Network Intrusion Detection

Big Data Analytics (W1)
Ali Munawar
F2019332007

Submitted To:
Mazhar Javed Awan

Overview

The intrusion detection systems are an integral part of modern communication networks. The business environments require a high level of security to safeguard their private data from any unauthorized personnel. The current intrusion detection systems are a step upgrade from the conventional anti-virus software. Two main categories based on their working. These are: • Network Intrusion Detection Systems (NIDS): These systems continuously monitor the network traffic and analyze the packets for a possible rule infringement. • Host-based Intrusion Detection Systems (HIDS): These systems monitor the operating system files of an end-user system to detect malicious software that might temper with its normal functioning.

About Dataset

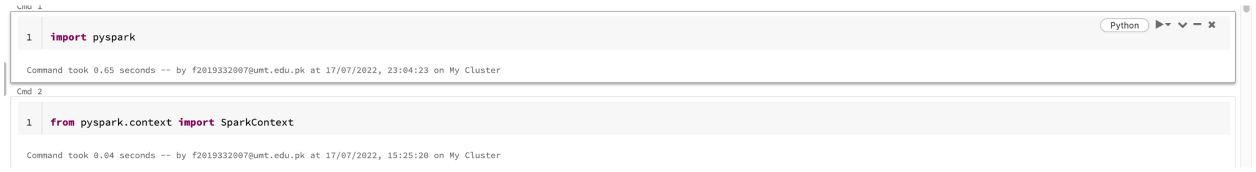
Background
The dataset to be audited was provided which consists of a wide variety of intrusions simulated in a military network environment. It created an environment to acquire raw TCP/IP dump data for a network by simulating a typical US Air Force LAN. The LAN was focused like a real environment and blasted with multiple attacks. A connection is a sequence of TCP packets starting and ending at some time duration between which data flows to and from a source IP address to a target IP address under some well-defined protocol. Also, 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.
For each TCP/IP connection, 41 quantitative and qualitative features are obtained from normal and attack data (3 qualitative and 38 quantitative features) .The class variable has two categories:

- Normal
- Anomalous

The dataset is taken from Kaggle:
https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection

Operations on Resilient distributed datasets

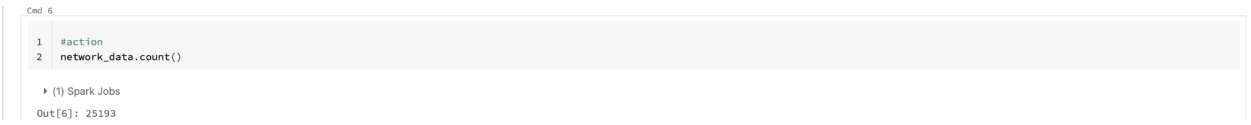
Importing pyspark and SparkContext in databricks



```
import pyspark
Command took 0.65 seconds -- by f2019332007@umt.edu.pk at 17/07/2022, 23:04:23 on My Cluster
Cmd 2
from pyspark.context import SparkContext
Command took 0.04 seconds -- by f2019332007@umt.edu.pk at 17/07/2022, 15:25:20 on My Cluster
```

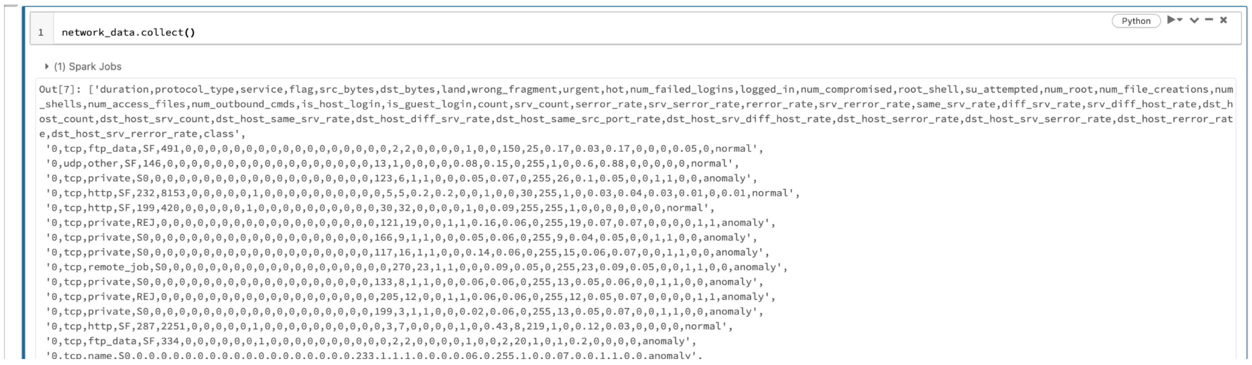
Action:

RDD actions are operations that return non-RDD values. Here we have the **count()** which will tell us about the total elements in the dataset.



```
Cmd 6
#action
network_data.count()

(1) Spark Jobs
Out[6]: 25193
```

Collect() is used to show the

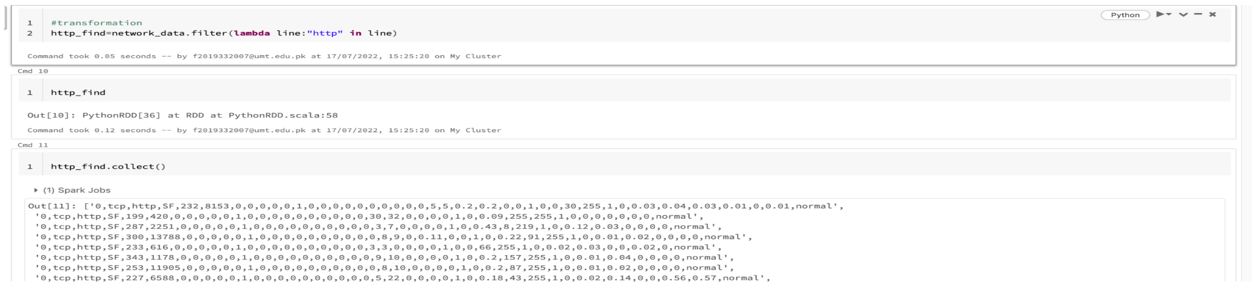


```
network_data.collect()

(1) Spark Jobs
Out[7]: ['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_h
ost_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_rat
e,dst_host_srv_rerror_rate,class',
 '0,tcp,ftp_data,SF,491,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0,0,0,0,1,0,0,150,25,0.17,0.03,0.17,0,0,0,0.05,0,normal',
 '0,udp,other,SF,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,13,1,0,0,0,0,0.08,0.15,0,255,1,0,0.6,0.88,0,0,0,0,0,normal',
 '0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,123,6,1,1,0,0,0.05,0.07,0,255,26,0.1,0.05,0,0,1,1,0,0,anomaly',
 '0,tcp,http,SF,232,8153,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,5,5,0.2,0.2,0,0,1,0,0,30,255,1,0,0.03,0.04,0.03,0.01,0,0.01,normal',
 '0,tcp,http,SF,199,420,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,30,32,0,0,0,0,1,0,0.09,255,255,1,0,0,0,0,0,0,0,normal',
 '0,tcp,private,REJ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,121,19,0,0,1,1,0.16,0.06,0,255,19,0.07,0.07,0,0,0,0,1,1,anomaly',
 '0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,166,9,1,1,0,0,0.05,0.06,0,255,9,0.04,0.05,0,0,1,1,0,0,anomaly',
 '0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,117,16,1,1,0,0,0.14,0.06,0,255,15,0.06,0.07,0,0,1,1,0,0,anomaly',
 '0,tcp,remote_job,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,270,23,1,1,0,0,0.09,0.05,0,255,23,0.09,0.05,0,0,1,1,0,0,anomaly',
 '0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,133,8,1,1,0,0,0.06,0.06,0,255,13,0.05,0.06,0,0,1,1,0,0,anomaly',
 '0,tcp,private,REJ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,205,12,0,0,1,1,0.06,0.06,0,255,12,0.05,0.07,0,0,0,0,1,1,anomaly',
 '0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,199,3,1,1,0,0,0.02,0.06,0,255,13,0.05,0.07,0,0,1,1,0,0,anomaly',
 '0,tcp,http,SF,287,2251,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,3,7,0,0,0,0,1,0,0.43,8,219,1,0,0.12,0.03,0,0,0,0,normal',
 '0,tcp,ftp_data,SF,334,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,2,2,0,0,0,0,1,0,0,2,20,1,0,1,0.2,0,0,0,0,anomaly',
 '0,tcp,name,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,233,1,1,1,0,0,0.06,0,255,1,0,0.07,0,0,1,1,0,0,anomaly',
```

Transformation:

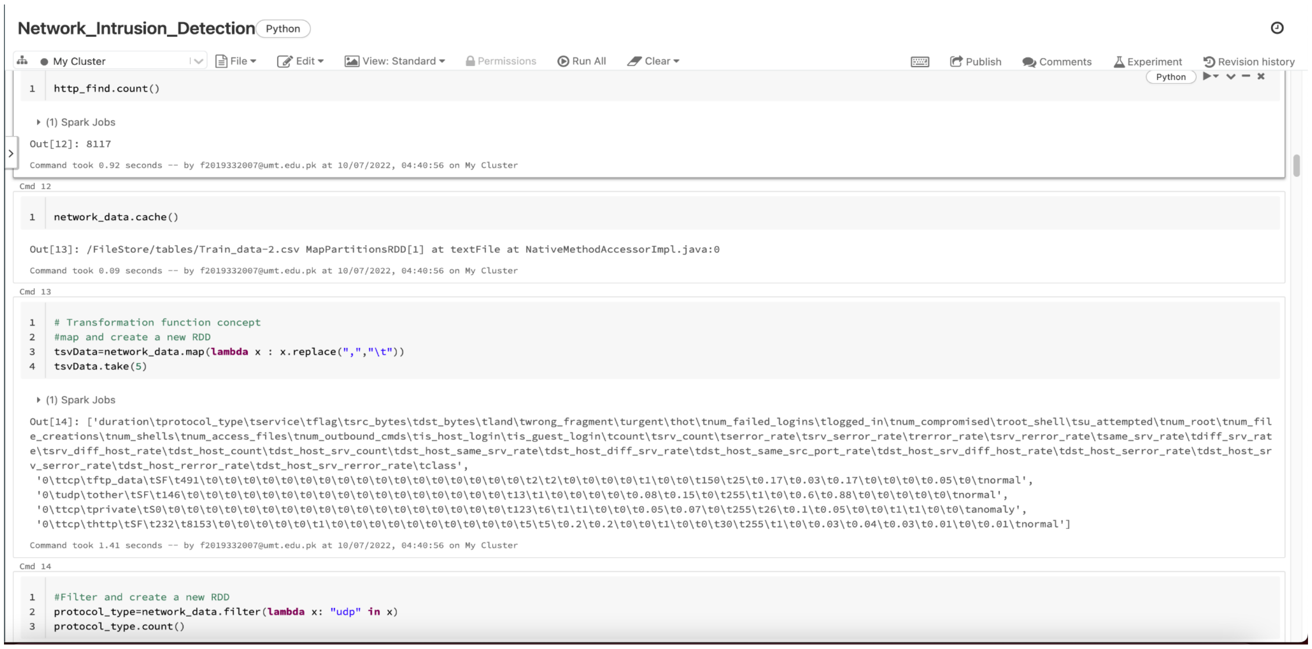
Spark Transformation is a function that produces new RDD from the existing RDDs.

