Part A:

Question 1: What is an Intrusion Detection System? Is it possible to implement an Intrusion Detection System on this dataset? Explain the workflow as described in the paper for implementing Intrusion Detection System.

An intrusion detection system (IDS) is a system that monitors and analyzes data to detect any intrusion in the system or network. There are three methods of detecting attack, signature-based techniques (known attacks), anomaly-based detection (compares current user activities against predefined profiles), and hybrid-based detection which is a mix of the first two to reduce false positive rates and provide accurate IDS.

It is possible to implement an IDS on the KDDCUP99 dataset. However, due to the high volume and variety of the dataset, it is very difficult to detect attacks by traditional techniques. Big data techniques are required for efficient and accurate IDS implementation on this dataset. Spark-Chi-SVM model is used for intrusion detection on the KDDCUP99 dataset.

The workflow of the Spark-Chi-SVM model is as below:

1. Load KDDCUP99 dataset and export it into Resilient Distributed Datasets (RDD) and DataFrame in Apache Spark.
2. Data preprocessing. Large scale datasets contain noisy, redundant, and different types of data, which cannot be directly fed into machine learning algorithms (SVM can only deal with numerical data). Data preprocessing helps get the data ready for data analysis. Data preprocessing includes string indexer, encoding, and standardization. The string indexer and encoding convert categorical data into numerical data in which the data undergoes standardization, which is a key technique to obtain reliable results. Standardization uses the unit variance method to ensure the features with big numbers do not dominate thus the results are more reliable. This will improve the classification efficiency.
3. Feature selection. Redundant and irrelevant features slow down the processing speed and make no improvement on the model performance. ChiSqSelector is used for feature selection and used to reduce the dimensionality of the dataset and determine the optimal feature subset. Chi-Squared test is used as a test of feature independence and helps decides which features to select. The numTopFeatures method is used to determine the optimal number of features in terms of accuracy and efficiency.
4. Train the Spark-Chi-SVM model with the training dataset. Support Vector Machine (SVM) classifier (i.e. SVMWithSGD) is a supervised training method that analyzes data in use of classification and regression, and used to train the intrusion detection model to classify data into normal or attack. SVM is computationally expensive due to its high generalization and learning ability, however, Apache Spark can be used to reduce the execution time.
5. Test and evaluate the model with the KDD dataset. The training time, predict time, Area under curve (AUROC), and Area under Precision-Recall Curve (AUPR) are listed for three different classifiers: SVM only (without Chi-selector technique for features selection), Chi-SVM, and logistic regression (with Chi-selector technique for features selection). The model built with Chi-SVM classifier achieves the highest AUROC and AUPR with the shortest training and predict time.

Alexander Cheng – 1001634298

Question 2: Import Data

```
# Question 2
import urllib.request
urllib.request.urlretrieve("http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data_10_percent.gz", "/tmp/kddcup_data.gz")
dbutils.fs.mv("file:/tmp/kddcup_data.gz", "dbfs:/kdd/kddcup_data.gz")
display(dbutils.fs.ls("dbfs:/kdd"))
```

▸ (3) Spark Jobs

Table +

	path	name	size
1	dbfs:/kdd/kddcup_data.gz	kddcup_data.gz	2144903

Showing 1 row. | 2.47 seconds runtime

Question 3: Set data into RDDs

```
# Question 3
# Load data into RDD
path = "dbfs:/kdd/kddcup_data.gz"
rdd = sc.textFile(path)
# Print 10 values of the RDD
print(rdd.take(10))
```

▸ (1) Spark Jobs

['0,tcp,http,SF,181,5450,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,0,9,1.00,0.00,0.11,0.00,0.00,0.00,0.00,0.00,normal.', '0,tcp,http,SF,239,486,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,19,19,1.00,0.00,0.05,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.', '0,tcp,http,SF,235,1337,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,29,29,1.00,0.00,0.03,0.00,0.00,0.00,0.00,0.00,normal.', '0,tcp,http,SF,219,1337,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,6,6,0.00,0.00,0.00,0.00,1.00,0.00,0.00,39,39,1.00,0.00,0.03,0.00,0.00,0.00,0.00,0.00,normal.', '0,tcp,http,SF,217,2032,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,6,6,0.00,0.00,0.00,0.00,1.00,0.00,0.00,49,49,1.00,0.00,0.02,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal.', '0,tcp,http,SF,217,2032,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,6,6,0.00,0.00,0.00,0.00,1.00,0.00,0.00,59,59,1.00,0.00,0.02,0.00,0.00,0.00,0.00,0.00,normal.', '0,tcp,http,SF,212,1940,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,2,0.00,0.00,0.00,0.00,1.00,0.00,1.00,1,69,1.00,0.00,1.00,0.04,0.00,0.00,0.00,0.00,normal.', '0,tcp,http,SF,159,4087,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,5,5,0.00,0.00,0.00,0.00,1.00,0.00,0.00,11,79,1.00,0.00,0.09,0.04,0.00,0.00,0.00,0.00,0.00,0.00,normal.', '0,tcp,http,SF,210,151,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,8,89,1.00,0.00,0.12,0.04,0.00,0.00,0.00,0.00,normal.', '0,tcp,http,SF,212,786,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,8,99,1.00,0.00,0.12,0.05,0.00,0.00,0.00,0.00,normal.']

