# CSC 215-01 Artificial Intelligence (Fall 2018) Network Intrusion Detection

1. **Problem Formulation**

Software to detect network intrusions protects a computer network from unauthorized users, including perhaps insiders. This project aims to build a network intrusion detector, a predictive model capable of distinguishing between bad connections, called intrusions or attacks, and good normal connections.

Model this problem as a BINARY classification problem. Use the following models to detect bad connections (intrusions). Compare the accuracy, recall, precision and F1-score of ALL the models. PRINT and PLOT the confusion matrix for each model. Show the ROC curve of each model.

* + Logistic Regression
  + Nearest Neighbor
  + Support Vector Machine
  + Gaussian Naive Bayes
  + Fully-Connected Neural Networks
  + Convolutional Neural Networks (CNN)

In all the models, encode good connections as “0” and bad connections as “1”. To achieve this, you may want to apply some operations/functions to the label column. Check this out if you need hints:

<https://chrisalbon.com/python/data_wrangling/pandas_apply_operations_to_dataframes/>

Hint: For CNN, find a way to view each sample data as an image. Please refer to our lab tutorial on using CNN to handle data other than images. You may use either Conv2D or Conv1D.

# Dataset

Download link: <https://drive.google.com/open?id=1J3Fgo1DCwuQpljJo8Yspe_PQad7oJb52>

This database contains a wide variety of intrusions simulated in a military network environment.

# Additional Requirements

* + You are required to split data for training and testing. Use training data to train your models and evaluate the model quality using test data
  + Drop any rows with missing values.
  + Encode categorical features and normalize numeric features.
  + You must use EarlyStopping and ModelCheckpoint when training neural networks and CNN using Tensorflow.
  + Tuning the following hyperparameters when training neural networks and CNN using Tensorflow to see how they affect performance
    - **Activation:** relu, sigmoid, tanh

## Layers and neuron counts

* + - **Optimizer:** adam, sgd, rmsprop, and others
    - **Kernel number and kernel size** (for CNN only)

# In-class defense

Each team member must defend your work during the scheduled defense session. Each team have **five minutes** to defend your work in class. In the defense, briefly describe **the basic steps** you took to finish this project by referring audience to the data/code/figures in your Jupyter notebook.

If you implement **additional features (novelty)**, please do mention them to receive credit for novelty. Failure to show up in defense session will result in **zero** point for the project.

# Additional Features

Some possible ways to explore for additional features:

1. Can we do feature selection based on **feature importance analysis** to improve the model quality? Can you use a plot to visualize the importance of each feature?
2. **Can you model this intrusion detection problem as a multi-class classification problem** so that we can detect the type of each intrusion? How good the predictive model can be in this case?
3. This data set has **a big number of redundant records**. Redundant records in the train set will cause learning algorithms to be biased towards the more frequent records, and thus prevent it from learning unfrequent records. Can you address this problem?
4. A grand challenge:

<https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/>

# Hints

* + The CSV file has no column headers. So you may want to add them using the following code after you load data into dataframe using pd.read\_csv():

df.columns = [ 'duration', 'protocol\_type', 'service',

'flag', 'src\_bytes', 'dst\_bytes', 'land',

'wrong\_fragment',

'urgent',

'hot', 'num\_failed\_logins', 'logged\_in', 'num\_compromised', 'root\_shell', 'su\_attempted', 'num\_root', 'num\_file\_creations', 'num\_shells', 'num\_access\_files', 'num\_outbound\_cmds', 'is\_host\_login', 'is\_guest\_login', 'count',

'srv\_count', 'serror\_rate', 'srv\_serror\_rate', 'rerror\_rate', 'srv\_rerror\_rate', 'same\_srv\_rate', 'diff\_srv\_rate', 'srv\_diff\_host\_rate', 'dst\_host\_count', 'dst\_host\_srv\_count',

'dst\_host\_same\_srv\_rate', 'dst\_host\_diff\_srv\_rate', 'dst\_host\_same\_src\_port\_rate', 'dst\_host\_srv\_diff\_host\_rate', 'dst\_host\_serror\_rate', 'dst\_host\_srv\_serror\_rate', 'dst\_host\_rerror\_rate', 'dst\_host\_srv\_rerror\_rate', 'outcome'

]