Here we have created a new RDD named http_find.



```
#transformation
http_find=network_data.filter(lambda line:"http" in line)
Command took 0.05 seconds -- by f2019332007@umt.edu.pk at 17/07/2022, 15:25:20 on My Cluster
Cmd 10
http_find
Out[10]: PythonRDD[36] at RDD at PythonRDD.scala:58
Command took 0.12 seconds -- by f2019332007@umt.edu.pk at 17/07/2022, 15:25:20 on My Cluster
Cmd 11
http_find.collect()

(1) Spark Jobs
Out[11]: ['0,tcp,http,SF,232,8153,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,5,5,0.2,0.2,0,0,1,0,0,30,255,1,0,0.03,0.04,0.03,0.01,0,0.01,normal',
 '0,tcp,http,SF,199,420,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,30,32,0,0,0,0,1,0,0.09,255,255,1,0,0,0,0,0,0,0,normal',
 '0,tcp,http,SF,287,2251,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,3,7,0,0,0,0,1,0,0.43,8,219,1,0,0.12,0.03,0,0,0,0,normal',
 '0,tcp,http,SF,300,13788,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,9,0,0.11,0,0,1,0,0.22,91,255,1,0,0.01,0.02,0,0,0,0,normal',
 '0,tcp,http,SF,233,616,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,3,3,0,0,0,0,1,0,0,66,255,1,0,0.02,0.03,0,0,0.02,0,normal',
 '0,tcp,http,SF,343,1178,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,9,10,0,0,0,0,1,0,0.2,157,255,1,0,0.01,0.04,0,0,0,0,normal',
 '0,tcp,http,SF,253,11905,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,10,0,0,0,0,1,0,0.2,87,255,1,0,0.01,0.02,0,0,0,0,normal',
 '0,tcp,http,SF,227,6588,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,5,22,0,0,0,0,1,0,0.18,43,255,1,0,0.02,0.14,0,0,0.56,0.57,normal',
```

Using **map()** to create a new RDD



```
http_find.count()
```

```
(1) Spark Jobs
Out[12]: 8117
Command took 0.92 seconds -- by f2019332007@umt.edu.pk at 10/07/2022, 04:40:56 on My Cluster
```

Cmd 12

```
network_data.cache()
```

```
Out[13]: /FileStore/tables/Train_data-2.csv MapPartitionsRDD[1] at textFile at NativeMethodAccessorImpl.java:0
Command took 0.09 seconds -- by f2019332007@umt.edu.pk at 10/07/2022, 04:40:56 on My Cluster
```

Cmd 13

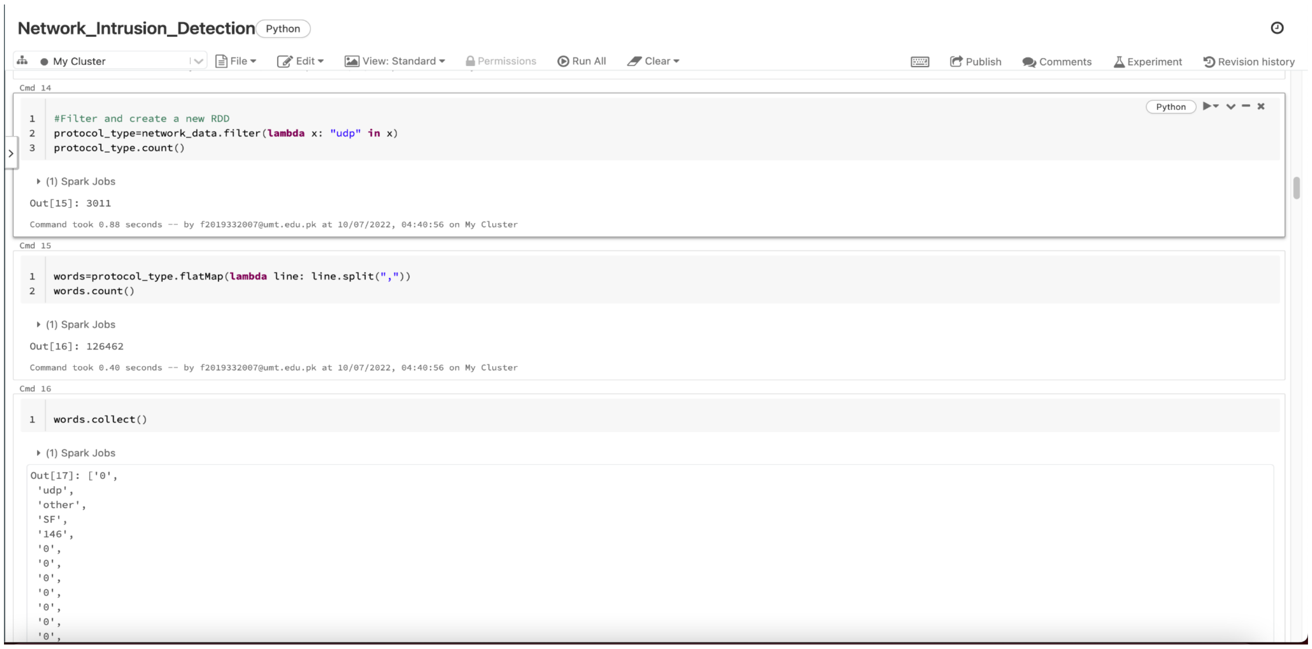
```
# Transformation function concept
#map and create a new RDD
tsvData=network_data.map(lambda x : x.replace(",","\t"))
tsvData.take(5)
```

```
(1) Spark Jobs
Out[14]: ['duration\tprotocol_type\tservice\tflag\tsrc_bytes\tdst_bytes\tland\twrong_fragment\turgent\thot\tnum_failed_logins\tlogged_in\tnum_compromised\troot_shell\tsu_attempted\tnum_root\tnum_fil
e_creations\tnum_shells\tnum_access_files\tnum_outbound_cmds\tis_host_login\tis_guest_login\tcount\tsrv_count\tserror_rate\tsrv_serror_rate\trerror_rate\tsrv_rerror_rate\tsame_srv_rate\tdiff_srv_rat
e\tsrv_diff_host_rate\tdst_host_count\tdst_host_srv_count\tdst_host_same_srv_rate\tdst_host_diff_srv_rate\tdst_host_same_src_port_rate\tdst_host_srv_diff_host_rate\tdst_host_serror_rate\tdst_host_sr
v_serror_rate\tdst_host_rerror_rate\tdst_host_srv_rerror_rate\tclass',
 '0\ttcp\tftp_data\tSF\t491\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t2\t2\t0\t0\t0\t0\t1\t0\t0\t150\t25\t0.17\t0.03\t0.17\t0\t0\t0\t0.05\t0\tnormal',
 '0\tudp\tother\tSF\t146\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t13\t1\t0\t0\t0\t0\t0.08\t0.15\t0\t255\t1\t0\t0.6\t0.88\t0\t0\t0\t0\t0\tnormal',
 '0\ttcp\tprivate\tS0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t123\t6\t1\t1\t0\t0\t0.05\t0.07\t0\t255\t26\t0.1\t0.05\t0\t0\t1\t1\t0\t0\tanomaly',
 '0\ttcp\thttp\tSF\t232\t8153\t0\t0\t0\t0\t0\t1\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t5\t5\t0.2\t0.2\t0\t0\t1\t0\t0\t30\t255\t1\t0\t0.03\t0.04\t0.03\t0.01\t0\t0.01\tnormal']
Command took 1.41 seconds -- by f2019332007@umt.edu.pk at 10/07/2022, 04:40:56 on My Cluster
```

Cmd 14

```
#Filter and create a new RDD
protocol_type=network_data.filter(lambda x: "udp" in x)
protocol_type.count()
```

Filtering using **flatmap()**



Cmd 14

```
#Filter and create a new RDD
protocol_type=network_data.filter(lambda x: "udp" in x)
protocol_type.count()
```

```
(1) Spark Jobs
Out[15]: 3011
Command took 0.88 seconds -- by f2019332007@umt.edu.pk at 10/07/2022, 04:40:56 on My Cluster
```

Cmd 15

```
words=protocol_type.flatMap(lambda line: line.split(","))
words.count()
```

```
(1) Spark Jobs
Out[16]: 126462
Command took 0.40 seconds -- by f2019332007@umt.edu.pk at 10/07/2022, 04:40:56 on My Cluster
```

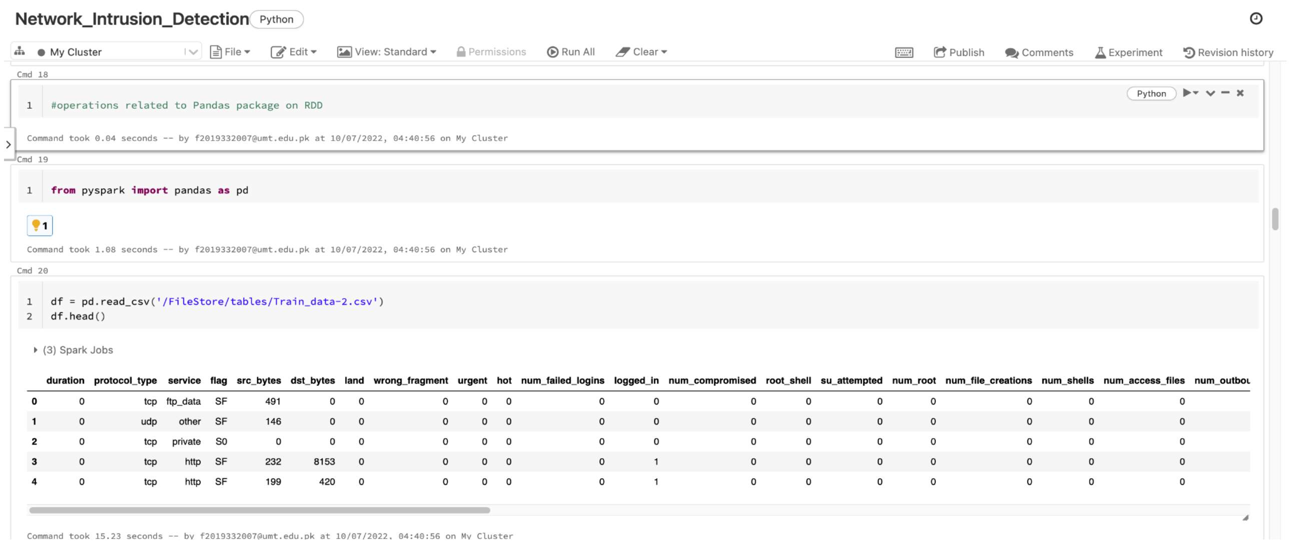
Cmd 16

```
words.collect()
```

```
(1) Spark Jobs
Out[17]: ['0',
 'udp',
 'other',
 'SF',
 '146',
 '0',
 '0',
 '0',
 '0',
 '0',
 '0',
 '0',
```

Operations related to pandas in RDD

Pandas is a software library written for the Python programming language for data manipulation and analysis, it offers data structures and operations for manipulating numerical tables and time series.