Command took 0.32 seconds -- by alexand.cheng@mail.utoronto.ca at 10/30/2022, 8:42:01 PM on Assign 3

Cmd 3

```
# Verify type od data structure of data
type(rdd)
```

Out[3]: pyspark.rdd.RDD

Command took 0.03 seconds -- by alexand.cheng@mail.utoronto.ca at 10/30/2022, 8:42:19 PM on Assign 3

Question 4: Split the Data

```
# Question 4
df = rdd.map(lambda x: x.split(",")).toDF()
```

▸ (1) Spark Jobs

▸ df: pyspark.sql.dataframe.DataFrame = [_1: string, _2: string ... 40 more fields]

Command took 0.41 seconds -- by alexand.cheng@mail.utoronto.ca at 10/30/2022, 8:42:22 PM on Assign 3

md 5

```
# Show total number of features (columns)
print("Total number of features:", len(df.columns))
```

Total number of features: 42

Command took 0.04 seconds -- by alexand.cheng@mail.utoronto.ca at 10/30/2022, 8:42:25 PM on Assign 3

md 6

```
# Print results
df.show()
```

▸ (1) Spark Jobs

_1	_2	_3	_4	_5	_6	_7	_8	_9	_10	_11	_12	_13	_14	_15	_16	_17	_18	_19	_20	_21	_22	_23	_24	_25	_26	_27	_28	_29	_30	_31	_32	_33	_34	_35	_36	_37	_38	_39	_40	_41	_42
0	tcp	http	SF	181	5450	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	8	8	0.00	0.00	0.00	0.00	1.00	0.00	0.00	9	9	1.00	0.00	0.11	0.00	0.00	0.00	0.00	normal.
0	tcp	http	SF	239	486	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	8	8	0.00	0.00	0.00	0.00	1.00	0.00	0.00	19	19	1.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00	normal.
0	tcp	http	SF	235	1337	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	8	8	0.00	0.00	0.00	0.00	1.00	0.00	0.00	29	29	1.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	normal.
0	tcp	http	SF	219	1337	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	6	6	0.00	0.00	0.00	0.00	1.00	0.00	0.00	39	39	1.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	normal.
0	tcp	http	SF	217	2032	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	6	6	0.00	0.00	0.00	0.00	1.00	0.00	0.00	49	49	1.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	normal.
0	tcp	http	SF	217	2032	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	6	6	0.00	0.00	0.00	0.00	1.00	0.00	0.00	59	59	1.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	normal.
0	tcp	http	SF	212	1940	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	2	0.00	0.00	0.00	0.00	1.00	0.00	1.00	1	69	1.00	0.00	1.00	0.04	0.00	0.00	0.00	0.00	normal.
0	tcp	http	SF	159	4087	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	5	5	0.00	0.00	0.00	0.00	1.00	0.00	0.00	11	79	1.00	0.00	0.09	0.04	0.00	0.00	0.00	0.00	normal.
0	tcp	http	SF	210	151	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	8	8	0.00	0.00	0.00	0.00	1.00	0.00	0.00	8	89	1.00	0.00	0.12	0.04	0.00	0.00	0.00	0.00	normal.
0	tcp	http	SF	212	786	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	8	8	0.00	0.00	0.00	0.00	1.00	0.00	0.00	8	99	1.00	0.00	0.12	0.05	0.00	0.00	0.00	0.00	normal.
0	tcp	http	SF	210	624	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	18	18	0.00	0.00	0.00	0.00	1.00	0.00	0.00	18	109	1.00	0.00	0.06	0.05	0.00	0.00	0.00	0.00	normal.
0	tcp	http	SF	177	1985	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0.00	0.00	0.00	0.00	1.00	0.00	0.00	28	119	1.00	0.00	0.04	0.04	0.00	0.00	0.00	0.00	normal.
0	tcp	http	SF	222	773	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	11	11	0.00	0.00	0.00	0.00	1.00	0.00	0.00	38	129	1.00	0.00	0.03	0.04	0.00	0.00	0.00	0.00	normal.
0	tcp	http	SF	256	1169	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	4	4	0.00	0.00	0.00	0.00	1.00	0.00	0.00	4	139	1.00	0.00	0.25	0.04	0.00	0.00	0.00	0.00	normal.
0	tcp	http	SF	241	259	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0.00	0.00	0.00	0.00	1.00	0.00	0.00	14	149	1.00	0.00	0.07	0.04	0.00	0.00	0.00	0.00	normal.
0	tcp	http	SF	260	1837	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	11	11	0.00	0.00	0.00	0.00	1.00	0.00	0.00	24	159	1.00	0.00	0.04	0.04	0.00	0.00	0.00	0.00	normal.
0	tcp	http	SF	241	261	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2	2	0.00	0.00	0.00	0.00	1.00	0.00	0.00	34	169	1.00	0.00	0.03	0.04	0.00	0.00	0.00	0.00	normal.
0	tcp	http	SF	257	818	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	12	12	0.00	0.00	0.00	0.00	1.00	0.00	0.00	44	179	1.00	0.00	0.02	0.03	0.00	0.00	0.00	0.00	normal.

