Network Intrusion Detector
(Design of intelligent cyber security systems against DOS and unauthorised access)

Name: Vishwath Ramachandran 20BCE1039
Anirudh CP 20BCE1394
Kritin D Madhavan 20BCE1536

Idea

Our concept is to develop and train a machine learning model to identify and detect between “bad” connections such as intrusions or attacks or “good” connections which are normal connections.

We have identified a dataset with over 92mb of compressed TCP dump data on over 2 million network connections over a 7 week period to train the model.

Scope

This model can be used to protect servers from the following 4 types of attacks:

- DOS: denial-of-service, e.g. syn flood;
- R2L: unauthorized access from a remote machine, e.g. guessing password;
- U2R: unauthorized access to local superuser (root) privileges, e.g., various ``buffer overflow'' attacks;
- probing: surveillance and other probing, e.g., port scanning.

Novelty

Our model will fully automate the system of identifying and detecting the types of intrusions and attacks. It is not specific to any one type of attack and can be used for the detection of various types of attacks. It can be scales to detect various toher types of attacks as well. It is important to note that the test data is not from the same probability distribution as the training data, and it includes specific attack types not in the training data. This makes the task more realistic. Some intrusion experts believe that most novel attacks are variants of known attacks and the "signature" of known attacks can be sufficient to catch novel variants

Comparative Statement

- **Overview of logistic regression model analysis and application** Zhonghua yu Fang yi xue za zhi [Chinese Journal of Preventive Medicine]
- **Analysis of a Random Forests Model** Gerard Biau Journal of Machine Learning Research 13 (2012)
- **A Review on the Long Short-Term Memory Model** Greg Van Houdt Carlos Mosquera Gonzalo N´apoles December 2020 Artificial Intelligence Review
- **An Introduction to Convolutional Neural Networks** Keiron Teilo O'Shea Neural and Evolutionary Computing (cs.NE)
- **A Comparative Analysis of XGBoost** Candice Bentéjac

Dataset

Lincoln Labs set up an environment to acquire nine weeks of raw TCP dump data for a local-area network (LAN) simulating a typical U.S. Air Force LAN. They operated the LAN as if it were a true Air Force environment, but peppered it with multiple attacks.

The raw training data was about four gigabytes of compressed binary TCP dump data from seven weeks of network traffic. This was processed into about five million connection records. Similarly, the two weeks of test data yielded around two million connection records.

A connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows to and from a source IP address to a target IP address under some well defined protocol. Each connection is labeled as either normal, or as an attack, with exactly one specific attack type. Each connection record consists of about 100 bytes.

Preprocessing

In [3]:

```
ids=load_data()
ids.head()
```

Out[3]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_
0	0	tcp	http	SF	181	5450	0	0	0	0	...	9
1	0	tcp	http	SF	239	486	0	0	0	0	...	19
2	0	tcp	http	SF	235	1337	0	0	0	0	...	29
3	0	tcp	http	SF	219	1337	0	0	0	0	...	39
4	0	tcp	http	SF	217	2032	0	0	0	0	...	49