Cmd 18

```
#operations related to Pandas package on RDD
```

Command took 0.04 seconds -- by f2019332007@umt.edu.pk at 10/07/2022, 04:40:56 on My Cluster

Cmd 19

```
from pyspark import pandas as pd
```

Command took 1.08 seconds -- by f2019332007@umt.edu.pk at 10/07/2022, 04:40:56 on My Cluster

Cmd 20

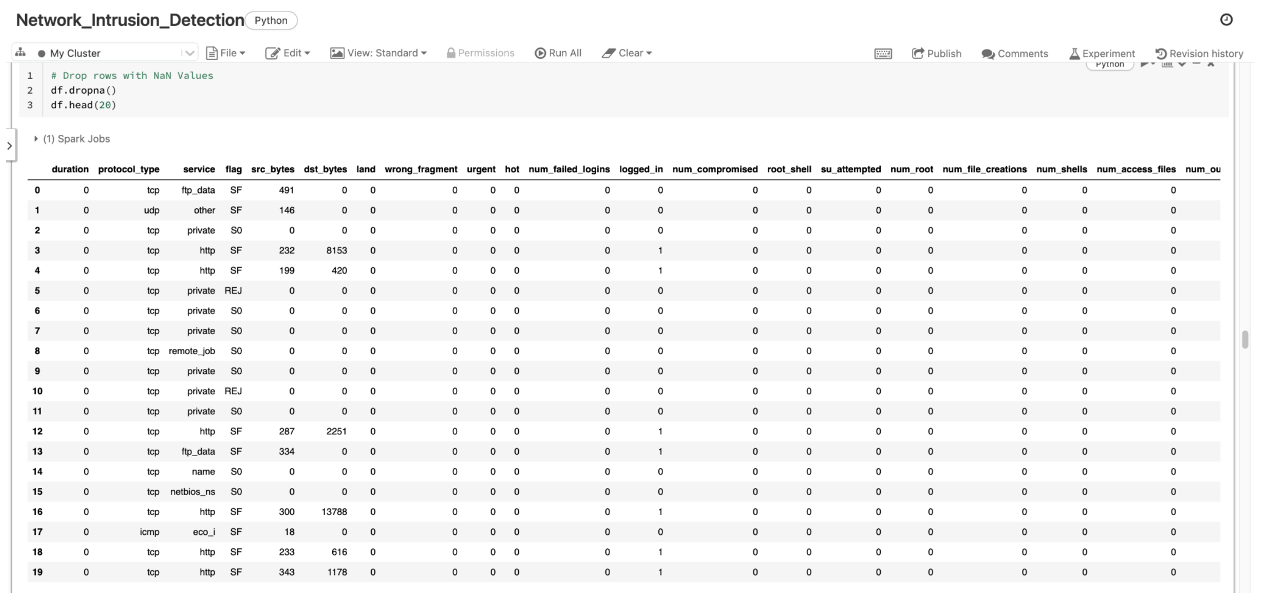
```
df = pd.read_csv('/FileStore/tables/Train_data-2.csv')
df.head()
```

(3) Spark Jobs

	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_outbou
0	0	tcp	ftp_data	SF	491	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	udp	other	SF	146	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	tcp	http	SF	232	8153	0	0	0	0	0	1	0	0	0	0	0	0	0	
4	0	tcp	http	SF	199	420	0	0	0	0	0	1	0	0	0	0	0	0	0	

Command took 15.23 seconds -- by f2019332007@umt.edu.pk at 10/07/2022, 04:40:56 on My Cluster

Dropping the null values in our dataset.



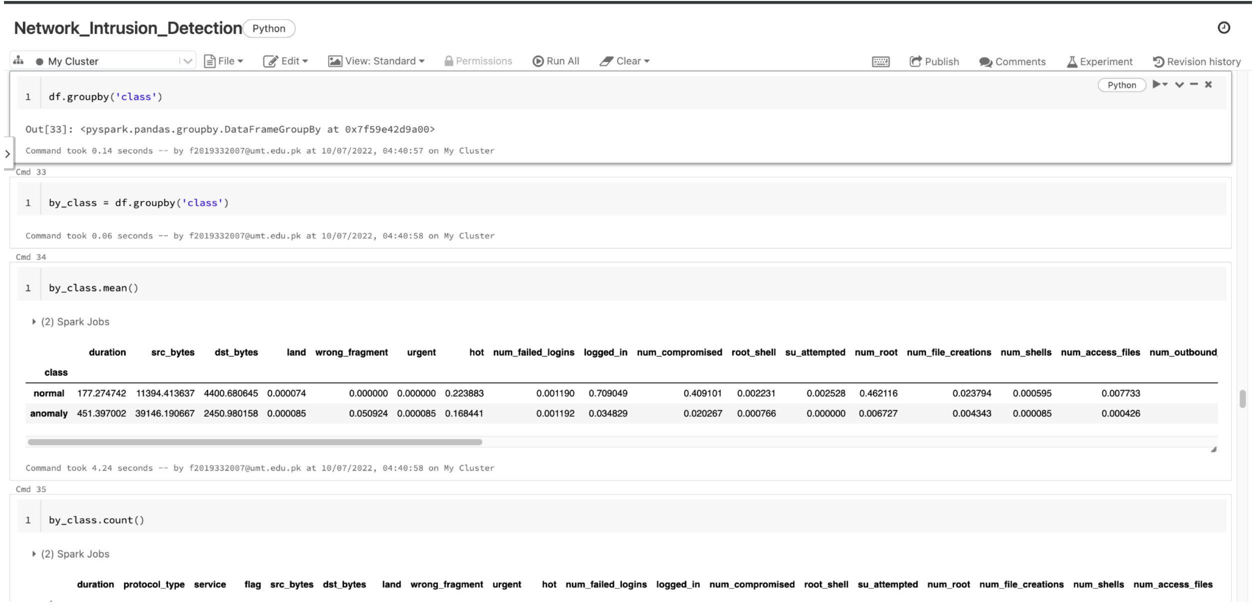
```
# Drop rows with NaN Values
df.dropna()
df.head(20)
```

(1) Spark Jobs

	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_ou
0	0	tcp	ftp_data	SF	491	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	udp	other	SF	146	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	tcp	http	SF	232	8153	0	0	0	0	0	1	0	0	0	0	0	0	0	
4	0	tcp	http	SF	199	420	0	0	0	0	0	1	0	0	0	0	0	0	0	
5	0	tcp	private	REJ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
6	0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
7	0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
8	0	tcp	remote_job	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
9	0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
10	0	tcp	private	REJ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
11	0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
12	0	tcp	http	SF	287	2251	0	0	0	0	0	1	0	0	0	0	0	0	0	
13	0	tcp	ftp_data	SF	334	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
14	0	tcp	name	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
15	0	tcp	netbios_ns	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
16	0	tcp	http	SF	300	13788	0	0	0	0	0	1	0	0	0	0	0	0	0	
17	0	icmp	eco_i	SF	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
18	0	tcp	http	SF	233	616	0	0	0	0	0	1	0	0	0	0	0	0	0	
19	0	tcp	http	SF	343	1178	0	0	0	0	0	1	0	0	0	0	0	0	0	

Applying pandas **GroupBy()**

It allows you to split your data into separate groups to perform computations for better analysis. So, we have split the class and taken the mean of it, to see the total normals and anomalies.



Network_Intrusion_Detection

```
df.groupby('class')
```

```
Out[33]: <pyspark.pandas.groupby.DataFrameGroupBy at 0x7f59e42d9a00>
Command took 0.14 seconds -- by f2019332007@umt.edu.pk at 10/07/2022, 04:40:57 on My Cluster
```

Cmd 33

```
by_class = df.groupby('class')
```

```
Command took 0.06 seconds -- by f2019332007@umt.edu.pk at 10/07/2022, 04:40:58 on My Cluster
```

Cmd 34

```
by_class.mean()
```

(2) Spark Jobs

class	duration	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
normal	177.274742	11394.413637	4400.680645	0.000074	0.000000	0.000000	0.223883	0.001190	0.709049	0.409101	0.002231	0.002528	0.462116	0.023794	0.000595	0.007733	
anomaly	451.397002	39146.190667	2450.980158	0.000085	0.050924	0.000085	0.168441	0.001192	0.034829	0.020267	0.000766	0.000000	0.006727	0.004343	0.000085	0.000426	

```
Command took 4.24 seconds -- by f2019332007@umt.edu.pk at 10/07/2022, 04:40:58 on My Cluster
```

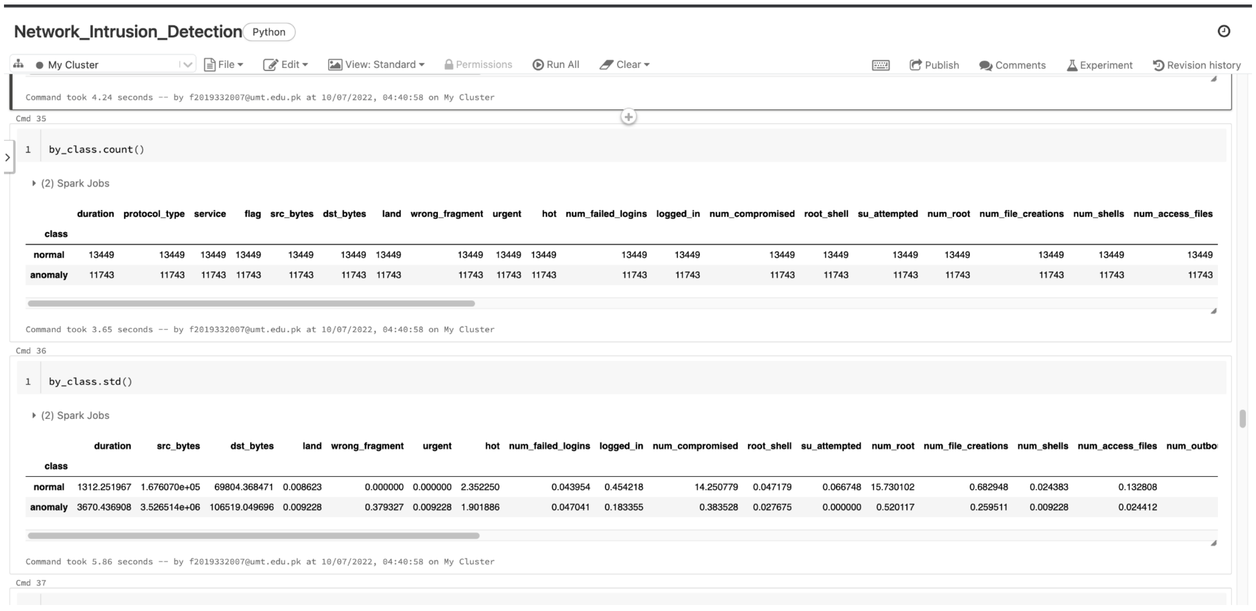
Cmd 35

```
by_class.count()
```

(2) Spark Jobs

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

Taking the count and std of our class variable.