Command took 0.60 seconds -- by alexand.cheng@mail.utoronto.ca at 10/30/2022, 8:42:30 PM on Assign 3

Alexander Cheng – 1001634298

Question 5: Extract the 6 columns and the label column

```
# Question 5
# Select first 6 features and label column
rdd2 = df.select('_1','_2','_3','_4','_5','_6', '_42').rdd
```

```
Command took 0.20 seconds -- by alexand.cheng@mail.utoronto.ca at 10/30/2022, 8:42:35 PM on Assign 3
```

```
Cmd 8
```

```
# Add header
df2 = rdd2.toDF(schema=['duration', 'protocol_type', 'service', 'flag', 'src_bytes', 'dst_bytes', 'label'])
```

```
▸ (1) Spark Jobs
▸ df2: pyspark.sql.dataframe.DataFrame = [duration: string, protocol_type: string ... 5 more fields]
Command took 0.60 seconds -- by alexand.cheng@mail.utoronto.ca at 10/30/2022, 8:42:38 PM on Assign 3
```

```
Cmd 9
```

```
# Change type from string to integer for columns duration, src_bytes, and dst_bytes
df2 = df2.withColumn("duration", df2.duration.cast('integer'))\
    .withColumn("src_bytes", df2.src_bytes.cast('integer'))\
    .withColumn("dst_bytes", df2.dst_bytes.cast('integer'))
```

```
▸ df2: pyspark.sql.dataframe.DataFrame = [duration: integer, protocol_type: string ... 5 more fields]
Command took 0.07 seconds -- by alexand.cheng@mail.utoronto.ca at 10/30/2022, 8:42:42 PM on Assign 3
```

```
Cmd 10
```

```
# Print schema
df2.printSchema()
```

```
root
 |-- duration: integer (nullable = true)
 |-- protocol_type: string (nullable = true)
 |-- service: string (nullable = true)
 |-- flag: string (nullable = true)
 |-- src_bytes: integer (nullable = true)
 |-- dst_bytes: integer (nullable = true)
 |-- label: string (nullable = true)

Command took 0.05 seconds -- by alexand.cheng@mail.utoronto.ca at 10/30/2022, 8:42:45 PM on Assign 3
```

```
1  # Display 10 values
2  df2.show(10)
```

▸ (1) Spark Jobs

```
+--------+-------------+-------+----+---------+---------+-------+
|duration|protocol_type|service|flag|src_bytes|dst_bytes|  label|
+--------+-------------+-------+----+---------+---------+-------+
|       0|          tcp|   http|  SF|      181|     5450|normal.|
|       0|          tcp|   http|  SF|      239|      486|normal.|
|       0|          tcp|   http|  SF|      235|     1337|normal.|
|       0|          tcp|   http|  SF|      219|     1337|normal.|
|       0|          tcp|   http|  SF|      217|     2032|normal.|
|       0|          tcp|   http|  SF|      217|     2032|normal.|
|       0|          tcp|   http|  SF|      212|     1940|normal.|
|       0|          tcp|   http|  SF|      159|     4087|normal.|
|       0|          tcp|   http|  SF|      210|      151|normal.|
|       0|          tcp|   http|  SF|      212|      786|normal.|
+--------+-------------+-------+----+---------+---------+-------+
only showing top 10 rows
```

Command took 1.01 seconds -- by alexand.cheng@mail.utoronto.ca at 10/30/2022, 8:42:51 PM on Assign 3

Question 6: Total number of connections in ascending order for protocol_type and service

```
1  # Question 6
2  # Protocol_type
3  df_protocol = df2.groupBy('protocol_type').count().orderBy('count', ascending=True)
4  df_protocol.show()
```