In [4]:
```
ids.shape
```

Out[4]:
```
(494021, 42)
```

In [5]:
```
ids.drop_duplicates(inplace=Tru
```

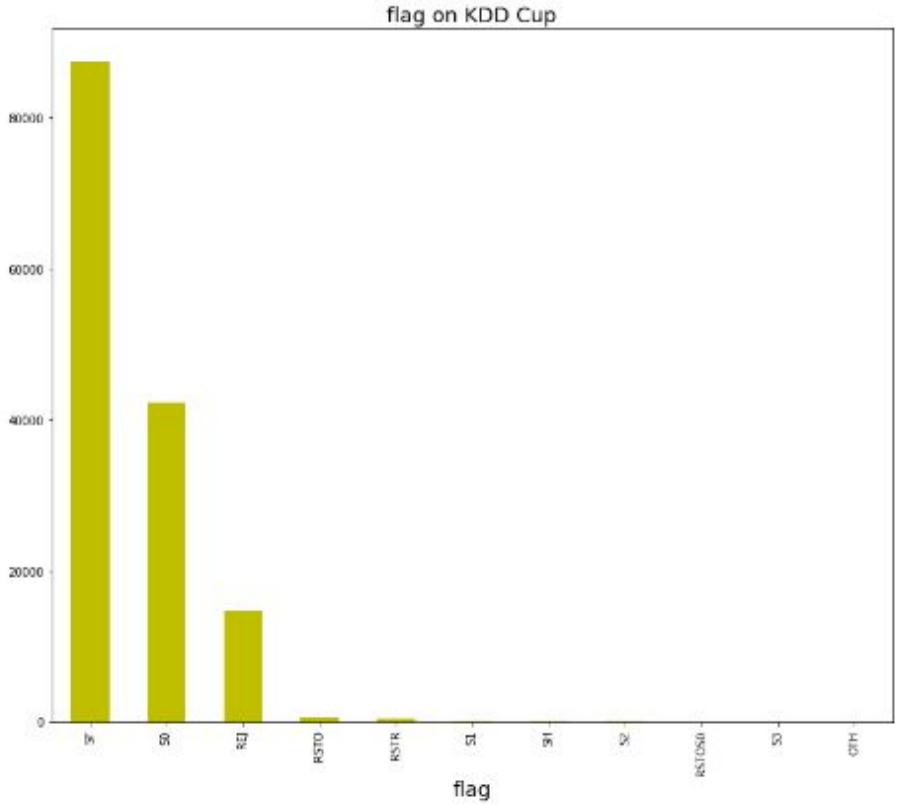
In [6]:
```
ids.shape
```

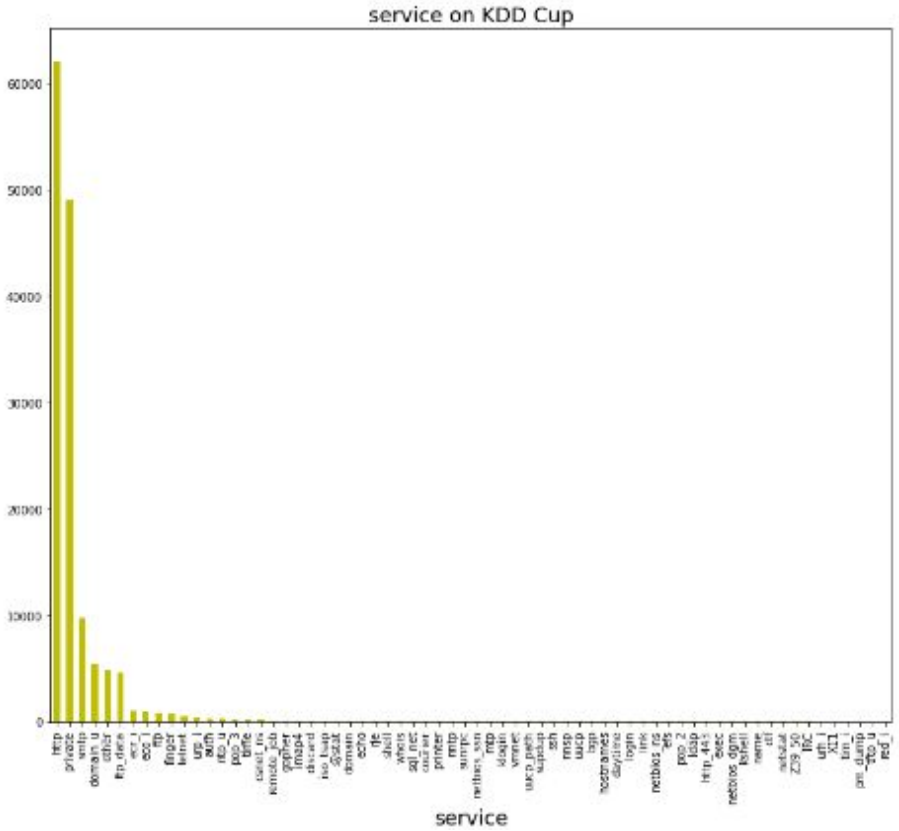
Out[6]:
```
(145585, 42)
```

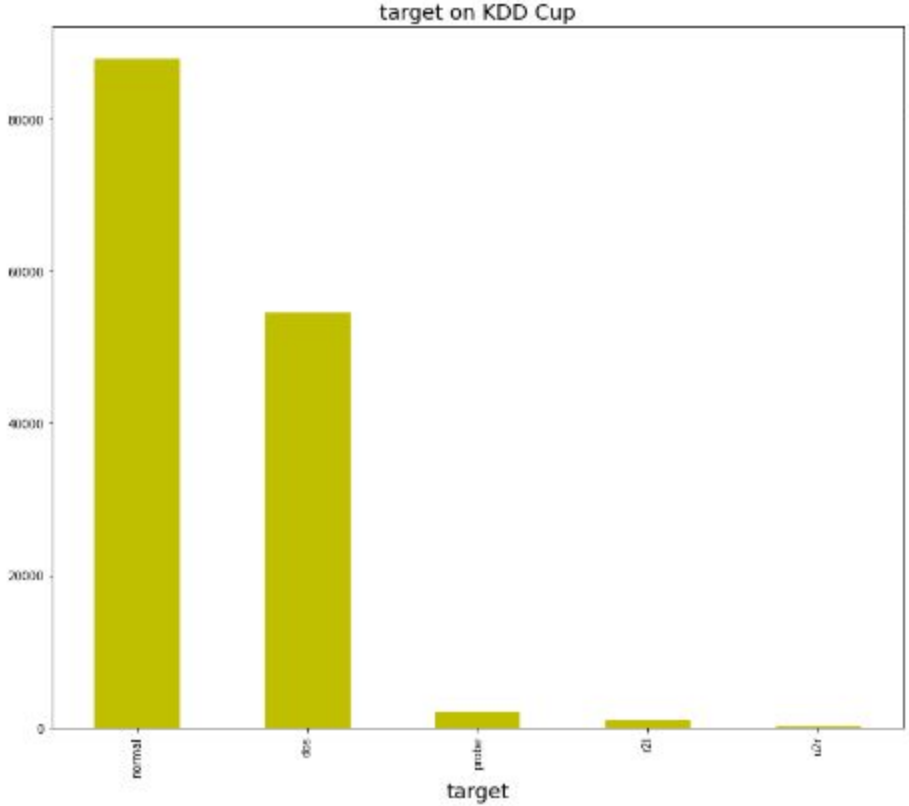
In [7]:
```
ids.info()
```

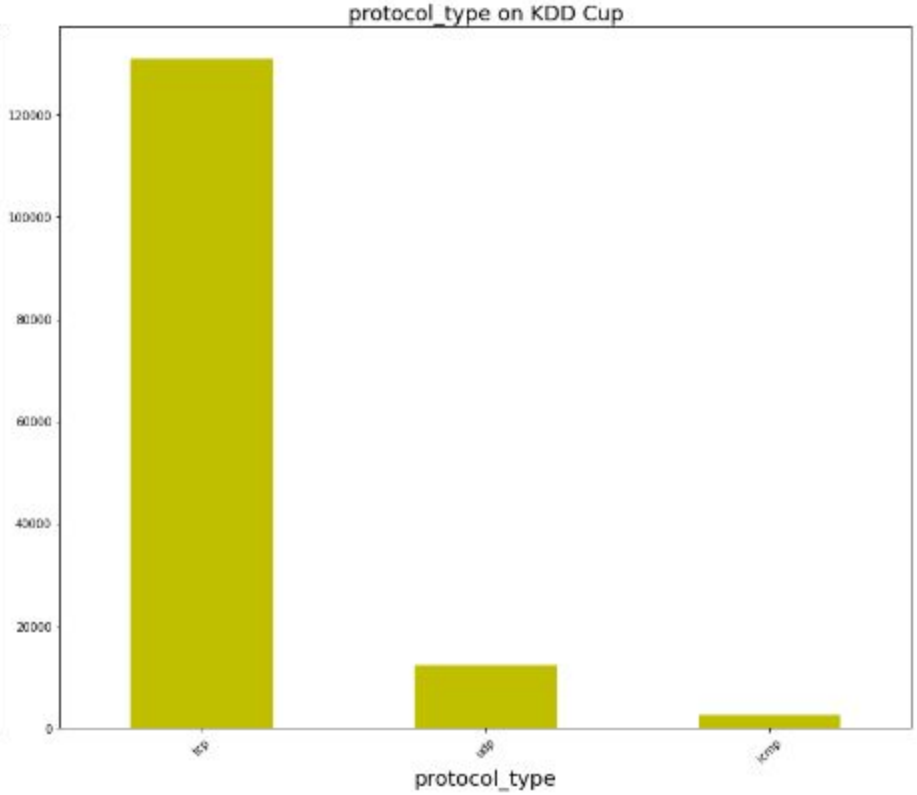
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 145585 entries, 0 to 494020
Data columns (total 42 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   duration                     145585 non-null  int64
 1   protocol_type                145585 non-null  object
 2   service                      145585 non-null  object
 3   flag                         145585 non-null  object
 4   src_bytes                    145585 non-null  int64
 5   dst_bytes                    145585 non-null  int64
 6   land                         145585 non-null  int64
 7   wrong_fragment               145585 non-null  int64
 8   urgent                       145585 non-null  int64
 9   hot                          145585 non-null  int64
 10  num_failed_logins            145585 non-null  int64
 11  logged_in                    145585 non-null  int64
 12  num_compromised              145585 non-null  int64
 13  root_shell                   145585 non-null  int64
 14  su_attempted                 145585 non-null  int64
```

Saving figure Categorical Column visualization






target on KDD Cup
80000
60000
40000
20000
0
normal
dos
probe
r2l
u2r
target


protocol_type on KDD Cup
120000
100000
80000
60000
40000
20000
0
tcp
udp
icmp
protocol_type

In [11]:

```
ids['target'].value_counts()
```

Out[11]:

```
normal    87832
dos       54572
probe      2130
r2l         999
u2r          52
Name: target, dtype: int64
```

Test bed

The model will be run on google colab using python 3.10 and the Nvidia Tesla K80 GPU provided by google.

Expected result

Our objective is to create a model with over 90% accuracy in detecting bad connections.