Network_Intrusion_Detection

```
Command took 4.24 seconds -- by f2019332007@umt.edu.pk at 10/07/2022, 04:40:58 on My Cluster
```

Cmd 35

```
by_class.count()
```

(2) Spark Jobs

class	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
normal	13449	13449	13449	13449	13449	13449	13449	13449	13449	13449	13449	13449	13449	13449	13449	13449	13449	13449	13449
anomaly	11743	11743	11743	11743	11743	11743	11743	11743	11743	11743	11743	11743	11743	11743	11743	11743	11743	11743	11743

```
Command took 3.65 seconds -- by f2019332007@umt.edu.pk at 10/07/2022, 04:40:58 on My Cluster
```

Cmd 36

```
by_class.std()
```

(2) Spark Jobs

class	duration	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_outbo
normal	1312.251967	1.676070e+05	69804.368471	0.008623	0.000000	0.000000	2.352250	0.043954	0.454218	14.250779	0.047179	0.066748	15.730102	0.682948	0.024383	0.132808	
anomaly	3670.436908	3.526514e+06	106519.049696	0.009228	0.379327	0.009228	1.901886	0.047041	0.183355	0.383528	0.027675	0.000000	0.520117	0.259511	0.009228	0.024412	

```
Command took 5.86 seconds -- by f2019332007@umt.edu.pk at 10/07/2022, 04:40:58 on My Cluster
```

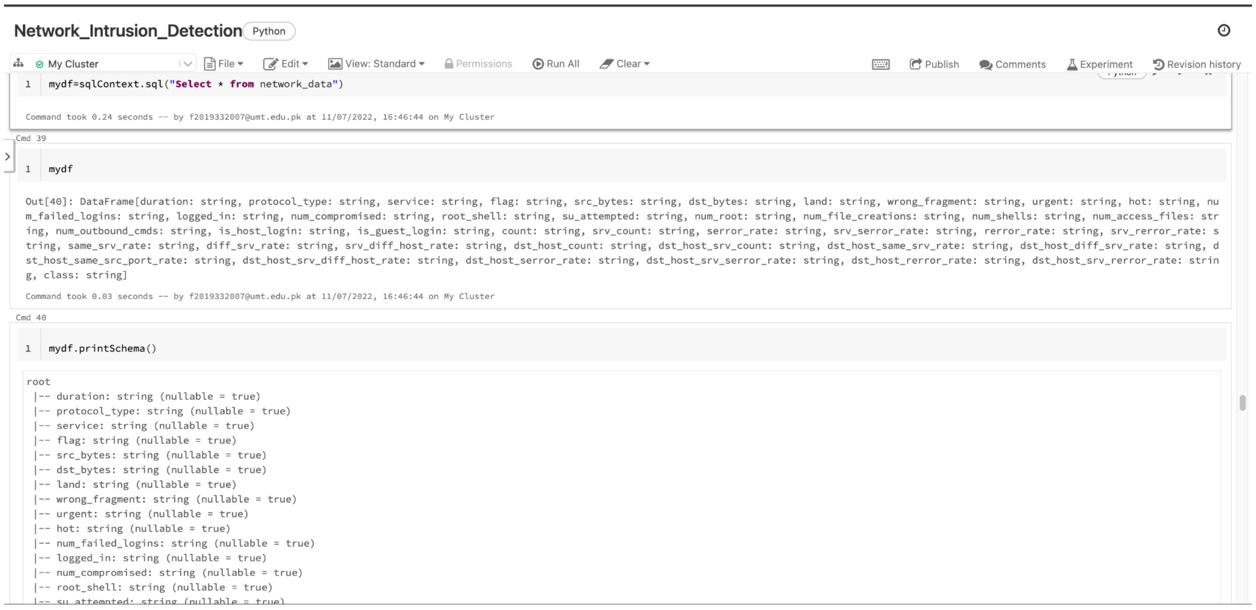
Cmd 37

Operations on Spark SQL

Spark SQL:

Spark SQL is a Spark module for structured data processing. It provides a programming abstraction called DataFrames and can also act as a distributed SQL query engine.

Converting our dataset into spark sql.



```
mydf=sqlContext.sql("Select * from network_data")

Command took 0.24 seconds -- by f2019332007@umt.edu.pk at 11/07/2022, 16:46:44 on My Cluster
```

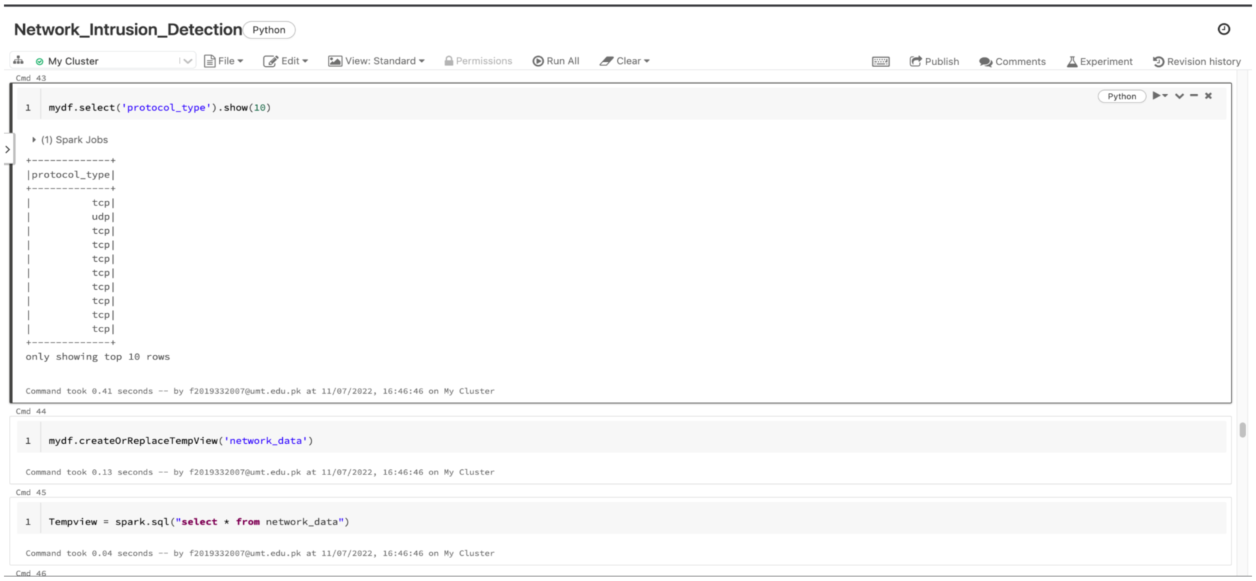
Cmd 39

```
mydf

Out[40]: DataFrame[duration: string, protocol_type: string, service: string, flag: string, src_bytes: string, dst_bytes: string, land: string, wrong_fragment: string, urgent: string, hot: string, nu
m_failed_logins: string, logged_in: string, num_compromised: string, root_shell: string, su_attempted: string, num_root: string, num_file_creations: string, num_shells: string, num_access_files: str
ing, num_outbound_cmds: string, is_host_login: string, is_guest_login: string, count: string, srv_count: string, serror_rate: string, srv_serror_rate: string, rerror_rate: string, srv_rerror_rate: s
tring, same_srv_rate: string, diff_srv_rate: string, srv_diff_host_rate: string, dst_host_count: string, dst_host_srv_count: string, dst_host_same_srv_rate: string, dst_host_diff_srv_rate: string, d
st_host_same_src_port_rate: string, dst_host_srv_diff_host_rate: string, dst_host_serror_rate: string, dst_host_srv_serror_rate: string, dst_host_rerror_rate: string, dst_host_srv_rerror_rate: strin
g, class: string]
Command took 0.03 seconds -- by f2019332007@umt.edu.pk at 11/07/2022, 16:46:44 on My Cluster
```

Cmd 40

```
mydf.printSchema()

root
 |-- duration: string (nullable = true)
 |-- protocol_type: string (nullable = true)
 |-- service: string (nullable = true)
 |-- flag: string (nullable = true)
 |-- src_bytes: string (nullable = true)
 |-- dst_bytes: string (nullable = true)
 |-- land: string (nullable = true)
 |-- wrong_fragment: string (nullable = true)
 |-- urgent: string (nullable = true)
 |-- hot: string (nullable = true)
 |-- num_failed_logins: string (nullable = true)
 |-- logged_in: string (nullable = true)
 |-- num_compromised: string (nullable = true)
 |-- root_shell: string (nullable = true)
 |-- su_attempted: string (nullable = true)
```

Using select command and making new table



Cmd 43

```
mydf.select('protocol_type').show(10)

(1) Spark Jobs
+-------------+
|protocol_type|
+-------------+
|          tcp|
|          udp|
|          tcp|
|          tcp|
|          tcp|
|          tcp|
|          tcp|
|          tcp|
|          tcp|
|          tcp|
+-------------+
only showing top 10 rows

Command took 0.41 seconds -- by f2019332007@umt.edu.pk at 11/07/2022, 16:46:46 on My Cluster
```

Cmd 44

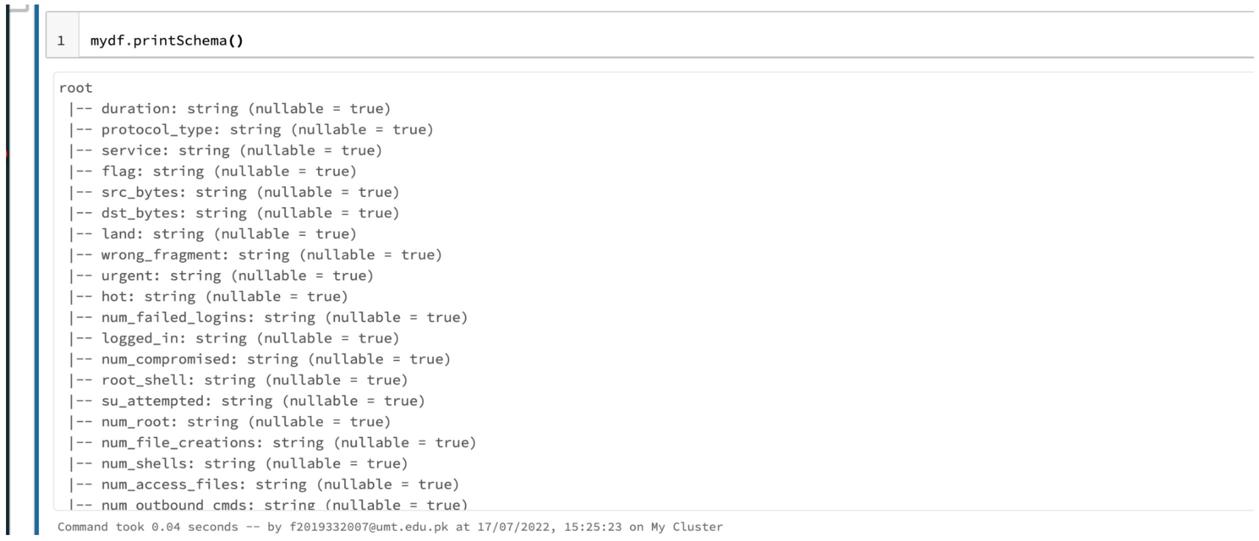
```
mydf.createOrReplaceTempView('network_data')

Command took 0.13 seconds -- by f2019332007@umt.edu.pk at 11/07/2022, 16:46:46 on My Cluster
```

Cmd 45

```
Tempview = spark.sql("select * from network_data")

Command took 0.04 seconds -- by f2019332007@umt.edu.pk at 11/07/2022, 16:46:46 on My Cluster
```