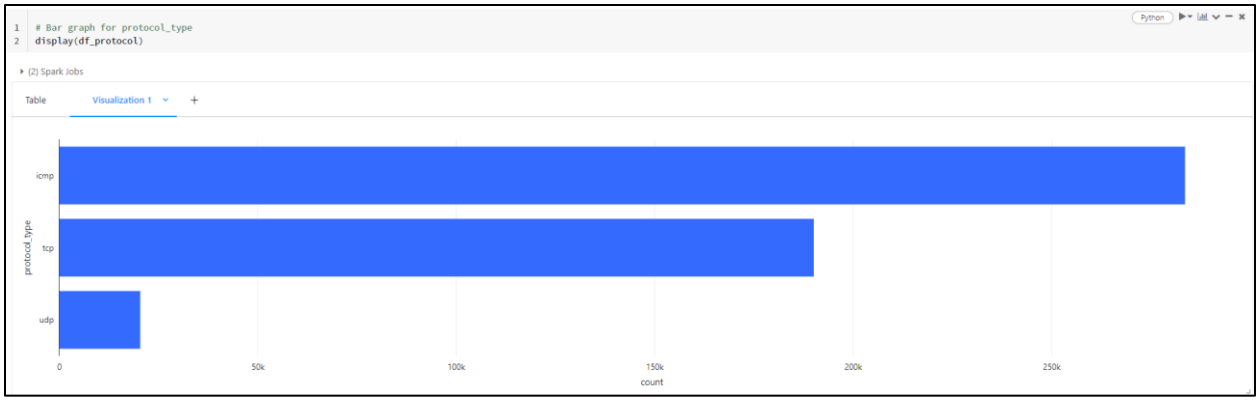
▸ (2) Spark Jobs

▸ df_protocol: pyspark.sql.dataframe.DataFrame = [protocol_type: string, count: long]

```
+-------------+------+
|protocol_type| count|
+-------------+------+
|          udp| 20354|
|          tcp|190065|
|         icmp|283602|
+-------------+------+
```

Command took 17.31 seconds -- by alexand.cheng@mail.utoronto.ca at 10/30/2022, 8:42:56 PM on Assign 3

Alexander Cheng – 1001634298



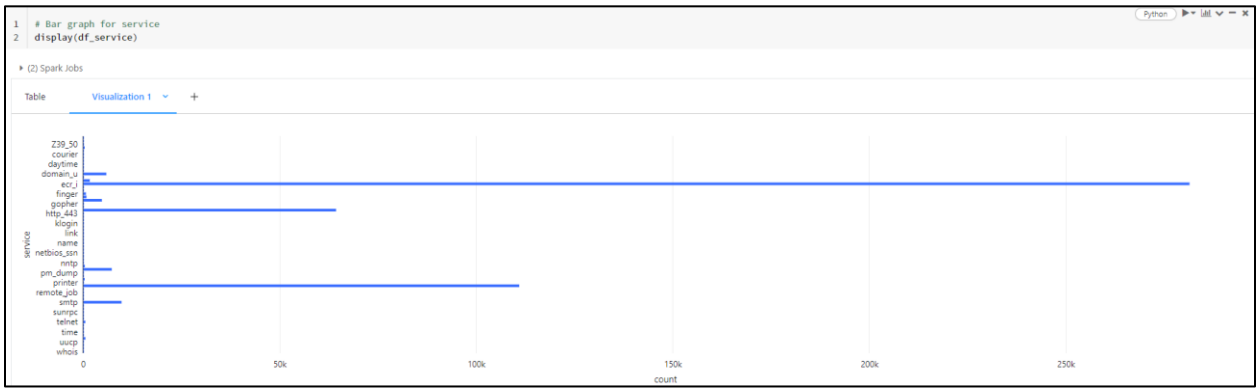

```
# Service
df_service = df2.groupBy('service').count().orderBy('count', ascending=True)
df_service.show()
```

(2) Spark Jobs

df_service: pyspark.sql.dataframe.DataFrame = [service: string, count: long]

```
----------+-----+
   service|count|
----------+-----+
     red_i|    1|
   pm_dump|    1|
    tftp_u|    1|
     tim_i|    7|
       X11|   11|
     urh_i|   14|
       IRC|   43|
    Z39_50|   92|
   netstat|   95|
       ctf|   97|
    kshell|   98|
      name|   98|
etbios_dgm|   99|
  http_443|   99|
      exec|   99|
      ldap|  101|
     pop_2|  101|
      link|  102|
```

mmand took 13.91 seconds -- by alexand.cheng@mail.utoronto.ca at 10/30/2022, 10:06:28 PM on Assign 3



Alexander Cheng – 1001634298

Question 7:

1. Average Duration and label activity type – Scatter Plot

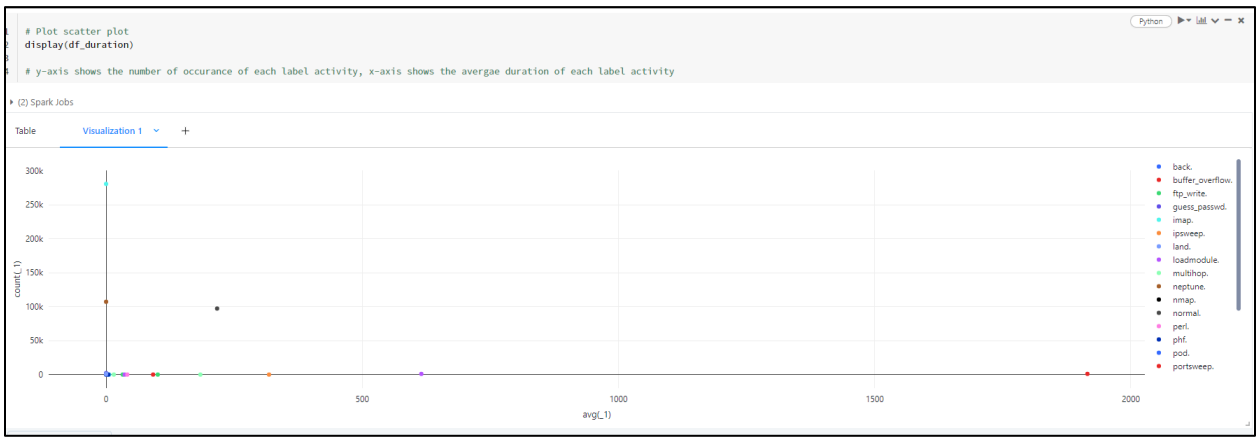
```
# Question 7 - Further exploratory data analysis

# Relationship between average duration and connection types
from pyspark.sql import functions as F
df_duration = df.groupBy("_42").agg(F.mean('_1'), F.count('_1')).orderBy('avg(_1)', ascending=False)
df_duration.show(20)
```

▸ (2) Spark Jobs

▸ df_duration: pyspark.sql.dataframe.DataFrame = [_42: string, avg(_1): double ... 1 more field]

_42	avg(_1)	count(_1)
portsweep.	1915.2990384615384	1040
warezclient.	615.2578431372549	1020
spy.	318.0	2
normal.	216.65732231336992	97278
multihop.	184.0	7
rootkit.	100.8	10
buffer_overflow.	91.7	30
perl.	41.333333333333336	3
loadmodule.	36.22222222222222	9
ftp_write.	32.375	8
warezmaster.	15.05	20
imap.	6.0	12
phf.	4.5	4
guess_passwd.	2.7169811320754715	53
back.	0.1289151157512483	2203
satan.	0.040276903713027064	1589
ipsweep.	0.034482758620689655	1247
smurf.	0.0	280790

Command took 14.68 seconds -- by alexand.cheng@mail.utoronto.ca at 10/30/2022, 8:45:01 PM on Assign 3



Portsweep label has a long average duration but does not occur often. Smurf and Neptune occurs very often but does not last for a long duration.

2. Average port_rate and label activity type:

```
# Relationship between average dst_host_same_src_port_rate and label types
df_dst_host_same_src_port_rate = df.groupBy("_42").agg(F.mean('_36'), F.count('_36')).orderBy('avg(_36)', ascending=False)
df_dst_host_same_src_port_rate.show(20)
```

▸ (2) Spark Jobs

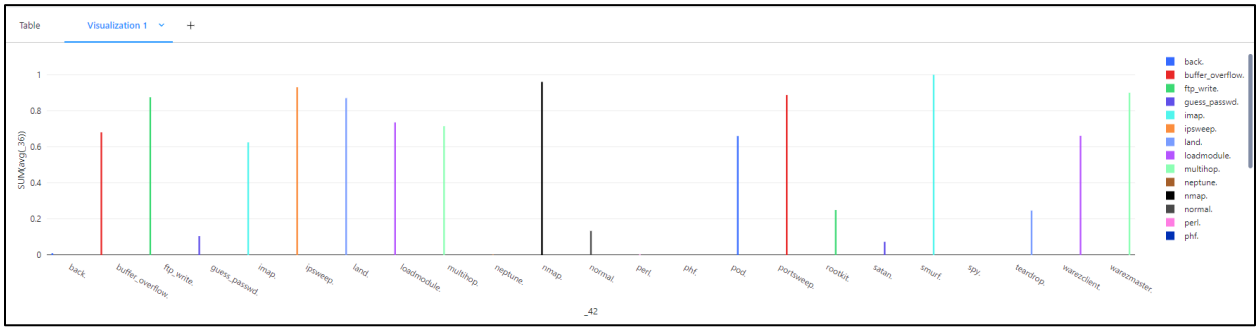
▸ df_dst_host_same_src_port_rate: pyspark.sql.dataframe.DataFrame = [_42: string, avg(_36): double ... 1 more field]

