.head()
```

Out[3]:

	Unnamed: 0	Flow ID	Source IP	Source Port	Destination IP	Destination Port	Protocol	Timestamp	Du
0	73217	10.42.0.42-121.14.255.84-49975-80-6	10.42.0.42	49975	121.14.255.84	80	6	17/07/2017 01:18:33	107
1	72089	172.217.6.226-10.42.0.42-443-49169-17	10.42.0.42	49169	172.217.6.226	443	17	17/07/2017 10:25:25	2
2	96676	10.42.0.1-10.42.0.42-53-37749-17	10.42.0.42	37749	10.42.0.1	53	17	30/06/2017 07:16:12	10
3	42891	10.42.0.1-10.42.0.42-53-41352-17	10.42.0.42	41352	10.42.0.1	53	17	13/07/2017 03:48:44	2
4	169326	10.42.0.151-107.22.241.77-44353-443-6	10.42.0.151	44353	107.22.241.77	443	6	05/07/2017 10:47:35	656

5 rows × 86 columns

Encoding

In [4]:

```
from sklearn import preprocessing

number = preprocessing.LabelEncoder()

df["Flow ID"] = number.fit_transform(df["Flow ID"])
df["Source IP"] = number.fit_transform(df["Source IP"])
df["Destination IP"] = number.fit_transform(df["Destination IP"])
df["Timestamp"] = number.fit_transform(df["Timestamp"])

df.head()
```

Out[4]:

	Unnamed: 0	Flow ID	Source IP	Source Port	Destination IP	Destination Port	Protocol	Timestamp	Flow Duration	To F Pack
0	73217	46111	7	49975	352	80	6	36269	10743584	
1	72089	74905	7	49169	895	443	17	39241	254217	
2	96676	9217	7	37749	7	53	17	42069	1023244	
3	42891	10418	7	41352	7	53	17	29885	286483	
4	169326	20763	5	44353	220	443	6	16589	65633087	

5 rows × 86 columns

Defining X and y and Splitting data

In [5]:
```
from sklearn.model_selection import train_test_split

# Columns used as predictors
X = df.drop(["Class"], axis = 1).values

y = df["Class"].values

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 0, test_size
```

In [6]:
```
# from pandas import read_csv
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error


#Scale data, otherwise model will fail.
#Standardize features by removing the mean and scaling to unit variance
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(X_train)

X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)


# define the model
#Experiment with deeper and wider networks
model = Sequential()

model.add(Dense(128, input_dim=85, activation='relu'))
model.add(Dense(64, input_dim=85, activation='relu'))

model.add(Dense(64, input_dim=85, activation='relu'))
model.add(Dense(64, input_dim=85, activation='relu'))

#Output layer
model.add(Dense(1, activation='relu'))

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()

history = model.fit(X_train_scaled, y_train, validation_split=0.2, epochs =20)

from matplotlib import pyplot as plt
#plot the training and validation accuracy and loss at each epoch
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'y', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
```

```
plt.legend()
plt.show()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 128)               11008
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_2 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_3 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 65
=================================================================
Total params: 27,649
Trainable params: 27,649
Non-trainable params: 0
_________________________________________________________________
Epoch 1/20
3550/3550 [==============================] - 8s 2ms/step - loss: 0.3514 - accuracy:
0.9336 - val_loss: 0.2308 - val_accuracy: 0.9801
Epoch 2/20
3550/3550 [==============================] - 6s 2ms/step - loss: 0.2012 - accuracy:
0.9824 - val_loss: 0.2795 - val_accuracy: 0.9779
Epoch 3/20
3550/3550 [==============================] - 6s 2ms/step - loss: 0.3015 - accuracy:
0.9770 - val_loss: 0.1510 - val_accuracy: 0.9889
Epoch 4/20
3550/3550 [==============================] - 6s 2ms/step - loss: 0.2356 - accuracy:
0.9827 - val_loss: 0.3882 - val_accuracy: 0.9735
Epoch 5/20
3550/3550 [==============================] - 6s 2ms/step - loss: 0.3117 - accuracy:
0.9788 - val_loss: 0.2039 - val_accuracy: 0.9864
Epoch 6/20
3550/3550 [==============================] - 7s 2ms/step - loss: 0.2640 - accuracy:
0.9822 - val_loss: 0.2931 - val_accuracy: 0.9804
Epoch 7/20
3550/3550 [==============================] - 6s 2ms/step - loss: 0.3285 - accuracy:
0.9782 - val_loss: 0.4454 - val_accuracy: 0.9709
Epoch 8/20
3550/3550 [==============================] - 6s 2ms/step - loss: 0.3022 - accuracy:
0.9800 - val_loss: 0.1580 - val_accuracy: 0.9894
Epoch 9/20
3550/3550 [==============================] - 6s 2ms/step - loss: 0.2209 - accuracy:
0.9854 - val_loss: 0.4135 - val_accuracy: 0.9730
Epoch 10/20
3550/3550 [==============================] - 7s 2ms/step - loss: 0.4600 - accuracy:
0.9700 - val_loss: 0.5015 - val_accuracy: 0.9671
Epoch 11/20
3550/3550 [==============================] - 7s 2ms/step - loss: 0.4589 - accuracy:
0.9698 - val_loss: 0.3011 - val_accuracy: 0.9798
Epoch 12/20
3550/3550 [==============================] - 6s 2ms/step - loss: 0.2166 - accuracy:
0.9857 - val_loss: 0.2024 - val_accuracy: 0.9867
Epoch 13/20
3550/3550 [==============================] - 6s 2ms/step - loss: 0.3955 - accuracy:
0.9740 - val_loss: 0.4089 - val_accuracy: 0.9734
Epoch 14/20
3550/3550 [==============================] - 6s 2ms/step - loss: 0.3316 - accuracy:
0.9782 - val_loss: 0.2407 - val_accuracy: 0.9843
```

```
Epoch 15/20
3550/3550 [==============================] - 7s 2ms/step - loss: 0.2787 - accuracy:
0.9818 - val_loss: 0.2496 - val_accuracy: 0.9836
Epoch 16/20
3550/3550 [==============================] - 6s 2ms/step - loss: 0.2550 - accuracy:
0.9833 - val_loss: 0.2402 - val_accuracy: 0.9843
Epoch 17/20
3550/3550 [==============================] - 6s 2ms/step - loss: 0.2995 - accuracy:
0.9804 - val_loss: 0.3010 - val_accuracy: 0.9802
Epoch 18/20
3550/3550 [==============================] - 6s 2ms/step - loss: 0.3126 - accuracy:
0.9796 - val_loss: 0.1930 - val_accuracy: 0.9873
Epoch 19/20
3550/3550 [==============================] - 6s 2ms/step - loss: 0.3341 - accuracy:
0.9782 - val_loss: 0.3903 - val_accuracy: 0.9745
Epoch 20/20
3550/3550 [==============================] - 7s 2ms/step - loss: 0.3210 - accuracy:
0.9790 - val_loss: 0.3507 - val_accuracy: 0.9770
```