Cmd 46

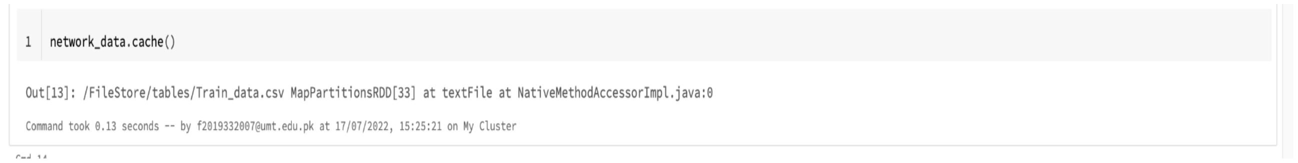
The **printSchema()** method is used to display the schema of the dataframe.



```
mydf.printSchema()
```

```
root
 |-- duration: string (nullable = true)
 |-- protocol_type: string (nullable = true)
 |-- service: string (nullable = true)
 |-- flag: string (nullable = true)
 |-- src_bytes: string (nullable = true)
 |-- dst_bytes: string (nullable = true)
 |-- land: string (nullable = true)
 |-- wrong_fragment: string (nullable = true)
 |-- urgent: string (nullable = true)
 |-- hot: string (nullable = true)
 |-- num_failed_logins: string (nullable = true)
 |-- logged_in: string (nullable = true)
 |-- num_compromised: string (nullable = true)
 |-- root_shell: string (nullable = true)
 |-- su_attempted: string (nullable = true)
 |-- num_root: string (nullable = true)
 |-- num_file_creations: string (nullable = true)
 |-- num_shells: string (nullable = true)
 |-- num_access_files: string (nullable = true)
 |-- num_outbound_cmds: string (nullable = true)
Command took 0.04 seconds -- by f2019332007@umt.edu.pk at 17/07/2022, 15:25:23 on My Cluster
```

Cache() DataFrame **(nework_data)** into memory

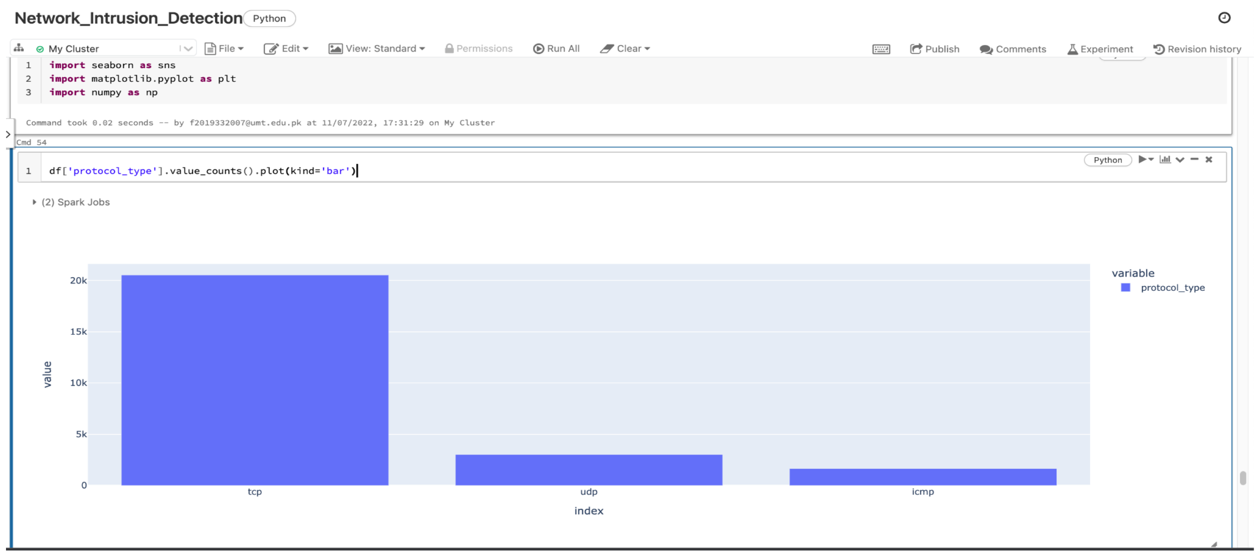


```
network_data.cache()
```

```
Out[13]: /FileStore/tables/Train_data.csv MapPartitionsRDD[33] at textFile at NativeMethodAccessorImpl.java:0

Command took 0.13 seconds -- by f2019332007@umt.edu.pk at 17/07/2022, 15:25:21 on My Cluster
```

Visualization using matplotlib and seaborn:

Matplotlib is a python library used extensively for the visualization of data. While Seaborn is a python library based on matplotlib. Seaborn provides a high-level interface for drawing attractive and informative statistical graphics.



```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

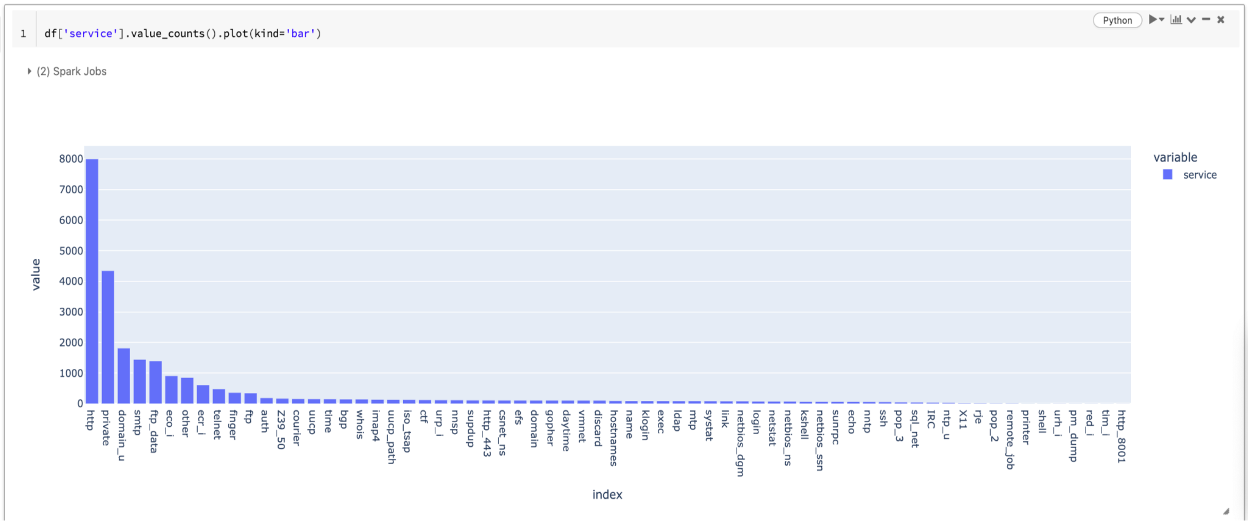
```
Command took 0.02 seconds -- by f2019332007@umt.edu.pk at 11/07/2022, 17:31:29 on My Cluster
```

Cmd 54

```
df['protocol_type'].value_counts().plot(kind='bar')
```

(2) Spark Jobs

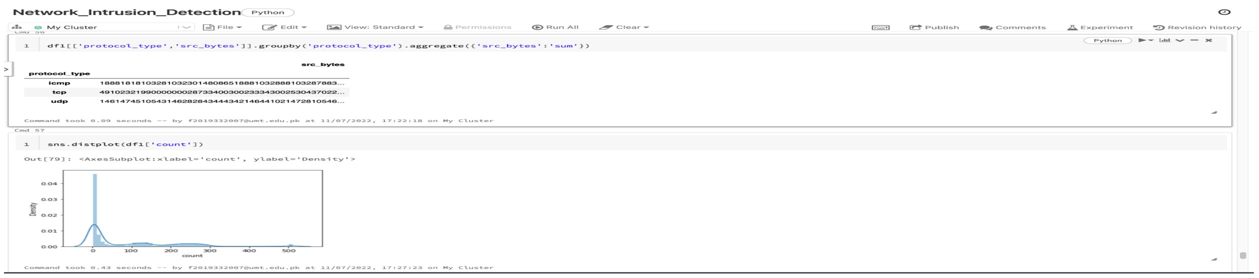
Plotting the total value counts of service



```
df['service'].value_counts().plot(kind='bar')
```

(2) Spark Jobs

Displot using seaborn



Network_Intrusion_Detection Python

```
df1[['protocol_type','src_bytes']].groupby('protocol_type').aggregate({'src_bytes':'sum'})
```

```
                                                        src_bytes
protocol_type
         icmp   18881818103281032301480865188810328881032878883...
          tcp   49102321990000000028733400300233343002530437022...
          udp   1461474510543146282843444342146441021472810546...

Command took 0.09 seconds -- by f2019332007@umt.edu.pk at 11/07/2022, 17:22:18 on My Cluster
```

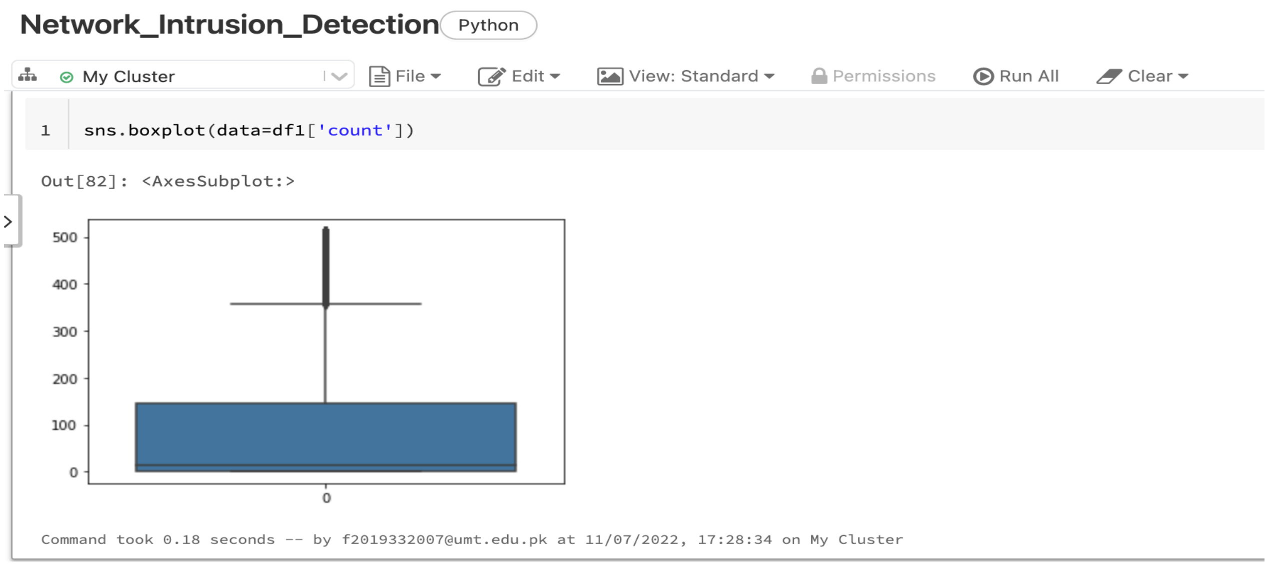
Cmd 57