_42	avg(_36)	count(_36)
smurf.	0.9996929733964887	280790
nmap.	0.9597835497835498	231
ipsweep.	0.93028067361668	1247
warezmaster.	0.9	20
portsweep.	0.8873269230769232	1040
ftp_write.	0.875	8
land.	0.87	21
loadmodule.	0.7355555555555555	9
multihop.	0.7142857142857143	7
buffer_overflow.	0.6806666666666668	30
warezclient.	0.6613039215686269	1020
pod.	0.6597348484848486	264
imap.	0.625	12
rootkit.	0.25	10
teardrop.	0.2468335035750768	979
normal.	0.13395197269685152	97278
guess_passwd.	0.10528301886792447	53
satan.	0.07350534927627442	1589

Command took 13.41 seconds -- by alexand.cheng@mail.utoronto.ca at 10/30/2022, 8:47:35 PM on Assign 3

md 19

```
# Plot bar graph
display(df_dst_host_same_src_port_rate)

# y-axis shows the average port rate each label experienced, x-axis shows the top 20 label
```



Multiple label activities have an average dst_host_same_src_port_rate of just under 1.0. Smurf, nmap, and ipsweep have high port rates.

Alexander Cheng – 1001634298

Average duration and service type:

```
# Relationship between average duration and service
df_service_time = df.groupBy("_3").agg(F.mean('_1'), F.count('_1')).orderBy('avg(_1)', ascending=False)
df_service_time.show(10)
```

(2) Spark Jobs

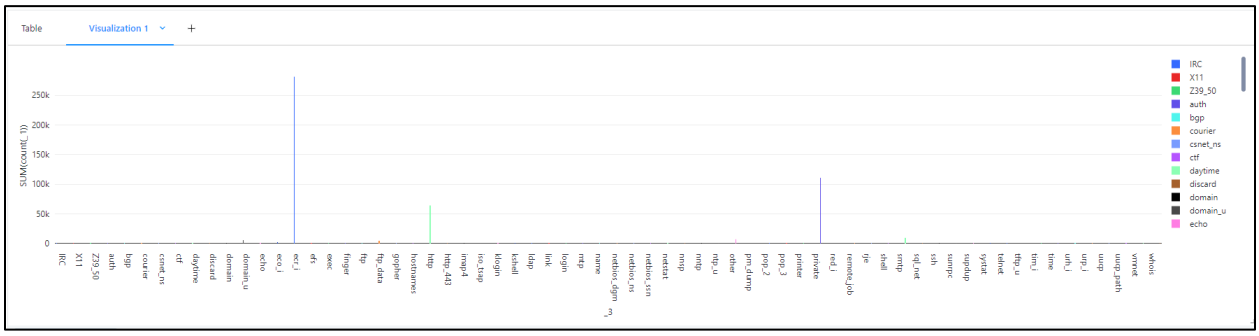
df_service_time: pyspark.sql.dataframe.DataFrame = [_3: string, avg(_1): double ... 1 more field]

```
+--------+------------------+----------+
|      _3|           avg(_1)|count(_1)|
+--------+------------------+----------+
|     IRC| 7343.093023255814|        43|
|   other|2815.9526046704436|      7237|
|  telnet| 890.2085769980507|       513|
|     ftp| 423.1390977443609|       798|
|  supdup|387.48571428571427|       105|
|csnet_ns|321.04761904761904|       126|
|     efs|299.36893203883494|       103|
|    uucp| 286.9622641509434|       106|
| printer| 283.8073394495413|       109|
| courier|283.50925925925924|       108|
+--------+------------------+----------+
only showing top 10 rows
```

Command took 13.78 seconds -- by alexand.cheng@mail.utoronto.ca at 10/30/2022, 8:55:16 PM on Assign 3

Cmd 21

```
# Plot bar graph
display(df_service_time)
# y-axis shows the average duration time, x-axis shows the top 20 services
```



We can observe that ecr_i, private, and http are the most common services with the most occurrences.

Alexander Cheng – 1001634298

Question 8:

```
# Question 8 - Preprocess data, stringindex categorical data, can also do onehot encoding but will not
from pyspark.ml.feature import StringIndexer
protocol_type_indexer = StringIndexer(inputCol="protocol_type",outputCol="protocol_type_index")
service_indexer = StringIndexer(inputCol="service",outputCol="service_index")
flag_indexer = StringIndexer(inputCol="flag",outputCol="flag_index")
protocol_type_indexed = protocol_type_indexer.fit(df2).transform(df2)
service_indexed = service_indexer.fit(df2).transform(protocol_type_indexed)
flag_indexed = flag_indexer.fit(df2).transform(service_indexed)

flag_indexed.show()
```

```
▸ (7) Spark Jobs
▸ protocol_type_indexed:  pyspark.sql.dataframe.DataFrame = [duration: integer, protocol_type: string ... 6 more fields]
▸ service_indexed:  pyspark.sql.dataframe.DataFrame = [duration: integer, protocol_type: string ... 7 more fields]
▸ flag_indexed:  pyspark.sql.dataframe.DataFrame = [duration: integer, protocol_type: string ... 8 more fields]
```