```
sns.distplot(df1['count'])
```

```
Out[79]: <AxesSubplot:xlabel='count', ylabel='Density'>
```

```
Command took 0.43 seconds -- by f2019332007@umt.edu.pk at 11/07/2022, 17:27:23 on My Cluster
```

Boxplot using seaborn



Network_Intrusion_Detection Python

```
sns.boxplot(data=df1['count'])
```

```
Out[82]: <AxesSubplot:>
```

```
Command took 0.18 seconds -- by f2019332007@umt.edu.pk at 11/07/2022, 17:28:34 on My Cluster
```

Shared Variables

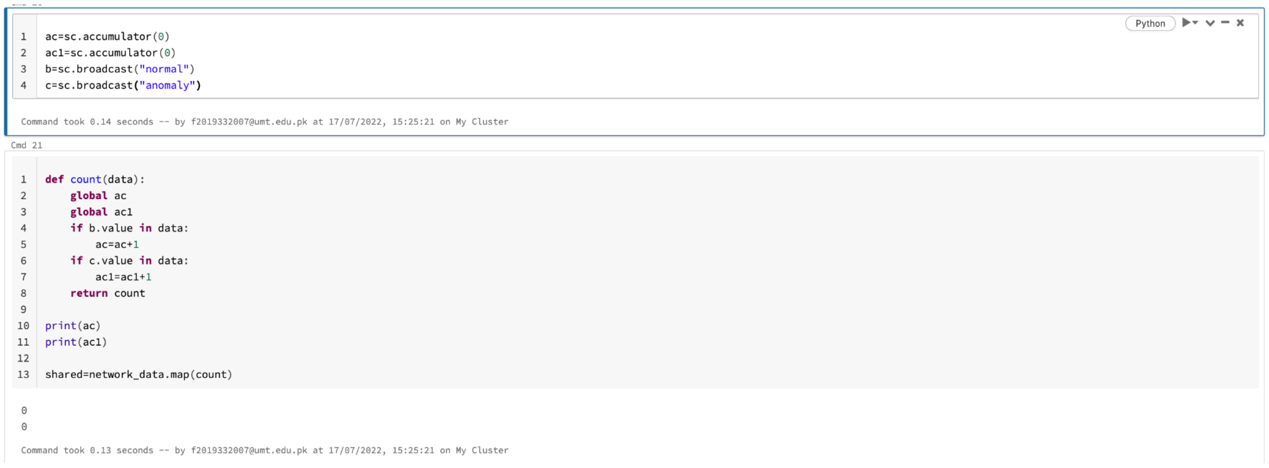
Spark supports two types of shared variables: broadcast variables, which can be used to cache a value in memory on all nodes, and accumulators, which are variables that are only “added” to, such as counters and sums.

Accumulators:

Accumulators are variables that are only “added” to through an associative operation and can therefore, be efficiently supported in parallel. They can be used to implement counters (as in MapReduce) or sums.

Broadcast:

Broadcast variables allow the programmer to keep a read-only variable cached on each machine rather than shipping a copy of it with tasks. They can be used, for example, to give every node a copy of a large input dataset in an efficient manner.



```
ac=sc.accumulator(0)
ac1=sc.accumulator(0)
b=sc.broadcast("normal")
c=sc.broadcast("anomaly")
```

```
Command took 0.14 seconds -- by f2019332007@umt.edu.pk at 17/07/2022, 15:25:21 on My Cluster
```

Cmd 21

```
def count(data):
    global ac
    global ac1
    if b.value in data:
        ac=ac+1
    if c.value in data:
        ac1=ac1+1
    return count

print(ac)
print(ac1)

shared=network_data.map(count)
```

```
0
0

Command took 0.13 seconds -- by f2019332007@umt.edu.pk at 17/07/2022, 15:25:21 on My Cluster
```

Spark Machine Learning

MLlib is Sparks machine learning (ML) library. Its goal is to make practical machine learning scalable and easy.

Use cases of machine learning with Spark:

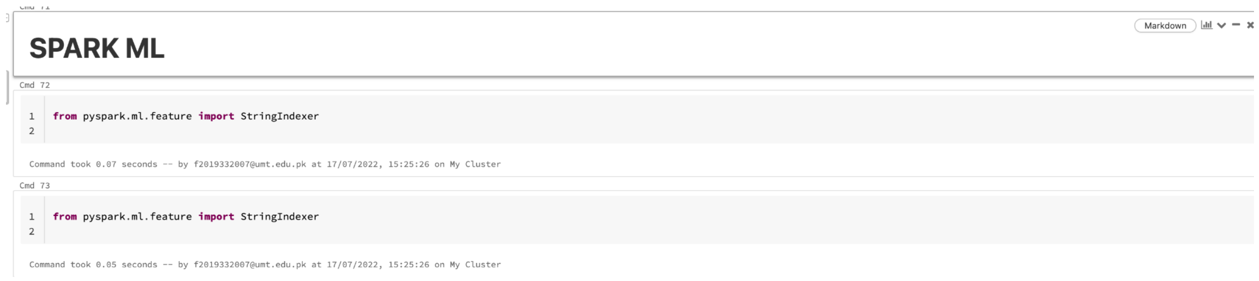
- ML Algorithms: common learning algorithms such as classification, regression, clustering, and collaborative filtering
- Featurization: feature extraction, transformation, dimensionality reduction, and selection

- Pipelines: tools for constructing, evaluating, and tuning ML Pipelines
- Persistence: saving and load algorithms, models, and Pipelines
- Utilities: linear algebra, statistics, data handling, etc.

Spark MLlib (w/ RDDs) versus Spark ML (w/ DataFrames)

Spark.mllib carries the original API built on top of RDDs whereas Spark.ml contains higher-level API built on top of DataFrames for constructing ML pipelines.

Importing **StringIndexer** from **pyspark.ml.feature**



SPARK ML

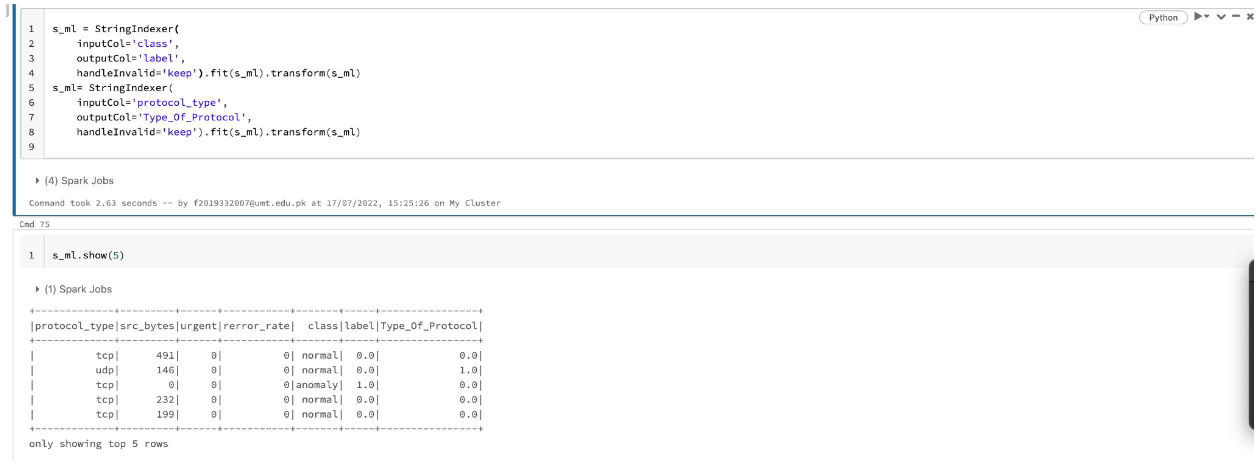
Cmd 72

```
from pyspark.ml.feature import StringIndexer

Command took 0.07 seconds -- by f2019332007@umt.edu.pk at 17/07/2022, 15:25:26 on My Cluster
```

Cmd 73

```
from pyspark.ml.feature import StringIndexer

Command took 0.05 seconds -- by f2019332007@umt.edu.pk at 17/07/2022, 15:25:26 on My Cluster
```

Extracting, Transforming and **selecting features** in our model through **StringIndexer**

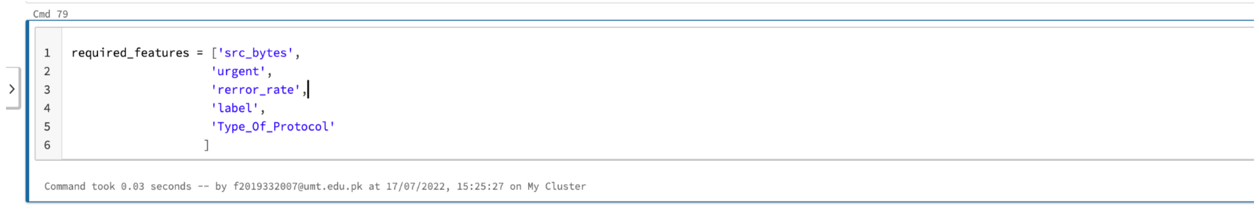


```
s_ml = StringIndexer(
    inputCol='class',
    outputCol='label',
    handleInvalid='keep').fit(s_ml).transform(s_ml)
s_ml= StringIndexer(
    inputCol='protocol_type',
    outputCol='Type_Of_Protocol',
    handleInvalid='keep').fit(s_ml).transform(s_ml)

▸ (4) Spark Jobs
Command took 2.63 seconds -- by f2019332007@umt.edu.pk at 17/07/2022, 15:25:26 on My Cluster
```

Cmd 75

```
s_ml.show(5)

▸ (1) Spark Jobs
+-------------+---------+------+-----------+-------+-----+----------------+
|protocol_type|src_bytes|urgent|rerror_rate|  class|label|Type_Of_Protocol|
+-------------+---------+------+-----------+-------+-----+----------------+
|          tcp|      491|     0|          0| normal|  0.0|             0.0|
|          udp|      146|     0|          0| normal|  0.0|             1.0|
|          tcp|        0|     0|          0|anomaly|  1.0|             0.0|
|          tcp|      232|     0|          0| normal|  0.0|             0.0|
|          tcp|      199|     0|          0| normal|  0.0|             0.0|
+-------------+---------+------+-----------+-------+-----+----------------+
only showing top 5 rows
```

Required Features



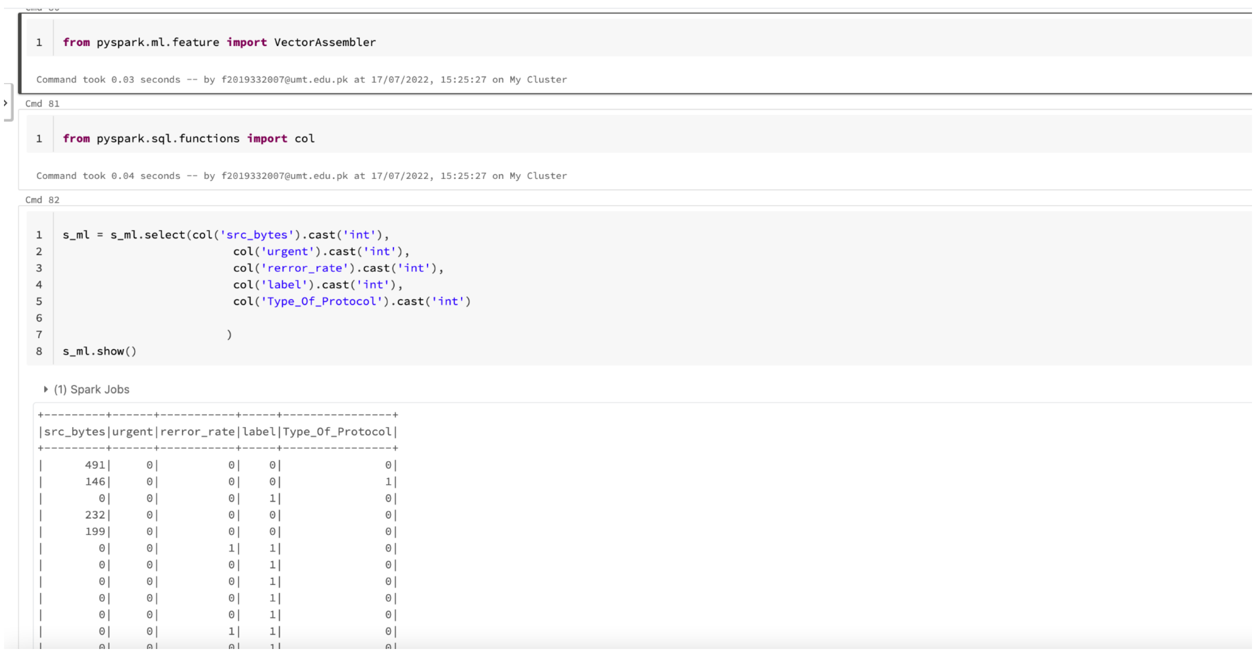
Cmd 79

```
required_features = ['src_bytes',
                     'urgent',
                     'rerror_rate',
                     'label',
                     'Type_Of_Protocol'
                    ]

Command took 0.03 seconds -- by f2019332007@umt.edu.pk at 17/07/2022, 15:25:27 on My Cluster
```

Vector Assembler:

VectorAssembler is a transformer that combines a given list of columns into a single vector column. It is useful for combining raw features and features generated by different feature transformers into a single feature vector, in order to train ML models.

Using VectorAssembler to convert the features into vector columns.



```
from pyspark.ml.feature import VectorAssembler
```

```
Command took 0.03 seconds -- by f2019332007@umt.edu.pk at 17/07/2022, 15:25:27 on My Cluster
```

Cmd 81

```
from pyspark.sql.functions import col
```

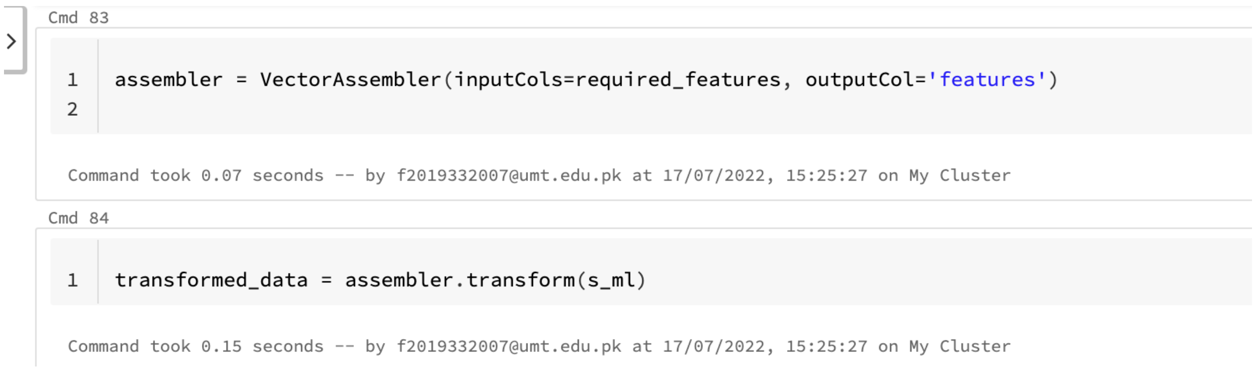
```
Command took 0.04 seconds -- by f2019332007@umt.edu.pk at 17/07/2022, 15:25:27 on My Cluster
```

Cmd 82

```
s_ml = s_ml.select(col('src_bytes').cast('int'),
                         col('urgent').cast('int'),
                         col('rerror_rate').cast('int'),
                         col('label').cast('int'),
                         col('Type_Of_Protocol').cast('int')

                        )
s_ml.show()
```

(1) Spark Jobs

```
+---------+------+-----------+-----+----------------+
|src_bytes|urgent|rerror_rate|label|Type_Of_Protocol|
+---------+------+-----------+-----+----------------+
|      491|     0|          0|    0|               0|
|      146|     0|          0|    0|               1|
|        0|     0|          0|    1|               0|
|      232|     0|          0|    0|               0|
|      199|     0|          0|    0|               0|
|        0|     0|          1|    1|               0|
|        0|     0|          0|    1|               0|
|        0|     0|          0|    1|               0|
|        0|     0|          0|    1|               0|
|        0|     0|          0|    1|               0|
|        0|     0|          1|    1|               0|
|        0|     0|          0|    1|               0|
```



Cmd 83

```
assembler = VectorAssembler(inputCols=required_features, outputCol='features')
```

```
Command took 0.07 seconds -- by f2019332007@umt.edu.pk at 17/07/2022, 15:25:27 on My Cluster
```

Cmd 84

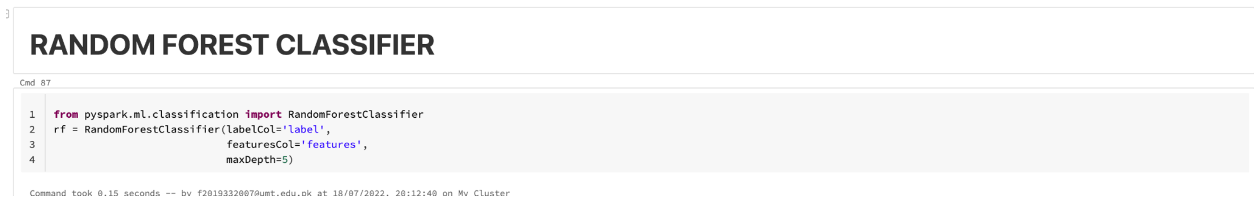
```
transformed_data = assembler.transform(s_ml)
```

```
Command took 0.15 seconds -- by f2019332007@umt.edu.pk at 17/07/2022, 15:25:27 on My Cluster
```

Machine Learning Algorithms

Following are the algorithms which I have implemented in this project:

- Random Forest Classifier
- Decision Tree Classification
- Logistic Regression
- Naive Bayes Classifier



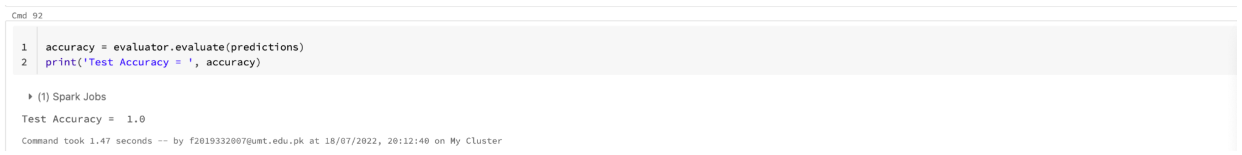
RANDOM FOREST CLASSIFIER

Cmd 87

```
from pyspark.ml.classification import RandomForestClassifier
rf = RandomForestClassifier(labelCol='label',
                            featuresCol='features',
                            maxDepth=5)
```

```
Command took 0.15 seconds -- by f2019332007@umt.edu.pk at 18/07/2022, 20:12:40 on My Cluster
```

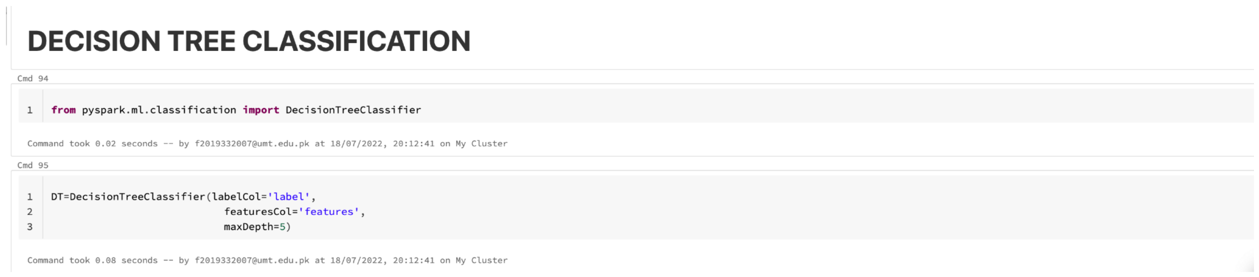
Test Accuracy of Random Forest Classifier



Cmd 92

```
accuracy = evaluator.evaluate(predictions)
print('Test Accuracy = ', accuracy)
```

```
▸ (1) Spark Jobs
Test Accuracy =  1.0
Command took 1.47 seconds -- by f2019332007@umt.edu.pk at 18/07/2022, 20:12:40 on My Cluster
```



DECISION TREE CLASSIFICATION

Cmd 94

```
from pyspark.ml.classification import DecisionTreeClassifier
```

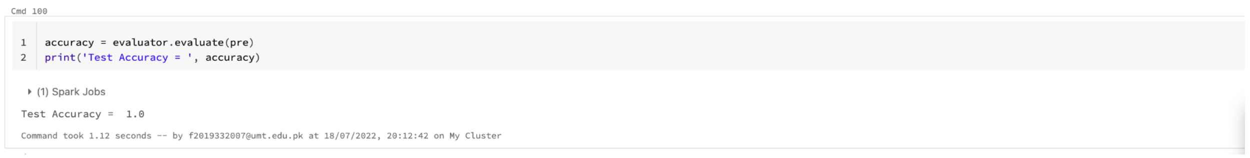
```
Command took 0.02 seconds -- by f2019332007@umt.edu.pk at 18/07/2022, 20:12:41 on My Cluster
```

Cmd 95

```
DT=DecisionTreeClassifier(labelCol='label',
                          featuresCol='features',
                          maxDepth=5)
```

```
Command took 0.08 seconds -- by f2019332007@umt.edu.pk at 18/07/2022, 20:12:41 on My Cluster
```

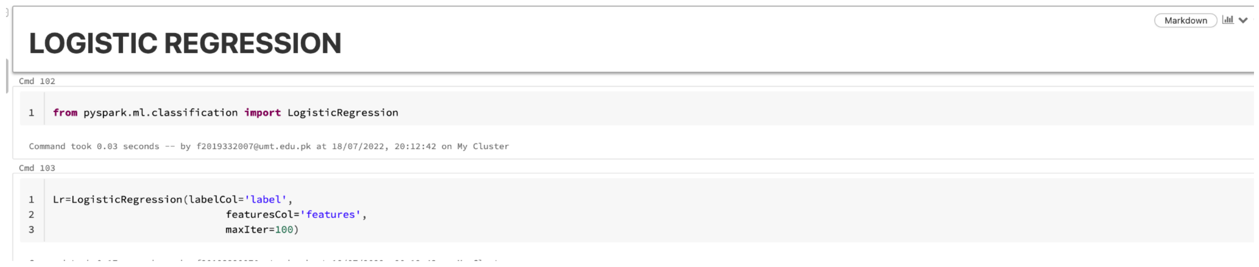
Test Accuracy of Decision Tree



Cmd 100