duration	protocol_type	service	flag	src_bytes	dst_bytes	label	protocol_type_index	service_index	flag_index
0	tcp	http	SF	181	5450	normal.	1.0	2.0	0.0
0	tcp	http	SF	239	486	normal.	1.0	2.0	0.0
0	tcp	http	SF	235	1337	normal.	1.0	2.0	0.0
0	tcp	http	SF	219	1337	normal.	1.0	2.0	0.0
0	tcp	http	SF	217	2032	normal.	1.0	2.0	0.0
0	tcp	http	SF	217	2032	normal.	1.0	2.0	0.0
0	tcp	http	SF	212	1940	normal.	1.0	2.0	0.0
0	tcp	http	SF	159	4087	normal.	1.0	2.0	0.0
0	tcp	http	SF	210	151	normal.	1.0	2.0	0.0
0	tcp	http	SF	212	786	normal.	1.0	2.0	0.0
0	tcp	http	SF	210	624	normal.	1.0	2.0	0.0
0	tcp	http	SF	177	1985	normal.	1.0	2.0	0.0
0	tcp	http	SF	222	773	normal.	1.0	2.0	0.0
0	tcp	http	SF	256	1169	normal.	1.0	2.0	0.0
0	tcp	http	SF	241	259	normal.	1.0	2.0	0.0
0	tcp	http	SF	260	1837	normal.	1.0	2.0	0.0
0	tcp	http	SF	241	261	normal.	1.0	2.0	0.0
0	tcp	http	SF	257	818	normal.	1.0	2.0	0.0

Need to preprocess the data, use stringindexer to convert categorical data to numerical data. Will focus on df2 (extracted columns from question 5). We can also perform one hot encoding but will choose not to keep it simple.

```
1  df_Q8 = flag_indexed.toPandas()
```

▸ (1) Spark Jobs

Command took 18.17 seconds -- by alexand.cheng@mail.utoronto.ca at 10/30/2022, 9:03:37 PM on Assign 3

Cmd 24

```
1  df_Q8 = df_Q8.drop('protocol_type', axis=1)
2  df_Q8 = df_Q8.drop('service', axis=1)
3  df_Q8 = df_Q8.drop('flag', axis=1)
```

Command took 0.18 seconds -- by alexand.cheng@mail.utoronto.ca at 10/30/2022, 9:04:02 PM on Assign 3

Cmd 25

```
1  df_Q8.loc[df_Q8["label"] != "normal.", "label"] = 0
2  df_Q8.loc[df_Q8["label"] == "normal.", "label"] = 1
```

Command took 0.20 seconds -- by alexand.cheng@mail.utoronto.ca at 10/30/2022, 9:04:05 PM on Assign 3

Dropped the categorical columns since we already converted to numerical.

```
1  target = df_Q8['label'].astype('float')
```

Command took 0.06 seconds -- by alexand.cheng@mail.utoronto.ca at 10/30/2022, 9:04:13 PM on Assign 3

Cmd 28

```
1  df_Q8 = df_Q8.drop('label', axis=1)
```

Command took 0.05 seconds -- by alexand.cheng@mail.utoronto.ca at 10/30/2022, 9:04:17 PM on Assign 3

Cmd 29

```
1  # Standardize data, can't have negative values in chi squared test
2  normalized_df_Q8=(df_Q8-df_Q8.min())/(df_Q8.max()-df_Q8.min())
```

Command took 0.61 seconds -- by alexand.cheng@mail.utoronto.ca at 10/30/2022, 9:04:37 PM on Assign 3

Cmd 30

```
1  normalized_df_Q8
```

	duration	src_bytes	dst_bytes	protocol_type_index	service_index	flag_index
0	0.0	2.610418e-07	0.001057	0.5	0.030769	0.0
1	0.0	3.446905e-07	0.000094	0.5	0.030769	0.0
2	0.0	3.389216e-07	0.000259	0.5	0.030769	0.0
3	0.0	3.158461e-07	0.000259	0.5	0.030769	0.0
4	0.0	3.129617e-07	0.000394	0.5	0.030769	0.0
...	...	...	...	...	...	...
494016	0.0	4.470881e-07	0.000365	0.5	0.030769	0.0
494017	0.0	4.067060e-07	0.000443	0.5	0.030769	0.0
494018	0.0	2.927706e-07	0.000233	0.5	0.030769	0.0
494019	0.0	4.196859e-07	0.000233	0.5	0.030769	0.0
494020	0.0	3.158461e-07	0.000239	0.5	0.030769	0.0

494021 rows × 6 columns

Command took 0.08 seconds -- by alexand.cheng@mail.utoronto.ca at 10/30/2022, 9:04:41 PM on Assign 3

Extract data target column and standardized the data.

```
### Split the data  ###
# Supposed to split the data first and then process the data, but will process data first to save time

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(normalized_df_Q8, target, test_size=0.3, random_state=0)
```

```
Command took 0.64 seconds -- by alexand.cheng@mail.utoronto.ca at 10/30/2022, 9:14:18 PM on Assign 3
```

```
Cmd 32
```

```
# Chi-squared method for feature selection, select the top 4 features

from sklearn import datasets
from sklearn.feature_selection import chi2
from sklearn.feature_selection import SelectKBest

test = SelectKBest(score_func=chi2, k=4)
fit = test.fit(X_train, y_train)
fit.scores_

# It seems like all features except for src_bytes is important
```

```
Out[35]: array([8.65145222e+02, 2.11540440e-02, 1.33523941e+02, 4.65547953e+04,
       2.77379051e+03, 8.95860598e+02])
```

```
Command took 0.31 seconds -- by alexand.cheng@mail.utoronto.ca at 10/30/2022, 9:14:25 PM on Assign 3
```

```
Cmd 33
```

```
X_new_train=test.fit_transform(X_train, y_train)
```

```
Command took 0.19 seconds -- by alexand.cheng@mail.utoronto.ca at 10/30/2022, 9:14:32 PM on Assign 3
```

Data was split into train/test, performed chi-squared method for feature selection. Found out at too late that we need to use the ML lib instead of Scikit learn. Will reduce the number of features used to improve efficiency. We can see that feature 2, src_bytes have a low chi-square value; therefore, feature is not significant in determining whether the label will be normal.

```
#Import svm model
from sklearn import svm

#Create a svm Classifier
clf = svm.SVC(kernel='linear', probability=True) # Linear Kernel

#Train the model using the training sets
clf.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

Alexander Cheng – 1001634298

Train using linear SVM model to predict/determine if the label will be normal or not normal, can use SVM with gradient descent for better results but will keep the model simple. We train the model using the training data and test against our test data.

```
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Accuracy: how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

# Model Precision: what percentage of positive tuples are labeled as such?
print("Precision:",metrics.precision_score(y_test, y_pred))

# Model Recall: what percentage of positive tuples are labelled as such?
print("Recall:",metrics.recall_score(y_test, y_pred))
```

```
Accuracy: 0.9779227701795462
Precision: 0.9559398443278273
Recall: 0.9306998594108974

Command complete
```

Can see that our model has an accuracy of 0.978, the precision is 0.956, and the recall is 0.931. We can probably improve the score if we decided to use gradient descent along with the SVM model. We can also improve the score if we decided to use other features than the ones extracted and then performed feature selection, but that's computationally expensive as well.

Part B:

Question 1:

1. A platform as a service (PaaS) solution that hosts web apps in Azure provides professional development services to continuously add features to custom applications.
    a. Yes, PaaS provides a framework with professional development services/tools that allows developers to customize or add features to cloud-based applications. The professional development service/tools include infrastructure (servers, storage, networking) as well as middleware, development tools, business intelligence services, database management systems, etc. It is designed to support the complete web application lifecycle (including updating). It also offers services such as workflow, directory security, and scheduling for apps.

2. A platform as a service (PaaS) database offering in Azure provides built in high availability.
    a. Yes, high availability is a general cloud feature which is included in an Azure PaaS database, Azure has a zone-redundant architecture, in which the platforms automatically replicate the resource and data across zones. So, if a failure ever occurs, the application can continue using the resource and data from the backup zone.

Questions 2-3:

2. A relational database must be used when:

D, Strong consistency guarantees are required. The schema of a relational database must be defined before any data can be added; therefore, the schema is not dynamic. Key-value pairs are stored in a non-relational database. Large images and videos are also stored in a non-relational database.

3. When you are implementing a Software as a Service solution, you are responsible for:

D, the user is only responsible for configuring the applications for his/her needs. The rest three are the responsibilities of the service provider.

Question 4:

1. To achieve a hybrid cloud model, a company must always migrate from a private cloud model
    a. No, a company can also start off as a public cloud and then combined with an on-premises infrastructure to create a hybrid cloud model
2. A company can extend the capacity of its internal network by using public cloud
    a. Yes, a company can extend the capacity of its internal network by configuring its cloud environment and connect the on-premises network to the cloud environment. This is much cheaper than expanding its existing on-premises infrastructure.
3. In a public cloud model, only guest users at your company can access the resources in the cloud
    a. No, anyone with an account in Azure Active Directory can be given access to the cloud resources

Alexander Cheng – 1001634298

Question 5:

a. A cloud service that remains available after a failure occurs: Fault tolerance, the ability to prevent failure and remain available after a failure occurs

b. A cloud service that can be recovered after a failure occurs: Disaster recovery, the ability to recover from a failure and to prevent the loss of data

c. A cloud service that performs quickly when demand increases: Dynamically Scalability, the ability to increase the capacity dynamically when the demand increases

d. A cloud service that can be accessed quickly from the internet: Low latency, the ability to provide service with minimum delay from the internet