```
accuracy = evaluator.evaluate(pre)
print('Test Accuracy = ', accuracy)
```

```
▸ (1) Spark Jobs
Test Accuracy =  1.0
Command took 1.12 seconds -- by f2019332007@umt.edu.pk at 18/07/2022, 20:12:42 on My Cluster
```



LOGISTIC REGRESSION

Cmd 102

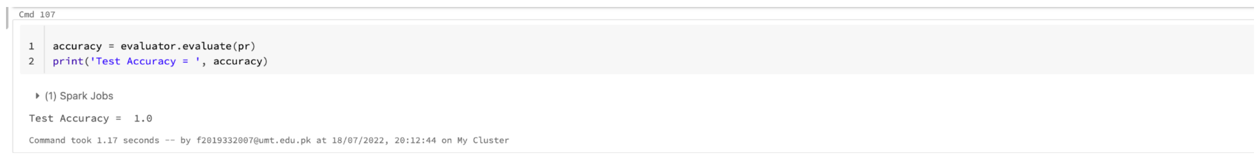
```
from pyspark.ml.classification import LogisticRegression
```

```
Command took 0.03 seconds -- by f2019332007@umt.edu.pk at 18/07/2022, 20:12:42 on My Cluster
```

Cmd 103

```
Lr=LogisticRegression(labelCol='label',
                      featuresCol='features',
                      maxIter=100)
```

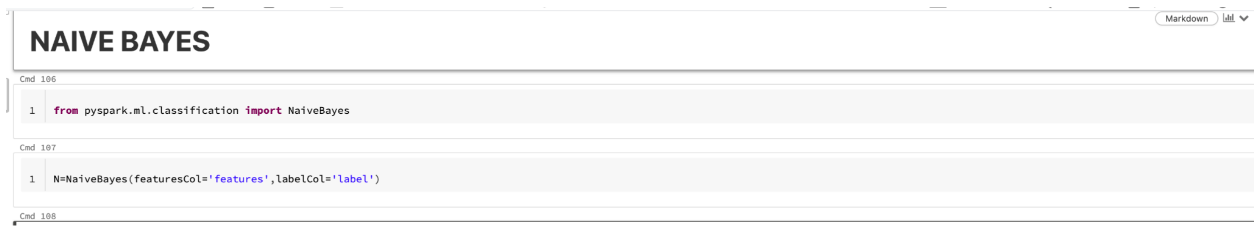
Test Accuracy of Logistic Regression



Cmd 107

```
accuracy = evaluator.evaluate(pr)
print('Test Accuracy = ', accuracy)
```

```
▸ (1) Spark Jobs
Test Accuracy =  1.0
Command took 1.17 seconds -- by f2019332007@umt.edu.pk at 18/07/2022, 20:12:44 on My Cluster
```



NAIVE BAYES

Cmd 106

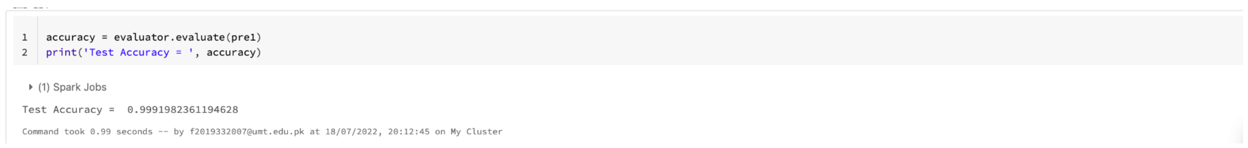
```
from pyspark.ml.classification import NaiveBayes
```

Cmd 107

```
N=NaiveBayes(featuresCol='features',labelCol='label')
```

Cmd 108

Test Accuracy of Naive Bayes

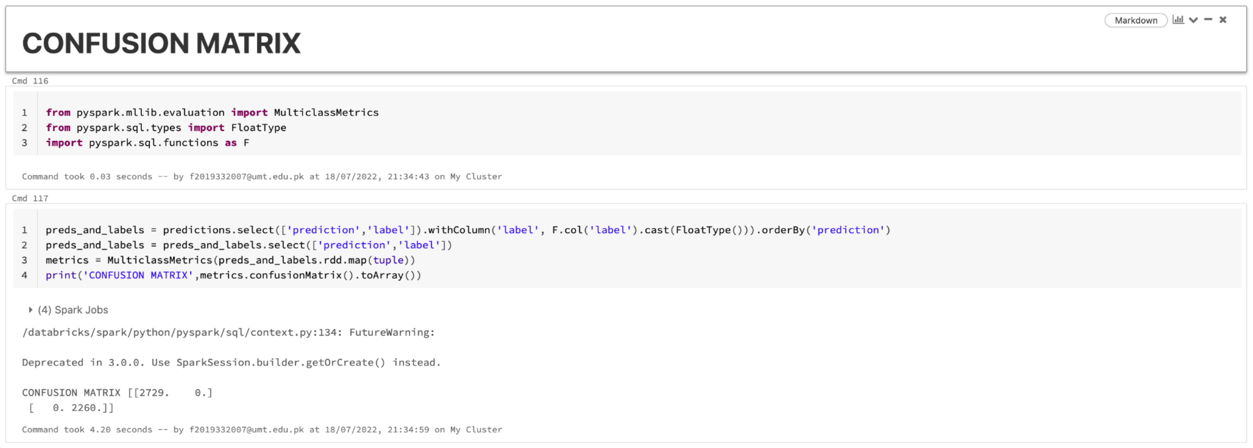


```
accuracy = evaluator.evaluate(pre1)
print('Test Accuracy = ', accuracy)
```

```
▸ (1) Spark Jobs
Test Accuracy =  0.9991982361194628
Command took 0.99 seconds -- by f2019332007@umt.edu.pk at 18/07/2022, 20:12:45 on My Cluster
```

Confusion Matrix:

Confusion matrix is a table that is used to define the performance of a classification algorithm. It visualizes and summarizes the performance of a classification algorithm.



CONFUSION MATRIX

Cmd 116

```
from pyspark.mllib.evaluation import MulticlassMetrics
from pyspark.sql.types import FloatType
import pyspark.sql.functions as F
```

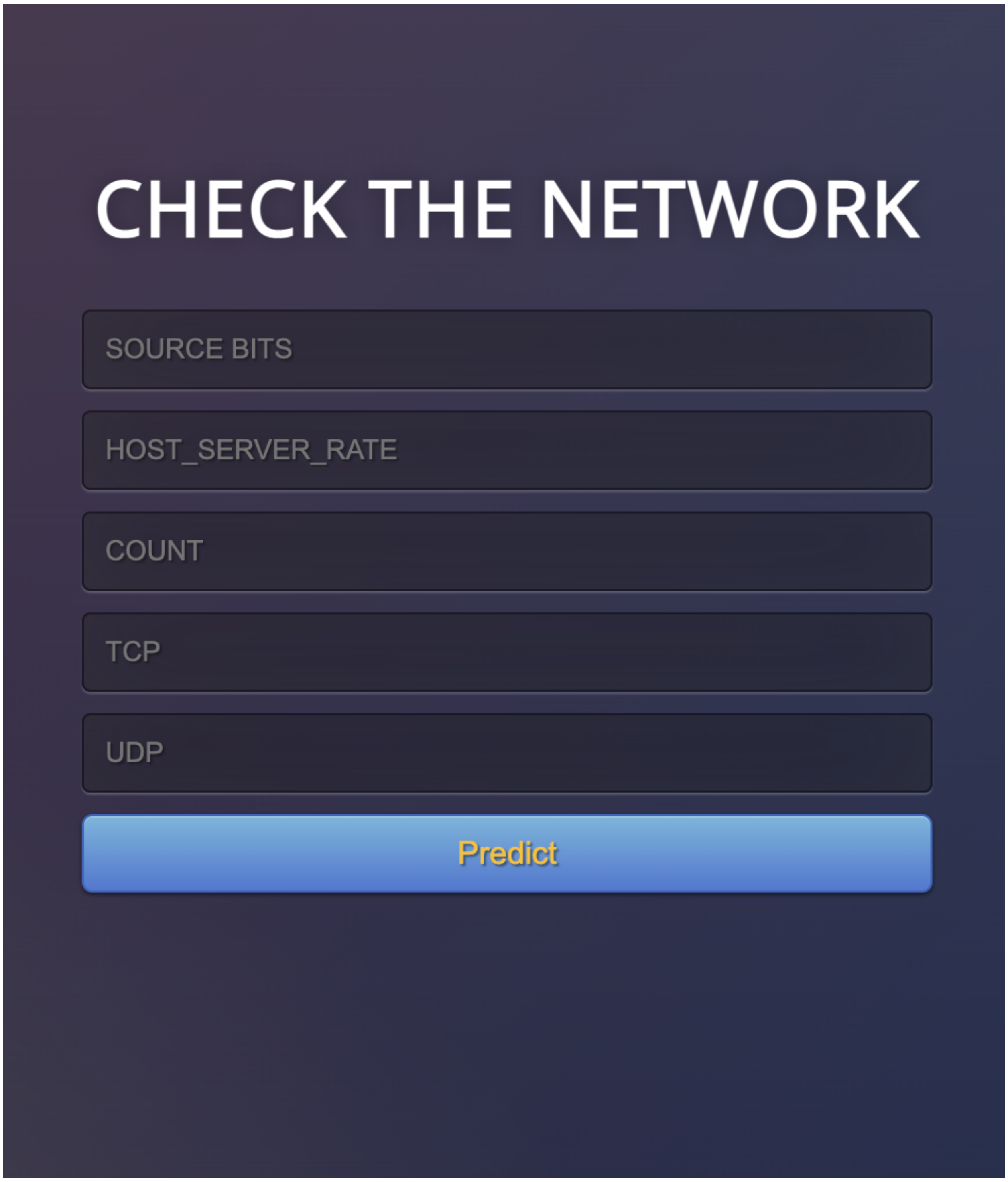
```
Command took 0.03 seconds -- by f2019332007@umt.edu.pk at 18/07/2022, 21:34:43 on My Cluster
```

Cmd 117

```
preds_and_labels = predictions.select(['prediction','label']).withColumn('label', F.col('label').cast(FloatType())).orderBy('prediction')
preds_and_labels = preds_and_labels.select(['prediction','label'])
metrics = MulticlassMetrics(preds_and_labels.rdd.map(tuple))
print('CONFUSION MATRIX',metrics.confusionMatrix().toArray())
```

```
▸ (4) Spark Jobs
/databricks/spark/python/pyspark/sql/context.py:134: FutureWarning:

Deprecated in 3.0.0. Use SparkSession.builder.getOrCreate() instead.

CONFUSION MATRIX [[2729.    0.]
 [   0. 2260.]]
Command took 4.20 seconds -- by f2019332007@umt.edu.pk at 18/07/2022, 21:34:59 on My Cluster
```

Deployment on Flask and Herokuapp


CHECK THE NETWORK
SOURCE BITS
HOST_SERVER_RATE
COUNT
TCP
UDP
Predict